論文 / 著書情報
Article / Book Information

題目(和文)	
Title(English)	Relaxation Approaches for Mixed-Integer Second-Order Cone Programming in Tree Breeding Problems
著者(和文)	Safarina Sena
Author(English)	Sena Safarina
出典(和文)	学位:博士(理学), 学位授与機関:東京工業大学, 報告番号:甲第11251号, 授与年月日:2019年9月20日, 学位の種別:課程博士, 審査員:山下 真,三好 直人,金森 敬文,福田 光浩,中野 張
Citation(English)	Degree:Doctor (Science), Conferring organization: Tokyo Institute of Technology, Report number:甲第11251号, Conferred date:2019/9/20, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

東京工業大学
Tokyo Institute of Technology

RELAXATION APPROACHES FOR MIXED-INTEGER SECOND-ORDER CONE PROGRAMMING IN TREE BREEDING PROBLEMS

Sena Safarina
Under Supervision of Associate Professor Makoto Yamashita

Submitted in partial fulfillment of the requirements for the degree of Doctor Philosophy

August, 2019

Abstract

An important application of Mixed-Integer Second-Order Cone Programming (MI-SOCP) is an optimal contribution selection (OCS) from tree breeding in which a key objective is to provide the best possible genetic contributions of genotypes under genetic diversity constraint. This project focuses on OCS in equal deployment problem (EDP) where the genetic contribution is fixed.

The EDP can be solved through mathematical optimization formulation. Since the genetic diversity can be formulated into a second-order cone constraint and the genetic contribution can be converted into an integer constraint, this problem can be considered as an MI-SOCP problem. However, the non-linearity embedded in MI-SOCP formulation induces a heavy computation.

To solve such MI-SOCP problems efficiently, in this paper, we examine conic relaxation approaches based on Linear Programming, Second-Order Cone Programming and Semidefinite Programming and propose a steepest-ascent method that utilizes the solution of conic relaxations as initial points. From the numerical experiments, we observed that our approach can provide favorable solutions in a short time. For exact solutions, we also discuss Lifted Polyhedral Programming approach with an active constraint selection method and a cone decomposition method that utilizes Lagrangian multiplier method.

Acknowledgements

I would like to express my gratitude towards my advisor Associate Professor Makoto Yamashita for the continuous support, supervising this project, and providing his invaluable advice and feedback throughout the academic years. I feel like having a second parent since his support is not only for my study, but also for my life in Japan.

My sincere gratitude is also toward Dr. Tim J. Mullin for fully supporting this project. His patience for understanding the mathematical term, sharing his knowledge about tree breeding, preparing the important data during his busiest time, encourage me a lot to finish my Ph.D.

I would be nothing without the support from Associate Professor Mituhiro Fukuda during seminars. I know that making mistakes is a learning process, but his patience while asking and sharing the invaluable knowledge give me better understanding during that process.

I appreciate a joint work with Associate Professor Satoko Moriguchi. I could not finish a part on the steep-ascent method without her valuable advice.

As a Japanese Government Scholarship Awardee, I am also very grateful to the Ministry of Education, Culture, Sports, Science and Technology (MEXT) for their fully support.

Finally, I wish to thank my family, especially my parents, for their love, encouragement, and support throughout this study and my life in general. I have sometimes lost motivation, however, their support gives much strength to make this thesis complete.

Contents

1 Introduction

The primary objective of this thesis is to propose numerical methods for solving mixed-integer second-order cone programming (MI-SOCP). The utilization of MI-SOCP has been increasing in a variety of important application areas; and recently it has been significant advances in solution approaches. For instance, Benson shows in his summary work [7] that MI-SOCP appears in variety aspects such as portfolio selection problem in financial engineering, pricing, network design, etc. Vielma [48] presents a relaxation approach to generate solution from an MI-SOCP problem.

An important application of MI-SOCP also arise from tree breeding. Our key focus is to develop efficient methods for MI-SOCP of such types in the field of tree breeding. For example, forest tree improvement is based on recurrent cycles of selection, mating and testing. In the selection phase, we should take genetic diversity into consideration, so that tree health and the potential for genetic gain in the future are conserved. A general objective of optimal contribution selection (OCS) [1, 27, 31, 55] is to maximize the total economic benefit under a genetic diversity constraint by determining the gene contribution to be made from each candidate. Based on the type of contribution, OCS problems can be classified into unequal and equal deployment problems. While an unequal deployment problem (UDP) does not require the same contribution for selected candidates, an equal deployment problem (EDP) stipulates that a specified number of selected individuals must contribute equally to the gene pool. EDP is the main target of this thesis.

Before discussing EDP, we introduce a mathematical optimization formulation for UDP given by Meuwissen [28], since UDP has a simpler structure than EDP. The formulation due to Meuwissen is as follows:

$$\begin{array}{lcl} \text{maximize} & : & \boldsymbol{g}^T\boldsymbol{x} \\ \text{subject to} & : & \boldsymbol{e}^T\boldsymbol{x} = 1,\ \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u},\ \boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} \leq 2\theta. \end{array} \tag{1.1}$$

The decision variable is $\boldsymbol{x} \in \mathbb{R}^m$ that corresponds to the gene contributions of individual candidates, where m is the total number of candidates. The objective is to maximize the total benefit $\boldsymbol{g}^T\boldsymbol{x}$ where the vector $\boldsymbol{g} = (g_1, g_2, \ldots, g_m)^T$ contains the estimated breeding values (EBVs) [26] representing the genetic value of candidates in $\boldsymbol{x}$. Throughout this thesis, we assume that $\boldsymbol{g}$ is given. Using a vector of ones $\boldsymbol{e} \in \mathbb{R}^m$, the first constraint $\boldsymbol{e}^T\boldsymbol{x} = 1$ requires that the total contribution of all candidates be unity. The second constraint is composed by a lower bound $\boldsymbol{l} \in \mathbb{R}^m$ and an upper bound $\boldsymbol{u} \in \mathbb{R}^m$.

The crucial constraint in (1.1) is $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} \leq 2\theta$ that requires the group coancestry $\frac{\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}}{2}$ be under an appropriate level $\theta \in \mathbb{R}_{++}$, where $\mathbb{R}_{++}$ is the set of positive real numbers. Group coancestry $\frac{\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}}{2}$ was originally introduced by Cockerham [12], and defined as

the probability that two genes sampled randomly from a population are identical by descent (IBD), i.e., have a common ancestor. Group coancestry can be derived from the numerator relationship matrix $\boldsymbol{A} \in \mathbb{R}^{m\times m}$ proposed earlier by Wright [52], where each element A_{ij} in the matrix $\boldsymbol{A}$ is the probability that genotypes i and j will carry genes at any given chromosome location that are IBD. If group coancestry $\frac{\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}}{2}$ is too high, the close relatedness among individuals in the population will cause genetic diversity to be lower, so that the long-term performance would be depreciated. Pong-Wong and Woolliams [36] observed that the matrix $\boldsymbol{A}$ is always positive definite, and they formulated the UDP as a semi-definite programming (SDP) problem. Their SDP approach gave the exact optimal value to the UDP for the first time, but Ahlinder et al. [1] showed that the computation cost of the SDP approach was very high, even when using a parallel SDP solver [53, 54]. To reduce the heavy computation burden, Yamashita et al. [55] proposed an efficient numerical method based on second-order cone programming (SOCP) [4].

This thesis is mainly concerned with EDP of form:

$$\begin{array}{lcl} \text{maximize} & : & \boldsymbol{g}^T\boldsymbol{x} \\ \text{subject to} & : & \boldsymbol{e}^T\boldsymbol{x} = 1,\ x_i \in \left\{0, \frac{1}{N}\right\}\ (i = 1, \ldots, m),\ \boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} \leq 2\theta. \end{array} \tag{1.2}$$

We should emphasize that the simple bound $\boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}$ in the UDP is replaced by another constraint $x_i \in \left\{0, \frac{1}{N}\right\}$ to require an equal contribution from each chosen candidate. Here, N is the parameter to indicate the number of chosen candidates. In short, we have to choose exactly N individuals from a list of m available candidates in the EDP. Through this thesis, we assume that (1.2) is feasible.

The OCS problem has been widely solved through a software package `GENCONT` developed by Meuwissen [27]. The numerical method implemented in `GENCONT` is based on Lagrange multipliers, but it forcibly fixes variables that exceed lower or upper bounds $\left(0 \leq x_i \leq \frac{1}{N}\right)$ at the corresponding lower and upper bound. Thus, even though `GENCONT` generates a solution quickly, the solution is often suboptimal. To resolve this difficulty in `GENCONT`, another tool `dsOpt`, incorporated in the software package `OPSEL` [30], was proposed by Mullin and Belotti [31]. `dsOpt` is an implementation of the branch-and-bound method combined with an outer approximation method [15]. This implementation was designed to acquire exact optimal solutions, but `dsOpt` generates a huge number of subproblems in the framework of branch-and-bound, so that computing the solution takes a long time. Hence, there has been a strong desire for a different approach to solve the EDP in a more practical time.

In contrast to existing implementations [27, 31], this thesis focuses on the fact that the crucial quadratic constraint $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} \leq 2\theta$ in (1.1) and (1.2) can be described as a second-order cone $\left(\sqrt{2\theta}, \boldsymbol{U}\boldsymbol{x}\right) \in \mathcal{K}^m$. The matrix $\boldsymbol{U}$ is the Cholesky factorization of $\boldsymbol{A}$ such that $\boldsymbol{A} = \boldsymbol{U}^T\boldsymbol{U}$. Throughout this thesis, we use $\mathcal{K}^m$ to denote the $(m+1)$-dimensional second-order cone:

$$\mathcal{K}^m = \{(v_0, \boldsymbol{v}) \in \mathbb{R}_+ \times \mathbb{R}^m : ||\boldsymbol{v}||_2 \leq v_0\}.$$

Here, $\mathbb{R}_+$ is the set of nonnegative real numbers. Introducing a new variable $\boldsymbol{y} = N\boldsymbol{x}$,

we convert the EDP (1.2) into an MI-SOCP formulation:

$$\begin{array}{lll} \text{maximize} & : & \frac{\boldsymbol{g}^T\boldsymbol{y}}{N} \\ \text{subject to} & : & \boldsymbol{e}^T\boldsymbol{y} = N,\ y_i \in \{0,1\}\ (i=1,\ldots,m),\ \left(\sqrt{2\theta}N, \boldsymbol{U}\boldsymbol{y}\right) \in \mathcal{K}^m. \end{array} \tag{1.3}$$

The main difficulty in this MI-SOCP formulation is the non-linearity arising from the second-order cone, and this leads to a heavy computation cost. Recently, many approaches have been proposed to solve general MI-SOCP formulations [7, 5, 14, 11], but these cannot directly exploit the structure of the matrices $\boldsymbol{A}$ or $\boldsymbol{U}$. The proposed approach in [55] can exploit the structure of $\boldsymbol{A}$, but the approach is designed for solving the UDP. Thus we cannot simply apply the approach in [55] to (1.3) and we consider other approaches for efficiently solving (1.3).

Our thesis is to examine three main approaches.

1. Conic relaxation (Chapters 3) in a combination with a steep-descent method (Chapter 4).
2. Lifted Polyhedral Programming and its improvement (Chapter 5).
3. Cone Decomposition Method (Chapter 6).

We firstly propose a conic relaxation for the EDP. One of conic relaxation applications can be found in a well-known paper due to Goemans and Williamson [19] which applied SDP to max-cut problems. They removed the rank-one constraint in a matrix formulation of max-cut problems to convert its nonconvex feasible set into the space of positive semidefinite matrices. In other words, they derived an SDP problem by ignoring the rank-one constraint. In particular, this SDP relaxation can lead to favorable approximate solutions. Following this achievement, the SDP relaxation approach has been widely applied to combinatorial optimization problems [51]. This research direction has been extended to conic relaxation with the use of LP, SOCP and SDP.

Discrete convex optimization has another relevant research direction. A convex function in continuous space can be considered as a discrete convex function if there is variable space restriction to the integer points. However, this intuition is not appropriate, since such a function does not always inherit useful convex function properties. Discrete convexity needs some deep combinatorial or discrete-mathematical considerations. The discrete convex analysis theory [33] explains two main convexity concepts, L-convexity and M-convexity. If a function is an M-convex function, a steepest-descent method proposed in [34] can find its global minimum.

Since relaxation problems give good approximation of the optimal value, our first goal in this thesis is to acquire a favorable solution that is available in a practical computation time.

To achieve this goal, we develop a steepest-ascent method that employs the solution obtained from the conic relaxation problems as a starting point. The usual steepest-descent method [34] minimizes an M-convex objective function on a particular feasible set. Since the EDP is a maximization problem, we consider a steepest-ascent method instead of a steepest-descent method. We embed the quadratic constraint $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} \leq 2\theta$ into the objective function as a penalty term with a weight computed from the Lagrange multiplier. This new objective function is not an M-concave function, so we cannot

guarantee that the solution obtained by the proposed steepest-ascent method is a global solution of the EDP. However, through numerical experiments, we observe that the steepest-ascent method generates qualified solutions for the EDP. In particular, the steepest-ascent method starting with the SOCP relaxation problem attains the best performance among the LP, SOCP, and SDP relaxation problems. It even performs better than existing methods like `GENCONT` and `dsOpt` from the viewpoints of both solution quality and computation time. For instance, the result of EDP with the largest size of candidates $m = 15222$ presents that the steepest-ascent method starting with SOCP reduces computation time into 4.72 seconds while `GENCONT` fails due to out of memory and `dsOpt` requires more than 3 hours to obtain the solution.

As the second approach, we consider lifted polyhedral programming (LPP) relaxation [48], which solves mixed-integer conic quadratic problems that have similar structure with our EDP problem. Since LPP makes the use of hyperplanes to approximate conic quadratic constraint, LPP needs a large number of hyperplanes to obtain an optimal solution. However, the numerical result shows that a larger number of constraints generated in LPP demands a heavy computation cost. Exploiting an extension of polyhedral programming relaxation for SOCP problems [6, 8, 48], we improve LPP by utilizing active constraint selection method (LPP-ASCM) that can removes the non-linearity. Out of expectation, LPP-ASCM somehow performed worse computation time than LPP itself, for example, for $m = 1050$ with the setting gap $= 1\%$, while LPP required almost 18 minutes, LPP-ACSM prematurely terminated due to out of memory.

Our last approach is a cone decomposition method (CDM) that is based on a geometric cut in a combination with a Lagrangian multiplier method. A geometric cut is a cutting plane such that the plane is computed with an orthogonal projection [9]. Cone decomposition itself was already used in `CPLEX`, but `CPLEX` used an outer approximation. Therefore, the proposed CDM generates a different linear approximation. Using the sparsity found in the inverse of Wright's numerator relationship matrix $\boldsymbol{A}^{-1}$, the proposed CDM successfully reduces heavy computation burden. This sparsity exploitation is the key property developed in [55] (See also Section 6.2). This sparsity and the geometric cuts are combined into a sparse linear approximation, and it strongly enhances the performance of CDM.

In addition, we prove that the Lagrangian multiplier method in the framework of CDM gives an analytical form for the geometric cut, therefore, the proposed CDM and and its sparse variant generate the linear cuts without relying on iterative methods. Comparing all methods through numerical results, CDM with sparse structure gives significant performance. For example, in the largest size of EDP $m = 15222$ with $N = 100$ and the tight gap $= 1\%$, the `dsOpt` and the other proposed method (LPP) encounter the insufficient memory problem, but CDM with sparse structure manage to solve the instance in 47 seconds, which is 10 times faster than CDM itself.

The remaining of this thesis is organized as follow. In Chapter 2, we present theoretical concepts such as SDP, SOCP, MI-SOCP, and LPP, as preliminaries. We explain a variety of conic relaxation for EDP and show their result in Chapter 3. Chapter 4 discuss a steepest-ascent method, and Chapter 5 focuses lifted polyhedral programming with an active selection. In Chapter 6, we consider the cone decomposition methods. Finally, the conclusion of this thesis can be found in Chapter 7.

2 Background

In this chapter, we present several aspects that are employed in this thesis for solving the optimal contribution selection from tree breeding problem in both unequal and equal deployment.

2.1 SDP Formulation

2.1.1 Semidefinite Program

Semidefinite program (SDP) is a conic programming optimization, which has wide-range applications in many fields like control theory, graph theory, and combinatorial optimization (See, for example, [3, 51]). Before going through the SDP formulation in the context of the EDP, let us first consider key properties of SDPs.

Definition 2.1.1. *The set of order n symmetric matrices is defined by*

$$\mathcal{S}^n = \{\boldsymbol{F} \in \mathbb{R}^{n\times n} : f_{ij} = f_{ji},\ 1 \le i < j \le n\}.$$

Definition 2.1.2. *A symmetric matrix $\boldsymbol{F}$ is positive semidefinite if $\boldsymbol{x}^T\boldsymbol{F}\boldsymbol{x} \ge 0$ for all $\boldsymbol{x} \in \mathbb{R}^n$, and positive definite if $\boldsymbol{x}^T\boldsymbol{F}\boldsymbol{x} > 0$ for all $\boldsymbol{x} \in \mathbb{R}^n\backslash\{0\}$.*

The set of order n positive semidefinite (positive definite) matrices is usually denoted by $\mathcal{S}^n_+ \subset \mathcal{S}^n$ ($\mathcal{S}^n_{++} \subset \mathcal{S}^n$, respectively). A matrix $\boldsymbol{F}$ is also denoted as $\boldsymbol{F} \succeq 0$ ($\boldsymbol{F} \succ 0$) if $\boldsymbol{F}$ is positive semidefinite (positive definite, respectively).

Fact 2.1.3. *[17] For $\boldsymbol{F} \in \mathcal{S}^n$, the following statements are equivalent:*

1. *$\boldsymbol{F}$ is positive semidefinite matrix*
2. *All the eigenvalues of $\boldsymbol{F}$ are non-negative*
3. *There exist an upper-triangular matrix $\boldsymbol{U} \in \mathbb{R}^{n\times n}$ such that $\boldsymbol{F} = \boldsymbol{U}^T\boldsymbol{U}$.*

In particular, the matrix $\boldsymbol{U}$ in the last statement is called *Cholesky factorization*. This factorization is often employed for formulating an optimal selection problem into second-order cone program. In addition, the following concept is also very useful for showing positive semi-definiteness of a matrix [44].

Fact 2.1.4. *Suppose a symmetric matrix*

$$\boldsymbol{F} = \begin{pmatrix} \boldsymbol{C} & \boldsymbol{D} \\ \boldsymbol{D}^T & \boldsymbol{E} \end{pmatrix}$$

with $\boldsymbol{C}$ and $\boldsymbol{E}$ symmetric matrices in appropriate dimensions and $\boldsymbol{C} \succ 0$. Then

$$\boldsymbol{F} \succeq 0 \quad (\succ 0) \text{ if and only if } \boldsymbol{E} - \boldsymbol{D}^T\boldsymbol{C}^{-1}\boldsymbol{D} \succeq 0 \quad (\succ 0).$$

The matrix $\boldsymbol{E} - \boldsymbol{D}^T\boldsymbol{C}^{-1}\boldsymbol{D}$ is called the Schur complement of $\boldsymbol{C}$ in $\boldsymbol{F}$.

Understanding the properties of semidefinite program, we can now formulate an SDP problem.

Definition 2.1.5. *A standard form (P) of semidefinite programs and its dual (D) are given by:*

$$\begin{array}{lll} (P) & \textit{minimize} & : \ \boldsymbol{C} \bullet \boldsymbol{X} \\ & \textit{subject to} & : \ \boldsymbol{F}_k \bullet \boldsymbol{X} = f_k \ (k = 1, ..., m), \ \boldsymbol{X} \succeq 0. \end{array} \tag{2.1}$$

$$\begin{array}{lll} (D) & \textit{maximize} & : \ \sum_{k=1}^{m} f_k y_k \\ & \textit{subject to} & : \ \sum_{k=1}^{m} \boldsymbol{F}_k y_k + \boldsymbol{Z} = \boldsymbol{C}, \ \boldsymbol{Z} \succeq 0. \end{array} \tag{2.2}$$

where $f_k \in \mathbb{R}^m, \boldsymbol{F}_1, \ldots, \boldsymbol{F}_m \in \mathcal{S}^n$ and $\boldsymbol{C} \in \mathcal{S}^n$ are the input data; $\boldsymbol{X} \in \mathcal{S}^n$ is the primal variable, while $(\boldsymbol{y}, \boldsymbol{Z}) \in \mathbb{R}^m \times \mathcal{S}^n$ is the dual variable in which $\boldsymbol{Z}$ is the slack matrix.

2.1.2 An example: SDP relaxation for the max-cut problem

Here, we present an example to illustrate a distinguished concept of SDP problem based on the SDP relaxation for the max-cut problem [19]. First, assume that we have an undirected graph $\boldsymbol{G} = (N, E)$, where $N = \{1, 2, \ldots, n\}$ is the set of n nodes and E the set of edges. Now, we define non-negative weights $\boldsymbol{w} \in \mathbb{R}^E_+$, that is w_{ij} for $(i, j) \in E$ and $\delta(K) := \{(i, j) \in E : i \in K, j \notin K\}$. Here, $K(\subseteq N)$ is a cut of a graph, and $(i, j) \in E$ is an edge to link nodes i and j.

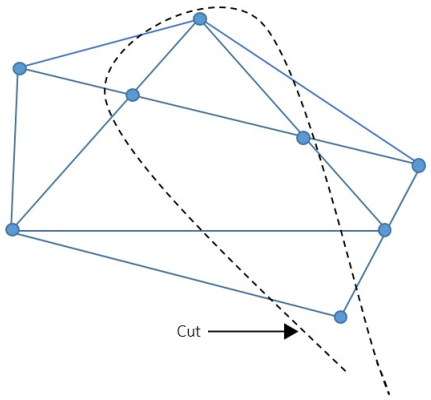

Figure 2.1: An example of max-cut problem

The purpose of the max-cut problem is to find a cut that attains the maximum weight $w(\delta(K)) := \sum_{(i,j)\in\delta(k)} w_{ij}$. Let

$$\begin{cases} x_i = 1 & \text{if } i \in K, \\ x_i = -1 & \text{if } i \notin K, \end{cases}$$

for $\boldsymbol{x} \in \mathbb{R}^n$ to represent the cut $\delta(K)$. Then, it is clear that $x_i x_j = -1$ if $(i, j) \in \delta(K)$ and $x_i x_j = 1$ if $(i, j) \notin \delta(k)$. Then, we have

$$w(\delta(K)) = \frac{1}{2} \sum_{i<j} w_{ij}(1 - x_i x_j) = \frac{1}{4} \sum_i \sum_j w_{ij}(1 - x_i x_j).$$

Defining $\boldsymbol{C} \in \mathcal{S}^n$ by:

$$C_{ij} = \begin{cases} -\frac{w_{ij}}{4} & \text{for } i \neq j \\ \sum_j \frac{w_{ij}}{4} & \text{for } i = j, \end{cases}$$

we can write the weight by $w(\delta(k)) = \boldsymbol{x}^T \boldsymbol{C} \boldsymbol{x}$.

Thus, the max-cut problem can now be written as in the following formulation:

$$\begin{array}{lll} \text{maximize} & : & \boldsymbol{x}^T \boldsymbol{C} \boldsymbol{x} \\ \text{subject to} & : & x_i^2 = 1 \text{ for } i = 1, ..., n, \end{array}$$

and it is equivalent to

$$\begin{array}{lll} \text{maximize} & : & \boldsymbol{x}^T \boldsymbol{C} \boldsymbol{x} \\ \text{subject to} & : & X_{ii} = 1,\ i = 1, ..., n, \\ & & \boldsymbol{X} \succeq 0,\ \text{rank}(\boldsymbol{X}) = 1. \end{array}$$

This problem is NP-hard due to the rank constraint. We can relax this formulation by ignoring the rank constraint into an SDP formulation as follows:

$$\begin{array}{lll} \text{maximize} & : & \boldsymbol{C} \bullet \boldsymbol{X} \\ \text{subject to} & : & X_{ii} = 1,\ i \in N, \\ & & \boldsymbol{X} \succeq 0. \end{array}$$

It is known that this SDP problem gives a good approximate value for the max-cut problem. For the details of this approximation, refer to [19].

2.1.3 Interior-Point Algorithm

In this section, we briefly present the primal-dual path-following algorithm; one of several algorithms for solving SDP problems. This algorithm is a type of interior-point methods discussed in Nesterov and Nemirovski [35].

Recalling the definitions of the standard SDP (2.1) and its dual (2.2), we define the sets of interior feasible solutions of the primal (P) and the dual (D) as follow:

$$\begin{aligned} \mathcal{F}^0(P) &= \{\boldsymbol{X} \in \mathcal{S}^n : \boldsymbol{F}_k \bullet \boldsymbol{X} = f_k (k = 1 \ldots, m),\ \boldsymbol{X} \succ 0\}, \\ \mathcal{F}^0(D) &= \left\{ (\boldsymbol{y}, \boldsymbol{Z}) \in \mathbb{R}^m \times \mathcal{S}^m : \sum_{k=1}^m y_k \boldsymbol{F}_k + \boldsymbol{Z} = \boldsymbol{C},\ \boldsymbol{Z} \succ 0 \right\}. \end{aligned}$$

Assuming $\mathcal{F}^0(P)$ and $\mathcal{F}^0(D)$ are nonempty, we can get the optimal solutions for both primal and dual problem, $\boldsymbol{X}^*$ and $(\boldsymbol{y}^*, \boldsymbol{Z}^*)$ such that $\boldsymbol{C} \bullet \boldsymbol{X}^* = \boldsymbol{f}^T \boldsymbol{y}^*$ due to the duality theorem [44].

For feasible solutions $\boldsymbol{X}$ and $(\boldsymbol{y}, \boldsymbol{Z})$ of the primal and dual pair, it holds that

$$\boldsymbol{C} \bullet \boldsymbol{X} - \boldsymbol{f}^T \boldsymbol{y} = \left(\sum_{k=1}^m \boldsymbol{F}_k \boldsymbol{y}_k + \boldsymbol{Z} - \boldsymbol{f}^T \boldsymbol{y} \right) = \boldsymbol{X} \bullet \boldsymbol{Z} + \sum_{k=1}^m \boldsymbol{y}_k (\boldsymbol{F}_k \bullet \boldsymbol{X}) - \boldsymbol{f}^T \boldsymbol{y} = \boldsymbol{X} \bullet \boldsymbol{Z}.$$

Hence we can alternatively express $\boldsymbol{C} \bullet \boldsymbol{X} = \boldsymbol{f}^T \boldsymbol{y}$ by $\boldsymbol{X} \bullet \boldsymbol{Z} = 0$.

We outline a framework of primal-dual interior-point algorithm for solving the pair of SDPs.

Algorithm 2.1.6. [A framework of interior-point methods [23]]

Step 1 Set $\boldsymbol{X}^0 \in \mathcal{F}^0(P)$, $(\boldsymbol{y}^0, \boldsymbol{Z}^0) \in \mathcal{F}^0(D)$, and the iteration counter $r = 0$.

Step 2 If $(\boldsymbol{X}^r, \boldsymbol{y}^r, \boldsymbol{Z}^r)$ satisfies a stopping criterion, stop and output $(\boldsymbol{X}^r, \boldsymbol{y}^r, \boldsymbol{Z}^r)$ as a solution.

Step 3 Let $\boldsymbol{X} = \boldsymbol{X}^r$, $(\boldsymbol{y}, \boldsymbol{Z}) = (\boldsymbol{y}^r, \boldsymbol{Z}^r)$, and $\mu = \frac{\boldsymbol{X} \bullet \boldsymbol{Z}}{n}$.

Step 4 Choose a centrality parameter $\beta \in (0, 1)$.

Step 5 Set $\boldsymbol{H} \equiv (\beta\mu \boldsymbol{I} - \boldsymbol{X}^{\frac{1}{2}} \boldsymbol{Z} \boldsymbol{X}^{\frac{1}{2}})$.

Step 6 Compute a search direction $(\Delta \boldsymbol{X}, \Delta \boldsymbol{y}, \Delta \boldsymbol{Z})$ in $\mathcal{S}^m \times \mathbb{R}^m \times \mathcal{S}^m$ by solving

$$\begin{cases} \boldsymbol{X}^{-\frac{1}{2}}(\boldsymbol{X}\Delta\boldsymbol{Z} + \Delta\boldsymbol{X}\boldsymbol{Z})\boldsymbol{X}^{\frac{1}{2}} + \boldsymbol{X}^{\frac{1}{2}}(\Delta\boldsymbol{Z}\boldsymbol{X} + \boldsymbol{Z}\Delta\boldsymbol{X})\boldsymbol{X}^{-\frac{1}{2}} = 2\boldsymbol{H}, \\ \boldsymbol{F}_k \Delta \boldsymbol{X} = 0, \text{ for all } k = 1, \ldots, m, \\ \sum_{k=1}^m \Delta \boldsymbol{y}_k \boldsymbol{F}_k + \Delta \boldsymbol{Z} = 0. \end{cases}$$

Step 7 Choose a step size $\alpha > 0$ such that

$$\begin{cases} \hat{\boldsymbol{X}} & \equiv & \boldsymbol{X} + \alpha \Delta \boldsymbol{X} \in \mathcal{S}^n_{++}, \\ (\hat{\boldsymbol{y}}, \hat{\boldsymbol{Z}}) & \equiv & (\boldsymbol{y}, \boldsymbol{Z}) + \alpha(\Delta \boldsymbol{y}, \Delta \boldsymbol{Z}) \in \mathbb{R}^m \times \mathcal{S}^n_{++}. \end{cases}$$

Step 8 Update $\boldsymbol{X}^{r+1} = \hat{\boldsymbol{X}}$, $(\boldsymbol{y}^{r+1}, \boldsymbol{Z}^{r+1}) = (\hat{\boldsymbol{y}}, \hat{\boldsymbol{Z}})$.

Step 9 Replace r by $r + 1$. Go to **Step 1**.

For further studies on primal-dual interior-point algorithms and theoretical analysis in the viewpoint of computational complexity, we refer [23], [17], and [29].

2.1.4 Converting an optimal selection problem into SDP

To get SDP formulation of UDP (1.1), we need positive semidefiniteness of Wright's numerator relationship matrix$\boldsymbol{A}$, thus, we should review a formula to evaluate the elements of $\boldsymbol{A}$. Henderson [20] described $\boldsymbol{A}$ as a matrix that has the following properties:

1. Symmetric.
2. $A_{ii} = 1 + f_i$, where A_{ii} is the ith diagonal of $\boldsymbol{A}$, and f_i is the inbreeding coefficient of the ith individual [52].
3. $A_{ij} = r_{ij}/\sqrt{A_{ii}A_{jj}}$, where r_{ij} is Wright's coefficient of relationship between individuals i and j [52].

We denote the parents of ith pedigree (individual) by $p(i)$ and $q(i)$. The matrix numerator relationship matrix $\boldsymbol{A}$ can be evaluated by a recursive method [20]. We classify the set of pedigree candidate members $P := \{1, 2, \ldots, m\}$ into the following groups:

- If neither parent is known (P_0)

$$\begin{cases} A_{ij} = A_{ji} = 0 & \text{for } i = 1, \ldots, m,\ j = 1, \ldots, i-1, \\ A_{ii} = 1 & \text{for } i = 1, \ldots, m. \end{cases}$$

- If only one parent $p(i)$ is known (P_1)

$$\begin{cases} A_{ij} = A_{ji} = \frac{A_{j,p(i)}}{2} & \text{for } i = 1, \ldots, m,\ j = 1, \ldots, i-1, \\ A_{ii} = 1 & \text{for } i = 1, \ldots, m. \end{cases}$$

- If both parents $p(i)$ and $q(i)$ are known (P_2)

$$\begin{cases} A_{ij} = A_{ji} = \frac{A_{j,p(i)} + A_{j,q(i)}}{2} & \text{for } i = 1, \ldots, m,\ j = 1, \ldots, i-1, \\ A_{ii} = 1 + \frac{A_{p(i),q(i)}}{2} & \text{for } i = 1, \ldots, m. \end{cases}$$

To understand how to evaluate them, we give an example of pedigree heredity with $m = 7$ and a diagram in Figure 2.2. In the example, we have $P_0 = \{1, 2\}, P_1 = \{4\}$, and $P_2 = \{3, 5, 6, 7\}$. The diagram shows the relationship among individuals. For example, the 6th pedigree has parents $p(6) = 3$ and $q(6) = 4$, thus the 6th pedigree is connected with pedigrees 3 and 4. Applying the recursive method, we acquire the matrix $\boldsymbol{A}$ as in Table 2.1.

Individual (Pedigree Id)	Parents	Group
1	unknown	P_0
2	unknown	P_0
3	1 and 2	P_2
4	1 and unknown	P_1
5	1 and 4	P_2
6	3 and 4	P_2
7	5 and 6	P_2

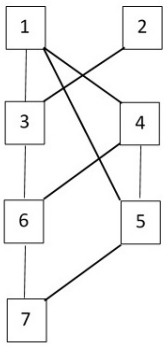

Figure 2.2: An example of pedigree heredity and its diagram.

	1	2	3	4	5	6	7
1	$\frac{32}{32}$	0	$\frac{16}{32}$	$\frac{16}{32}$	$\frac{24}{32}$	$\frac{16}{32}$	$\frac{10}{32}$
2	0	$\frac{32}{32}$	$\frac{16}{32}$	0	0	$\frac{8}{32}$	$\frac{4}{32}$
3	$\frac{16}{32}$	$\frac{16}{32}$	$\frac{32}{32}$	$\frac{8}{32}$	$\frac{12}{32}$	$\frac{20}{32}$	$\frac{16}{32}$
4	$\frac{16}{32}$	0	$\frac{8}{32}$	$\frac{32}{32}$	$\frac{24}{32}$	$\frac{20}{32}$	$\frac{22}{32}$
5	$\frac{24}{32}$	0	$\frac{12}{32}$	$\frac{24}{32}$	$\frac{40}{32}$	$\frac{18}{32}$	$\frac{29}{32}$
6	$\frac{16}{32}$	$\frac{8}{32}$	$\frac{20}{32}$	$\frac{20}{32}$	$\frac{18}{32}$	$\frac{36}{32}$	$\frac{27}{32}$
7	$\frac{10}{32}$	$\frac{4}{32}$	$\frac{16}{32}$	$\frac{22}{32}$	$\frac{29}{32}$	$\frac{27}{32}$	$\frac{41}{32}$

Table 2.1: Numerator relationship matrix.

It was Pong-Wong and Wooliams [36] who recognized for the first time that matrix $\boldsymbol{A}$ is always positive definite. They used this attribute to convert the group-coancestry constraint of (1.1) into a positive semidefinite condition using the Schur complement as explained in Fact 2.1.4, so that

$$\frac{\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}}{2} \leq \theta \quad \Leftrightarrow \quad \begin{pmatrix} 2\theta & \boldsymbol{x}^T \\ \boldsymbol{x} & \boldsymbol{A}^{-1} \end{pmatrix} \in \mathcal{S}_+^{1+m}.$$

Using this positive semidefinite constraint, they converted (1.1) into the standard SDP form (2.1). Therefore we now can post the optimal selection problem for unequal optimal selection problem as the following formulation:

$$\begin{aligned} \text{maximize} \quad &: \quad \boldsymbol{g}^T\boldsymbol{x} \\ \text{subject to} \quad &: \quad \boldsymbol{e}^T\boldsymbol{x} = 1, \\ &\quad \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}, \\ &\quad \begin{pmatrix} 2\theta & \boldsymbol{x}^T \\ \boldsymbol{x} & \boldsymbol{A}^{-1} \end{pmatrix} \in \mathcal{S}_+^{1+m}. \end{aligned}$$

2.2 SOCP Formulation

2.2.1 Second-Order Cone Programming

SOCP is a convex optimization problem, which is included in SDP as a special case, therefore, SOCP can also be solved in polynomial time by interior-point methods. Here, we consider properties for formulating SOCP problems.

Fact 2.2.1. *A matrix $\boldsymbol{V} \in \mathcal{S}^{n_r}$ of form*

$$\boldsymbol{V} = \begin{pmatrix} v_0 & \bar{\boldsymbol{v}}^T \\ \bar{\boldsymbol{v}} & v_0\boldsymbol{I} \end{pmatrix}$$

is positive semidefinite if and only if $\boldsymbol{v} = (v_0\ \bar{\boldsymbol{v}}^T)^T \in \mathcal{K}^{n_r}$. This equivalence can be verified using the Schur complement in Fact 2.1.4.

Definition 2.2.2. *A standard form of the second-order cone programming (SOCP) problem is described by*

$$\begin{aligned} \textit{maximize} \quad &: \quad \boldsymbol{c}^T\boldsymbol{z} \\ \textit{subject to} \quad &: \quad \boldsymbol{f}_0 - \boldsymbol{F}\boldsymbol{z} \in \mathbb{R}_+^{n_l} \times \mathcal{K}^{n_r}. \end{aligned} \tag{2.3}$$

where $\mathbb{R}_+^{n_l} := \{\boldsymbol{v} \in \mathbb{R}^{n_l} : v_i \geq 0\ (i = 1, ..., n_l)\}$. Here, $\boldsymbol{z} \in \mathbb{R}^m$ is the variable vector and $\boldsymbol{c} \in \mathbb{R}^m$, $\boldsymbol{f}_0 \in \mathbb{R}^{n_l+n_r+1}$, and $\boldsymbol{F} \in \mathbb{R}^{(n_l+n_r+1)\times m}$ are the input data.

Note that we can also include the Cartesian product of second-order cones as a more general form, but we use only one second-order cone in this study.

2.2.2 Formulating an optimal selection problem as SOCP

Considering positive definiteness of the numerator relationship matrix $\boldsymbol{A}$, we can apply the Cholesky factorization to matrix $\boldsymbol{A}$ to obtain an upper triangular matrix $\boldsymbol{U}$ such that $\boldsymbol{A} = \boldsymbol{U}^T\boldsymbol{U}$. This factorization derives a second-order cone condition which corresponds to the group coancestry constraint as follows:

$$\frac{\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}}{2} \leq \theta \;\Leftrightarrow\; \boldsymbol{x}^T\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{x} \leq 2\theta \;\Leftrightarrow\; ||\boldsymbol{U}\boldsymbol{x}|| \leq \sqrt{2\theta} \;\Leftrightarrow\; \left(\sqrt{2\theta}, \boldsymbol{U}\boldsymbol{x}\right) \in \mathcal{K}^m.$$

Using this factorization and the standard form of SOCP 2.3, Yamashita et al. [55] formulated the UDP (1.1) into the following problem:

$$\begin{aligned} \text{maximize} \quad &: \quad \boldsymbol{g}^T\boldsymbol{x} \\ \text{subject to} \quad &: \quad \boldsymbol{e}^T\boldsymbol{x} = 1, \\ &\quad\;\; (\sqrt{2\theta}, \boldsymbol{U}\boldsymbol{x}) \in \mathcal{K}^m, \\ &\quad\;\; \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}. \end{aligned}$$

They also utilized a sparsity of the numerical relationship matrix $\boldsymbol{A}$ in a combination with Henderson algorithm [20] for reformulating the SOCP constraint. Consequently, they proposed three SOCP formulation called *straight*, *sparse*, and *compact* SOCP formulation. Using the compact SOCP formulation, they remarkably reduced computation time of 11 hours into a few seconds.

2.3 Mixed-Integer SOCP

SOCP have been studied well for solving optimization problem efficiently. If we extend the concept of SOCP incorporating integer variables, we can have a new class of optimization problems called mixed-integer SOCP (MI-SOCP). MI-SOCP appears in different contexts such as portfolio selection problems in financial engineering or network design and operations; see [7].

Definition 2.3.1. *Mixed-integer second order cone programming is of form:*

$$\begin{aligned} \textit{maximize} \quad &: \quad \boldsymbol{c}^T\boldsymbol{z} \\ \textit{subject to} \quad &: \quad \boldsymbol{f}_0 - \boldsymbol{F}\boldsymbol{z} \in \mathbb{R}^{n_l}_+ \times \mathcal{K}^{n_r}, \\ &\quad\;\; z_i \in \mathbb{Z} \textit{ for } i \in V \subset \{1, ..., m\}. \end{aligned} \tag{2.4}$$

In order to solve MI-SOCP, many approaches have been proposed, particularly on the branch-and-bound method. This method was originally proposed by Land and Doig [24], and implemented by Little et al. [25] for solving the integer program arising from traveling salesman problem. The method basically solves a problem by dividing the feasible region into subregions (branching), evaluating bounds on the objective value of each corresponding subproblem, and using them to eliminate some unsuccesfull subproblems (bounding). The bounds can be computed by replacing the current subproblem with an easier relaxation problem. Hence the new solution of the latter yields

an upper bound for the former. The procedure terminates when it encountered one of two conditions. A first condition is that if each subproblem produces an infeasible solution. The other condition is when the subproblem contains no better solution than a previous one. For further studies of the branch-and-bound method, we refer to [43].

There are other approaches for solving MI-SOCP problems besides the branch-and-bound method, for example, an outer-approximation [15], generalized Benders decomposition [18]. Here, we will present an outer-approximation algorithm for solving MI-SOCP.

2.3.1 An Outer-Approximation Algorithm

An outer-approximation approach was originally introduced by Duran and Grossman [15] for solving the following MINLP problem:

$$\begin{aligned} z = \min\ & \boldsymbol{c}^T\boldsymbol{y} + f(\boldsymbol{x}) \\ \text{s.t.}\quad & g(\boldsymbol{x}) + \boldsymbol{B}\boldsymbol{y} \leq 0, \\ & \boldsymbol{x} \in X \subset \mathbb{R}^n, \\ & \boldsymbol{y} \in \{0,1\}^m \end{aligned} \tag{2.5}$$

where $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ are convex function. Here $X = \{\boldsymbol{x} :\ \boldsymbol{x} \in \mathbb{R}^n,\ \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{a}\}$, with $\boldsymbol{A}$ and $\boldsymbol{a}$ being a matrix and a vector, respectively. They solved the MINLP problem by imposing linear constraints successively instead of the integer constraints.

We now present an outer-approximation algorithm (Algorithm 2) based on [15]. For a given $\boldsymbol{x}^j \in \mathbb{R}^n$, define

$$\begin{aligned} C(\boldsymbol{x}^j) \quad = \quad & \{(\boldsymbol{x}, \boldsymbol{y}, \mu) : f(\boldsymbol{x}^j) + \nabla f(\boldsymbol{x}^j)^T(\boldsymbol{x} - \boldsymbol{x}^j) - \mu \leq 0, \\ & g(\boldsymbol{x}^j) + \nabla g(\boldsymbol{x}^j)^T(\boldsymbol{x} - \boldsymbol{x}^j) + \boldsymbol{B}\boldsymbol{y} \leq 0\}, \end{aligned}$$

where $\mu \in [f_L, f_U]$ is an auxiliary scalar variable. Here, $f_L = \min\{f(x) : \boldsymbol{x} \in \boldsymbol{X}\}$ and $f_U = \max\{f(x) : x \in X\}$ are valid finite bounds of f on $\boldsymbol{X}$ [15].

Algorithm 2.3.2. [An outer-approximation [15]]

Step 1 Set $\gamma^0 = \mathbb{R}^n \times \mathbb{R}^m,\ z^0 = -\infty,\ z^u = +\infty,\ j = 1,\ \boldsymbol{y}^1 \in \{0,1\}^m$. Here z^0 is a lower bound and z^u is an upper bound.

Step 2 Solve an NLP subproblem $S(\boldsymbol{y}^j)$ with respect to $\boldsymbol{x}$ by fixing $\boldsymbol{y}$ at $\boldsymbol{y}^j$:

$$\begin{aligned} z(\boldsymbol{y}^j) \quad = \quad & \boldsymbol{c}^T\boldsymbol{y}^j + \min\ f(\boldsymbol{x}) \\ \text{s.t.} \quad & g(\boldsymbol{x}) + \boldsymbol{B}\boldsymbol{y}^j \leq \boldsymbol{0}, \qquad\qquad (S(\boldsymbol{y}^j)) \\ & \boldsymbol{x} \in X. \end{aligned}$$

Step 2-1 If $S(\boldsymbol{y}^j)$ has a finite optimal solution $\{\boldsymbol{x}^j, z(\boldsymbol{y}^j)\}$, update $z^u = \min\{z^u, z(\boldsymbol{y}^j)\}$. Then, set $\gamma^j = \gamma^{j-1} \cap C(\boldsymbol{x}^j)$. Go to **Step 3**.

Step 2-2 If $S(\boldsymbol{y}^j)$ outputs infeasible solution $\boldsymbol{y}^j \notin \{0,1\}^m$ for (2.5) with $\boldsymbol{x}^j$, eliminate $\boldsymbol{y}^j$ by adding an *integer cut* to the master problem M^j below. Then, set $\gamma^j = \gamma^{j-1} \cap C(\boldsymbol{x}^j)$.

Step 3 Solve the relaxed MILP program M^j:

$$\begin{array}{rcll} z^j & = & \min & \boldsymbol{c}^T\boldsymbol{y} + \mu \\ & & \text{s.t} & (\boldsymbol{x}, \boldsymbol{y}, \mu) \in \gamma^j, \\ & & & \boldsymbol{y} \in \text{(set of integer cuts)}, \\ & & & z^{j-1} \leq \boldsymbol{c}^T\boldsymbol{y} + \mu < z^u, \\ & & & \boldsymbol{x} \in X,\ \boldsymbol{y} \in \{0,1\}^m,\ \mu \in \mathbb{R}^1 \end{array} \qquad (M^j)$$

Step 3-1 If M^j is infeasible, **STOP**. The optimal solution of problem (2.5) is the current z^u, $\boldsymbol{x}^j$, and $\boldsymbol{y}^j$.

Step 3-2 If M^j has a finite optimal solution, set $\boldsymbol{y}^{j+1} = \boldsymbol{y}^j$ and $j = j + 1$. Return to Step 2.

The integer cut in **Step 2-2** is derived using the following lemma.

Lemma 2.3.3. [15] *Given any integer combination $\boldsymbol{y}^j = (y_1^j, y_2^j, \ldots, y_m^j)^T \in \{0,1\}^m$ with index sets $B^j = \{j : y_i^j = 1\},\ NB^j = \{j : y_i^j = 0\}$ s.t $\left|B^i\right| + \left|NB^i\right| = m$, the integer constraint*

$$\sum_{i \in B^j} y_i - \sum_{i \in NB^j} y_i \leq \left|B^i\right| - 1$$

will be violated only by $\boldsymbol{y}^j$ and no other $\boldsymbol{y} \in \{0,1\}^m$ at the kth iteration.

Mullin et al. [31] applied this outer-approximation approach combined with the branch-and-bound method to solve EDP. The MINLP formulation is:

$$\begin{array}{rcl} \text{maximize} & : & \frac{\boldsymbol{g}^T\boldsymbol{y}}{N} \\ \text{subject to} & : & \boldsymbol{e}^T\boldsymbol{y} = N, \\ & & \boldsymbol{y}^T\boldsymbol{A}\boldsymbol{y} \leq 2\theta N^2, \\ & & y_i \in \{0,1\},\ i = 1, \ldots, m. \end{array}$$

They approximated the group coancestry constraint $\boldsymbol{y}^T\boldsymbol{A}\boldsymbol{y} \leq 2\theta N^2$ with linear constraints

$$\boldsymbol{y}^{iT}\boldsymbol{A}\boldsymbol{y} \leq \sqrt{2\theta \boldsymbol{y}^{iT}\boldsymbol{A}\boldsymbol{y}^i} \quad i = 1, 2, \ldots$$

for $\{\boldsymbol{y}^i\} \subset \{\boldsymbol{y} : \boldsymbol{y}^T\boldsymbol{A}\boldsymbol{y} > 2\theta\}$. The derivation of such constraint can be found in [31]. However, the computation cost based on such an approximation was very demanding. Therefore, there is a high demand for another approach for solving the MI-SOCP problem (1.3).

3 Conic Relaxation

In this chapter, we first derive an SDP relaxation problem of an EDP. Then, by a further relaxation using the dual first-level relaxation of the scaled diagonally dominant sum-of-squares (SDSOS) relaxations hierarchy [2], we obtain an LP relaxation problem. Finally, we apply a continuous relaxation technique to the EDP to obtain an SOCP relaxation problem. The reason we employ a different relaxation approach for only the SOCP relaxation is that we can exploit a structural sparsity in Wright's numerator matrix $\boldsymbol{A}$.

3.1 Conic Relaxation

As a first step to derive an SDP relaxation from the EDP (1.2), we remove the variables that can be fixed from the box constraints. More precisely, if $l_i > 0$, we fix $x_i = \frac{1}{N}$. Similarly, we fix $x_i = 0$ if $u_i < \frac{1}{N}$. We then define two sets F and V so that the two sets separate the set $\{1, \ldots, m\}$ disjointly and x_i is fixed to $c_i \in \left\{0, \frac{1}{N}\right\}$ for $i \in F$ while x_i remains as a decision variable for $i \in V$.

Without loss of generality, we assume that $V = \{1, 2, \ldots, |V|\}$, $F = \{|V|+1, |V|+2, \ldots, m\}$, and $g_1 \geq g_2 \geq \ldots \geq g_{|V|}$. Along with these V and F, we introduce the vectors $\boldsymbol{x}_V$ and $\boldsymbol{c}_F$ that divide $\boldsymbol{x} \in \mathbb{R}^m$ into the two parts $\boldsymbol{x} = \begin{pmatrix} \boldsymbol{x}_V \\ \boldsymbol{c}_F \end{pmatrix}$. We also divide the Wright numerator matrix $\boldsymbol{A}$ into the four parts; $\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}_{VV} & \boldsymbol{A}_{VF} \\ \boldsymbol{A}_{FV} & \boldsymbol{A}_{FF} \end{pmatrix}$. The sizes of $\boldsymbol{A}_{VV}$, $\boldsymbol{A}_{FV} (= \boldsymbol{A}_{VF}^T)$, and $\boldsymbol{A}_{FF}$ are $|V| \times |V|$, $|F| \times |V|$, and $|F| \times |F|$, respectively. We further partition the vectors and the matrices that appear in the EDP into the corresponding parts;

$$\begin{array}{rcll} OPT_{ED} & = & \max & : \boldsymbol{g}_V^T \boldsymbol{x}_V + \boldsymbol{g}_F^T \boldsymbol{c}_F \\ & & \text{subject to} & : \boldsymbol{x}_V^T \boldsymbol{A}_{VV} \boldsymbol{x}_V + 2\boldsymbol{c}_F^T \boldsymbol{A}_{FV} \boldsymbol{x}_V + \boldsymbol{c}_F^T \boldsymbol{A}_{FF} \boldsymbol{c}_F \leq 2\theta, \\ & & & \boldsymbol{e}_V^T \boldsymbol{x}_V + \boldsymbol{e}_F^T \boldsymbol{c}_F = 1, \\ & & & x_i \in \left\{0, \frac{1}{N}\right\} \text{ for } i \in V. \end{array} \tag{3.1}$$

Note that we also removed the box constraints $\boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}$ from the EDP by fixing the variables in $\boldsymbol{x}_F$ to $\boldsymbol{c}_F$. We count the number of x_i that is fixed to c_i by $p := \left|\left\{i \in F : x_i = \frac{1}{N}\right\}\right|$.

Remark 3.1.1. *We can assume $p \leq N$ and $|V| \geq 2$ without loss of generality. In the case $p > N$, we can detect the infeasibility of the problem (1.2). If $|V| = 1$, we have $F = \{2, \ldots, m\}$. Therefore, x_1 is also fixed with $x_1 = 1 - \sum_{i=2}^m c_i$, and all the variables can be fixed without solving (3.1).*

We change the decision variables by $\boldsymbol{y}_V := 2N\boldsymbol{x}_V - \boldsymbol{e}_V \in \mathbb{R}^{|V|}$ and we use y_i to denote the ith element of $\boldsymbol{y}_V$, so that the binary constraints $x_1, \ldots, x_{|V|} \in \{0, \frac{1}{N}\}$ are mapped to $y_1, \ldots, y_{|V|} \in \{-1, 1\}$. We define $g_{\min} := \min\{g_i : i = 1, \ldots, m\}$, $\bar{\boldsymbol{g}}_V := \frac{1}{4N}(\boldsymbol{g}_V - g_{\min}\boldsymbol{e}_V)$, $\bar{g} := \frac{1}{2N}(\boldsymbol{g}_V - g_{\min}\boldsymbol{e}_V)^T\boldsymbol{e}_V + (\boldsymbol{g}_F - g_{\min}\boldsymbol{e}_F)^T\boldsymbol{c}_F + g_{\min}$, $\bar{\boldsymbol{c}}_F := \boldsymbol{A}_{VV}\boldsymbol{e}_V + 2N\boldsymbol{A}_{VF}\boldsymbol{c}_F$, $\bar{\theta} := 2N^2(2\theta - \boldsymbol{c}_F^T\boldsymbol{A}_{FF}\boldsymbol{c}_F) - \frac{1}{2}\boldsymbol{e}_V^T\boldsymbol{A}_{VV}\boldsymbol{e}_V - 2N\boldsymbol{c}_F^T\boldsymbol{A}_{FV}\boldsymbol{e}_V$, and $\bar{N} := 2N(1 - \boldsymbol{e}_F^T\boldsymbol{c}_F) - |V| = 2(N-p) - |V|$. From these definitions, it is easy to check $\bar{\boldsymbol{g}}_V \geq \boldsymbol{0}$ and $\boldsymbol{g}^T\boldsymbol{x} = 2\bar{\boldsymbol{g}}_V^T\boldsymbol{y}_V + \bar{g}$ using $\boldsymbol{e}^T\boldsymbol{x} = 1$. We now have another expression of the EDP;

$$\begin{array}{rcll} OPT_{ED} & = & \max & : \ 2\bar{\boldsymbol{g}}_V^T\boldsymbol{y}_V + \bar{g} \\ & & \text{subject to} & : \ \boldsymbol{y}_V^T\boldsymbol{A}_{VV}\boldsymbol{y}_V + 2\bar{\boldsymbol{c}}_F^T\boldsymbol{y}_V \leq 2\bar{\theta}, \\ & & & \ \boldsymbol{e}_V^T\boldsymbol{y}_V = \bar{N}, \\ & & & \ y_i \in \{-1, 1\} \text{ for } i \in V. \end{array} \tag{3.2}$$

We introduce a variable matrix $\boldsymbol{Y}_{VV} \in \mathbb{S}^{|V|}$ and we will denote the (i,j)th element of $\boldsymbol{Y}_{VV}$ by Y_{ij}. The condition $\boldsymbol{Y}_{VV} = \boldsymbol{y}_V\boldsymbol{y}_V^T$ holds if and only if $\begin{pmatrix} 1 & \boldsymbol{y}_V^T \\ \boldsymbol{y}_V & \boldsymbol{Y}_{VV} \end{pmatrix} \succeq \boldsymbol{O}$, $\mathrm{rank}\left(\begin{pmatrix} 1 & \boldsymbol{y}_V^T \\ \boldsymbol{y}_V & \boldsymbol{Y}_{VV} \end{pmatrix}\right) = 1$. By ignoring the rank constraint that embraces the nature of combinatorial optimization, we build an SDP relaxation problem and we denote its optimal value by OPT_{SDP};

$$\begin{array}{rcll} OPT_{SDP} & := & \max & : \ \begin{pmatrix} 0 & \bar{\boldsymbol{g}}_V^T \\ \bar{\boldsymbol{g}}_V & \boldsymbol{O} \end{pmatrix} \bullet \begin{pmatrix} 1 & \boldsymbol{y}_V^T \\ \boldsymbol{y}_V & \boldsymbol{Y}_{VV} \end{pmatrix} + \bar{g} \\ & & \text{subject to} & : \ (\boldsymbol{y}_V, \boldsymbol{Y}_{VV}) \in \mathcal{Y} \\ & & & \ \begin{pmatrix} 1 & \boldsymbol{y}_V^T \\ \boldsymbol{y}_V & \boldsymbol{Y}_{VV} \end{pmatrix} \succeq \boldsymbol{O}. \end{array} \tag{3.3}$$

Here, $\mathcal{Y}$ is a set that is composed of linear constraints

$$\begin{array}{rcl} \mathcal{Y} & := & \left\{(\boldsymbol{y}_V, \boldsymbol{Y}_{VV}) \in \mathbb{R}^n \times \mathbb{S}^n : \begin{pmatrix} -2\bar{\theta} & \bar{\boldsymbol{c}}_F^T \\ \bar{\boldsymbol{c}}_F & \boldsymbol{A}_{VV} \end{pmatrix} \bullet \begin{pmatrix} 1 & \boldsymbol{y}_V^T \\ \boldsymbol{y}_V & \boldsymbol{Y}_{VV} \end{pmatrix} \leq 0, \right. \\ & & \begin{pmatrix} -2\bar{N} & \boldsymbol{e}_V^T \\ \boldsymbol{e}_V & \boldsymbol{O} \end{pmatrix} \bullet \begin{pmatrix} 1 & \boldsymbol{y}_V^T \\ \boldsymbol{y}_V & \boldsymbol{Y}_{VV} \end{pmatrix} = 0, \begin{pmatrix} -\bar{N}^2 & \boldsymbol{0}^T \\ \boldsymbol{0} & \boldsymbol{e}_V\boldsymbol{e}_V^T \end{pmatrix} \bullet \begin{pmatrix} 1 & \boldsymbol{y}_V^T \\ \boldsymbol{y}_V & \boldsymbol{Y}_{VV} \end{pmatrix} = 0, \\ & & \left. \begin{pmatrix} -1 & \boldsymbol{0}^T \\ \boldsymbol{0} & \boldsymbol{e}_i\boldsymbol{e}_i^T \end{pmatrix} \bullet \begin{pmatrix} 1 & \boldsymbol{y}_V^T \\ \boldsymbol{y}_V & \boldsymbol{Y}_{VV} \end{pmatrix} = 0 \text{ for } i \in V \right\}. \end{array} \tag{3.4}$$

In this definition, we introduced a redundant constraint $(\boldsymbol{e}_V\boldsymbol{e}_V^T) \bullet \boldsymbol{Y}_{VV} = \bar{N}^2$ based on the relation $(\boldsymbol{e}_V^T\boldsymbol{y}_V)^2 = \bar{N}^2$ that holds when $\boldsymbol{Y}_{VV} = \boldsymbol{y}_V\boldsymbol{y}_V^T$. It is known that redundant constraints of this type make the SDP relaxation tighter, and we can often obtain a better approximate solution.

However, such a condition requires a heavy computation cost, thus we consider a further relaxation based on the dual first level of SDSOS relaxations [2]. We now define $\hat{W}$ to denote the non-diagonal upper-triangular position of $\boldsymbol{Y}_{VV}$, that is $\hat{W} := \{(i,j) \in V \times V : i < j\}$. The application of the SDSOS relaxation in this context replaces the positive semidefinite condition

Table 3.1: The numbers of nonzero elements in the Wright numerator relationship matrix $\boldsymbol{A}$ and its inverse $\boldsymbol{A}^{-1}$

m	$\boldsymbol{A}$	$\boldsymbol{A}^{-1}$
200	2000	900
1050	634762	5550
2045	815099	10027
5050	16586610	25550
10100	56754980	51100
15222	15102210	75612

in (3.3) with the positive semidefinite conditions of principal submatrices of dimension 2.

$$\begin{pmatrix} 1 & y_i \\ y_i & Y_{ii} \end{pmatrix} \succeq \boldsymbol{O} \ (i \in V), \quad \begin{pmatrix} Y_{ii} & Y_{ij} \\ Y_{ij} & Y_{jj} \end{pmatrix} \succeq \boldsymbol{O} \ ((i,j) \in \hat{W}). \tag{3.5}$$

From the constraint $\begin{pmatrix} -1 & \boldsymbol{0}^T \\ \boldsymbol{0} & \boldsymbol{e}_i\boldsymbol{e}_i^T \end{pmatrix} \bullet \begin{pmatrix} 1 & \boldsymbol{y}_V^T \\ \boldsymbol{y}_V & \boldsymbol{Y}_{VV} \end{pmatrix} = 0$ in (3.4), we know $Y_{ii} = 1$ for $i \in V$. Therefore, (3.5) can be described by linear constraints. Consequently, we reach an LP relaxation problem, whose optimal value is denoted as OPT_{LP}.

$$\begin{array}{rcll} OPT_{LP} & = & \max & : \begin{pmatrix} 0 & \bar{\boldsymbol{g}}_V^T \\ \bar{\boldsymbol{g}}_V & \boldsymbol{O} \end{pmatrix} \bullet \begin{pmatrix} 1 & \boldsymbol{y}_V^T \\ \boldsymbol{y}_V & \boldsymbol{Y}_{VV} \end{pmatrix} + \bar{g} \\ & & \text{subject to} & : (\boldsymbol{y}_V, \boldsymbol{Y}_{VV}) \in \mathcal{Y}, \\ & & & \quad -1 \le y_i \le 1 \quad \text{for} \quad i \in V, \\ & & & \quad -1 \le Y_{ij} \le 1 \quad \text{for} \quad (i,j) \in \hat{W}. \end{array} \tag{3.6}$$

In a fashion similar to the above step that derives (3.6) from (3.3), it may be possible to apply an SOCP relaxation technique developed in [21] to (3.3). In contrast, we utilize a continuous relaxation technique that converts the binary constraint $x_i \in \{0, \frac{1}{N}\}$ into a continuous constraint $0 \le x_i \le \frac{1}{N}$. The main reason of this continuous relaxation is that we can extensively exploit a structural sparsity hidden in the inverse matrix of the Wright numerator matrix $\boldsymbol{A}$. Table 3.1 reports the numbers of nonzero elements in $\boldsymbol{A}$ and its inverse $\boldsymbol{A}^{-1}$ for the data set we used in Section 3.2. From this table, it is clear that $\boldsymbol{A}^{-1}$ is much sparser than $\boldsymbol{A}$. Simply speaking, the elements in $\boldsymbol{A}$ reflect the information of all ancestors, but those in $\boldsymbol{A}^{-1}$ are computed by the information of direct parents. The number of direct parents for each genotype is only two, therefore $\boldsymbol{A}^{-1}$ has a remarkable sparsity. More details can be found in [20]. To utilize this sparsity in an efficient manner, Yamashita et al. [55] introduced a new vector $\boldsymbol{z} := \boldsymbol{A}\boldsymbol{x} \in \mathbb{R}^m$, and converted the quadratic constraint $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} \le 2\theta$ into $||\boldsymbol{B}\boldsymbol{z}|| \le \sqrt{2\theta}$ with a matrix $\boldsymbol{B} \in \mathbb{R}^{m \times m}$ that satisfies $\boldsymbol{B}^T\boldsymbol{B} = \boldsymbol{A}^{-1}$. Due a similar nature to $\boldsymbol{A}^{-1}$, the matrix $\boldsymbol{B}$ possesses favorable sparsity. The computation time reduction reported in [55] was mainly derived from this sparsity. Using this new vector $\boldsymbol{z}$ and matrix $\boldsymbol{B}$, we transformed the EDP (1.2) into the following SOCP problem with integer constraints;

$$\begin{array}{lcll} \max & : & (\boldsymbol{A}^{-1}\boldsymbol{g})^T\boldsymbol{z} & \\ \text{subject to} & : & (\boldsymbol{A}^{-1}\boldsymbol{e})^T\boldsymbol{z} = 1, & \\ & & \begin{pmatrix} \sqrt{2\theta} \\ \boldsymbol{B}\boldsymbol{z} \end{pmatrix} \in \mathcal{K}^m, & \\ & & [\boldsymbol{A}^{-1}\boldsymbol{z}]_i \in \{0, \frac{1}{N}\} & \text{for} \quad i \in V, \\ & & [\boldsymbol{A}^{-1}\boldsymbol{z}]_i = c_i & \text{for} \quad i \in F. \end{array}$$

Here, we use the notation $[\boldsymbol{A}^{-1}\boldsymbol{z}]_i$ to denote the ith element of $\boldsymbol{A}^{-1}\boldsymbol{z}$. It may seem that we would remove $\boldsymbol{x}_F$ from this formulation by fixing $\boldsymbol{x}_F = \boldsymbol{c}_F$ and reduce the size the problem; however, such elimination would strongly diminish the efficiency of the SOCP problem, since it completely destroys the favorable sparsity that appear in $\boldsymbol{A}^{-1}$ and $\boldsymbol{B}$.

By applying the continuous relaxation to the binary constraints, we obtain an SOCP relaxation problem of the EDP (1.2):

$$\begin{array}{rcllll} OPT_{SOCP} & := & \max & : & (\boldsymbol{A}^{-1}\boldsymbol{g})^T\boldsymbol{z} & \\ & & \text{subject to} & : & (\boldsymbol{A}^{-1}\boldsymbol{e})^T\boldsymbol{z} = 1, & \\ & & & & \begin{pmatrix} \sqrt{2\theta} \\ \boldsymbol{B}\boldsymbol{z} \end{pmatrix} \in \mathcal{K}^m, & \\ & & & & 0 \leq [\boldsymbol{A}^{-1}\boldsymbol{z}]_i \leq \frac{1}{N} & \text{for} \quad i \in V, \\ & & & & [\boldsymbol{A}^{-1}\boldsymbol{z}]_i = c_i & \text{for} \quad i \in F. \end{array} \tag{3.7}$$

Note that most constraints in this problem are linear constraints, but one constraint involves a second-order cone, therefore, this problem is an SOCP problem.

From the derivation of the LP relaxation problem (3.6), it is natural that the SDP relaxation problem (3.3) gives a closer-to-optimal value than the LP relaxation, that is, we know $OPT_{SDP} \leq OPT_{LP}$. In contrast, the relation of the SOCP relaxation (3.7) is not so explicit, since the SOCP relaxation was derived by a continuous relaxation, independent from the SDP or LP relaxation.

The strength of these relaxation problems can be summarized in Lemma 3.1.3. For the discussion there, we prepare some notation and introduce an assumption. We use $\mathcal{S}_k(\boldsymbol{v})$ to denote the sum of the k smallest elements of $\boldsymbol{v} \in \mathbb{R}^n$, that is, $\mathcal{S}_k(\boldsymbol{v}) := \sum_{i=1}^k \hat{v}_i$ where $\hat{v}_1 \leq \hat{v}_2 \leq \ldots \leq \hat{v}_n$ is the sorted vector of $\boldsymbol{v}$ in the ascending order. The symbol $\hat{A}_{\hat{W}}$ indicates the set of the collection of $\boldsymbol{A}_{VV}$ with respect to $\hat{W}$, that is, $\hat{A}_{\hat{W}} := \left\{ A_{ij} : (i,j) \in \hat{W} \right\}$. We define a vector $\hat{\boldsymbol{y}}_V \in \mathbb{R}^{|V|}$ by $[\hat{y}_V]_i := 1$ for $i = 1, \ldots, N-p$ and $[\hat{y}_V]_i := -1$ for $i = N-p+1, \ldots, |V|$. This vector satisfies $\boldsymbol{e}_V^T\hat{\boldsymbol{y}}_V = \bar{N}$. In the following discussion, we make the following assumption on the input data of the EDP (1.2). From preliminary tests, we numerically verified that this assumption holds for practical datasets for estimated genetic values for pedigreed pine populations and datasets generated by simulations. The details of these datasets will be described in Section 3.2.

Assumption 3.1.2. *The input data of (1.2) satisfies*

$$\mathcal{S}_{\hat{N}}(\hat{A}_{\hat{W}}) \leq \frac{2\bar{\theta} - 2\mathit{Trace}(\boldsymbol{A}_{VV}) + \boldsymbol{e}_V^T\boldsymbol{A}_{VV}\boldsymbol{e}_V - 2\bar{\boldsymbol{c}}_F^T\hat{\boldsymbol{y}}_V}{4},$$

where $\hat{N} := \frac{\bar{N}^2 + |V|^2 - 2|V|}{4}$.

We should ensure that $\hat{N}$ is a positive integer, otherwise we need to manage a fractional number in the definition of $\mathcal{S}$. The positiveness is derived from $\bar{N}^2 + |V|^2 - 2|V| \geq \bar{N}^2 + 1 \geq 1$ by $|V| \geq 2$ of Remark 3.1.1, and $\hat{N}$ is integer by

$$\begin{aligned} & \bar{N}^2 + |V|^2 - 2|V| = \left\{2N(1 - \boldsymbol{e}_F^T\boldsymbol{c}_F) - |V|\right\}^2 + |V|^2 - 2|V| \\ = \quad & \left\{2N(1 - \frac{p}{N}) - |V|\right\}^2 + |V|^2 - 2|V| = 4\left\{(N-p)^2 - |V|(N-p) + \frac{|V|(|V|-1)}{2}\right\}. \end{aligned}$$

We are now prepared to examine the relation between the relaxation problems.

Lemma 3.1.3. *It holds for the optimal values of the relaxation problems that*

$$OPT_{ED} \leq OPT_{SDP} \leq OPT_{SOCP}.$$

Furthermore, if Assumption 3.1.2 holds, then

$$OPT_{ED} \leq OPT_{SDP} \leq OPT_{SOCP} \leq OPT_{LP}.$$

Proof:

[$OPT_{ED} \leq OPT_{SDP}$] When we derived (3.3), we ignored the rank-1 constraint; hence, the feasible region of (3.3) is substantially wider than that of (3.2), so we have $OPT_{ED} \leq OPT_{SDP}$.

[$OPT_{SDP} \leq OPT_{SOCP}$] We take any feasible solution $\boldsymbol{y}_V \in \mathbb{R}^{|V|}$ and $\boldsymbol{Y}_{VV} \in \mathbb{S}^{|V|}$ of (3.3). It is enough to check that $\boldsymbol{z} = \boldsymbol{A} \begin{pmatrix} \frac{\boldsymbol{y}_V + \boldsymbol{e}_V}{2N} \\ \boldsymbol{c}_F \end{pmatrix}$ is a feasible solution of (3.7). From $\begin{pmatrix} -2\bar{N} & \boldsymbol{e}_V^T \\ \boldsymbol{e}_V & \boldsymbol{O} \end{pmatrix} \bullet \begin{pmatrix} 1 & \boldsymbol{y}_V^T \\ \boldsymbol{y}_V & \boldsymbol{Y}_{VV} \end{pmatrix} = 0$, we obtain $\boldsymbol{e}_V^T \boldsymbol{y}_V = \bar{N} = 2N(1 - \boldsymbol{e}_F^T \boldsymbol{c}_F) - |V|$, hence, $(\boldsymbol{A}^{-1}\boldsymbol{e})^T \boldsymbol{z} = \begin{pmatrix} \boldsymbol{e}_V \\ \boldsymbol{e}_F \end{pmatrix}^T \begin{pmatrix} \frac{\boldsymbol{y}_V + \boldsymbol{e}_V}{2N} \\ \boldsymbol{c}_F \end{pmatrix} = \frac{\boldsymbol{e}_V^T \boldsymbol{y}_V + |V|}{2N} + \boldsymbol{e}_F^T \boldsymbol{c}_F = 1$. By applying the Schur complement to the positive semi-definite condition $\begin{pmatrix} 1 & \boldsymbol{y}_V^T \\ \boldsymbol{y}_V & \boldsymbol{Y}_{VV} \end{pmatrix} \succeq \boldsymbol{O}$, it holds $\boldsymbol{Y}_{VV} - \boldsymbol{y}_V \boldsymbol{y}_V^T \succeq \boldsymbol{O}$. Since $\boldsymbol{A} \bullet \boldsymbol{X} \geq 0$ holds for any two positive semidefinite matrices of the same dimension $\boldsymbol{A}$ and $\boldsymbol{X}$ (as shown in [44]) and the Wright numerator matrix is always positive definite, it holds $\boldsymbol{A}_{VV} \bullet (\boldsymbol{Y}_{VV} - \boldsymbol{y}_V \boldsymbol{y}_V^T) \geq 0$, therefore, $\boldsymbol{A}_{VV} \bullet \boldsymbol{Y}_{VV} \geq \boldsymbol{y}_V^T \boldsymbol{A}_{VV} \boldsymbol{y}_V$. Using the relationship $\begin{pmatrix} -2\bar{\theta} & \bar{\boldsymbol{c}}_F^T \\ \bar{\boldsymbol{c}}_F & \boldsymbol{A}_{VV} \end{pmatrix} \bullet \begin{pmatrix} 1 & \boldsymbol{y}_V^T \\ \boldsymbol{y}_V & \boldsymbol{Y}_{VV} \end{pmatrix} \leq 0$, we obtain $\boldsymbol{y}_V^T \boldsymbol{A}_{VV} \boldsymbol{y}_V + 2\bar{\boldsymbol{c}}_F^T \boldsymbol{y}_V \leq 2\bar{\theta}$. From the definitions of $\boldsymbol{y}_V, \bar{\boldsymbol{c}}_F, \bar{\theta}, \boldsymbol{B}$ and $\boldsymbol{z}$, we can derive $\boldsymbol{z}^T \boldsymbol{B}^T \boldsymbol{B} \boldsymbol{z} \leq 2\theta$, therefore, $\begin{pmatrix} \sqrt{2\theta} \\ \boldsymbol{B}\boldsymbol{z} \end{pmatrix} \in \mathcal{K}^m$. From $\boldsymbol{Y}_{VV} - \boldsymbol{y}_V \boldsymbol{y}_V^T \succeq \boldsymbol{O}$, we also have $Y_{ii} \geq y_i^2$ for $i = 1, \ldots, |V|$. Furthermore, due to the constraint $\begin{pmatrix} -1 & \boldsymbol{0}^T \\ \boldsymbol{0} & \boldsymbol{e}_i \boldsymbol{e}_i^T \end{pmatrix} \bullet \begin{pmatrix} 1 & \boldsymbol{y}_V^T \\ \boldsymbol{y}_V & \boldsymbol{Y}_{VV} \end{pmatrix} = 0$, it holds $Y_{ii} = 1$ for $i = 1, \ldots, |V|$, and consequently $-\boldsymbol{e}_V \leq \boldsymbol{y}_V \leq \boldsymbol{e}_V$. From $\boldsymbol{A}^{-1}\boldsymbol{z} = \begin{pmatrix} \frac{\boldsymbol{y}_V + \boldsymbol{e}_V}{2N} \\ \boldsymbol{c}_F \end{pmatrix}$, it is now clear that $0 \leq [\boldsymbol{A}^{-1}\boldsymbol{z}]_i \leq \frac{1}{N}$ for $i \in V$ and that $[\boldsymbol{A}^{-1}\boldsymbol{z}]_i = c_i$ for $i \in F$. Furthermore, the objective value of (3.3) at $\boldsymbol{y}_V$ is same as that of (3.7) at $\boldsymbol{z}$ if $\boldsymbol{z} = \boldsymbol{A} \begin{pmatrix} \frac{\boldsymbol{y}_V + \boldsymbol{e}_V}{2N} \\ \boldsymbol{c}_F \end{pmatrix}$; hence, we obtain $OPT_{SDP} \leq OPT_{SOCP}$.

[$OPT_{SOCP} \leq OPT_{LP}$] If we ignore the quadratic constraint of (3.7) and we reverse the variable into $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{z}$, we obtain an optimization problem of form

$$\begin{array}{lll} \max & : & \boldsymbol{g}_V^T \boldsymbol{x}_V + \boldsymbol{g}_F^T \boldsymbol{c}_F, \\ \text{subject to} & : & \boldsymbol{e}_V^T \boldsymbol{x}_V = 1 - \frac{p}{N}, \\ & & 0 \leq x_i \leq \frac{1}{N} \quad \text{for} \quad i \in V. \end{array} \tag{3.8}$$

Since $g_1 \geq g_2 \geq \ldots \geq g_{|V|}$, the optimal value of (3.8) is given by $-\frac{\mathcal{S}_{N-p}(-\boldsymbol{g}_V)}{N} + \boldsymbol{g}_F^T \boldsymbol{c}_F$ and an optimal solution is $\hat{\boldsymbol{x}}_V := \frac{\hat{\boldsymbol{y}}_V + \boldsymbol{e}_V}{2N}$. Therefore, it holds that $OPT_{SOCP} \leq \boldsymbol{g}_V^T \hat{\boldsymbol{x}}_V + \boldsymbol{g}_F^T \boldsymbol{c}_F$.

Next, we define ρ_{LP} to denote the optimal value of the following LP problem;

$$\begin{array}{lllll} \rho_{LP} & := & \min & : & \boldsymbol{A}_{VV} \bullet \boldsymbol{Y}_{VV} \\ & & \text{subject to} & : & (\boldsymbol{e}_V \boldsymbol{e}_V^T) \bullet \boldsymbol{Y}_{VV} = \bar{N}^2, \\ & & & & Y_{ii} = 1 \text{ for } i \in V, \\ & & & & -1 \leq Y_{ij} \leq 1 \text{ for } (i,j) \in \hat{W}, \\ & & & & \boldsymbol{Y}_{VV} \in \mathbb{S}^{|V|}. \end{array} \tag{3.9}$$

We convert this problem introducing $\bar{X}_{ij} := \frac{Y_{ij}+1}{2}$ for $(i,j) \in \hat{W}$. The following LP problem is equivalent to (3.9), so its optimal value must be ρ_{LP}.

$$\begin{array}{lllll} \rho_{LP} & = & \min & : & 4\sum_{(i,j)\in\hat{W}} A_{ij}\bar{X}_{ij} - 2\sum_{(i,j)\in\hat{W}} A_{ij} + \text{Trace}(\boldsymbol{A}_{VV}) \\ & & \text{subject to} & : & \sum_{(i,j)\in\hat{W}} \bar{X}_{ij} = \hat{N}, \\ & & & & 0 \leq \bar{X}_{ij} \leq 1 \text{ for } (i,j) \in \hat{W}. \end{array} \tag{3.10}$$

The structure of this problem is similar to (3.8), hence, it holds that $\rho_{LP} = 4\mathcal{S}_{\hat{N}}(\hat{A}_{\hat{N}}) - \boldsymbol{e}_V^T \boldsymbol{A}_{VV} \boldsymbol{e}_V + 2\mathrm{Trace}(\boldsymbol{A}_{VV})$.

Let $\hat{\boldsymbol{Y}}_{VV}$ be a part of an optimal solution of (3.9). From Assumption 3.1.2, it holds that

$$\boldsymbol{A}_{VV} \bullet \hat{\boldsymbol{Y}}_{VV} + 2\bar{\boldsymbol{c}}_F^T \hat{\boldsymbol{y}}_V = \rho_{LP} + 2\bar{\boldsymbol{c}}_F^T \hat{\boldsymbol{y}}_V = 4\mathcal{S}_{\hat{N}}(\hat{A}_{\hat{N}}) - \boldsymbol{e}_V^T \boldsymbol{A}_{VV} \boldsymbol{e}_V + 2\mathrm{Trace}(\boldsymbol{A}) + 2\bar{\boldsymbol{c}}_F^T \hat{\boldsymbol{y}}_V \leq 2\bar{\theta}.$$

Furthermore, $\hat{\boldsymbol{y}}_V$ satisfies $-1 \leq \hat{y}_i \leq 1$ for $i \in V$ and $\boldsymbol{e}_V^T \hat{\boldsymbol{y}}_V = \bar{N}$ by its definition and $\hat{\boldsymbol{Y}}_{VV}$ satisfies all the constraints of (3.9). Consequently, the pair $\hat{\boldsymbol{y}}_V$ and $\hat{\boldsymbol{Y}}_{VV}$ is a feasible solution of (3.6) and this leads to the inequality we wanted to obtain.

$$OPT_{LP} \geq \begin{pmatrix} 0 & \bar{\boldsymbol{g}}_V^T \\ \bar{\boldsymbol{g}}_V & \boldsymbol{O} \end{pmatrix} \bullet \begin{pmatrix} 1 & \hat{\boldsymbol{y}}_V^T \\ \hat{\boldsymbol{y}}_V & \hat{\boldsymbol{Y}}_{VV} \end{pmatrix} + \bar{g} = 2\bar{\boldsymbol{g}}_V^T \hat{\boldsymbol{y}}_V + \bar{g} = \boldsymbol{g}_V^T \hat{\boldsymbol{x}}_V + \boldsymbol{g}_F^T \boldsymbol{c}_F \geq OPT_{SOCP}.$$

□

Remark 3.1.4. *Since an optimal solution of a further relaxation problem of (3.6)*

$$\begin{array}{lll} \max & : & 2\bar{\boldsymbol{g}}_V^T \boldsymbol{y}_V + \bar{g} \\ \textit{subject to} & : & \boldsymbol{e}_V^T \boldsymbol{y}_V = \bar{N}, \\ & & -1 \leq y_i \leq 1 \textit{ for } i \in V \end{array}$$

is also $\hat{\boldsymbol{y}}_V$, its optimal value $2\bar{\boldsymbol{g}}_V^T \hat{\boldsymbol{y}}_V + \bar{\boldsymbol{g}}$ must be an upper bound of OPT_{LP}. On the other hand, from the proof of Lemma 3.1.3, when Assumption 3.1.2 holds, there exists some $\hat{\boldsymbol{Y}}_{VV} \in \mathbb{S}^{|V|}$ such that the pair of $\hat{\boldsymbol{y}}_V$ and $\hat{\boldsymbol{Y}}_{VV}$ is a feasible solution of (3.6) with the objective value $2\bar{\boldsymbol{g}}_V^T \hat{\boldsymbol{y}}_V + \bar{g}$. Therefore, $\hat{\boldsymbol{y}}_V$ is also an optimal solution of (3.6). This indicates that we can obtain the solution of (3.6) at the computation cost for sorting $\boldsymbol{g}_V$ instead of solving (3.6) as an LP problem, when Assumption 3.1.2 holds.

This remark implies that the LP relaxation (3.6) is not so tight against the original EDP (1.2). In contrast, we observed through preliminary numerical tests that the vector $\begin{pmatrix} \hat{\boldsymbol{x}}_V \\ \boldsymbol{c}_F \end{pmatrix}$ defined with an optimal solution $\hat{\boldsymbol{x}}_V$ of (3.8), is not always a feasible solution of (3.7) even if Assumption 3.1.2 holds. Therefore, the feasible region of the SOCP relaxation problem is strictly narrower than that of the LP relaxation problem, and the SOCP relaxation gives a tighter approximation than the LP relaxation in general, even though the relaxation was derived independently.

Remark 3.1.5. *The SDP relaxation problem (3.3) has no interior-feasible point.*

If a pair $\boldsymbol{y}_V \in \mathbb{R}^{|V|}$ and $\boldsymbol{Y}_{VV} \in \mathbb{S}^{|V|}$ satisfies all the constraint of (3.3), the pair is a feasible point. When the matrix $\begin{pmatrix} 1 & \boldsymbol{y}_V^T \\ \boldsymbol{y}_V & \boldsymbol{Y}_{VV} \end{pmatrix}$ is a positive-definite matrix for some feasible point $\boldsymbol{y}_V \in \mathbb{R}^{|V|}$ and $\boldsymbol{Y}_{VV} \in \mathbb{S}^{|V|}$, we say that (3.3) has an interior-feasible point. We can show that $\begin{pmatrix} 1 & \boldsymbol{y}_V^T \\ \boldsymbol{y}_V & \boldsymbol{Y}_{VV} \end{pmatrix}$ is not positive definite for any feasible point of (3.3). From the feasibility of (3.3), we have $\boldsymbol{e}_V^T \boldsymbol{y}_V = \bar{N}$ and $(\boldsymbol{e}_V \boldsymbol{e}_V^T) \bullet \boldsymbol{Y}_{VV} = \bar{N}^2$. If $\bar{N} \neq 0$, it holds

$$\begin{pmatrix} 1 \\ -\boldsymbol{e}_V/\bar{N} \end{pmatrix}^T \begin{pmatrix} 1 & \boldsymbol{y}_V^T \\ \boldsymbol{y}_V & \boldsymbol{Y}_{VV} \end{pmatrix} \begin{pmatrix} 1 \\ -\boldsymbol{e}_V/\bar{N} \end{pmatrix} = 1 - 2\boldsymbol{e}_V^T \boldsymbol{y}_V/\bar{N} + \boldsymbol{e}_V^T \boldsymbol{Y}_{VV} \boldsymbol{e}_V/\bar{N}^2 = 0.$$

In addition, for the case $\bar{N} = 0$, it holds

$$\begin{pmatrix} 0 \\ \boldsymbol{e}_V \end{pmatrix}^T \begin{pmatrix} 1 & \boldsymbol{y}_V^T \\ \boldsymbol{y}_V & \boldsymbol{Y}_{VV} \end{pmatrix} \begin{pmatrix} 0 \\ e_V \end{pmatrix} = \boldsymbol{e}_V^T \boldsymbol{Y}_{VV} \boldsymbol{e}_V = \bar{N}^2 = 0.$$

In either case, there exists a nonzero vector that makes the quadratic form zero, the matrix is not positive definite, therefore, (3.3) has no interior-feasible point.

Remark 3.1.6. *If we can apply the primal of the first level of SDSOS relaxations [2] to the SDP relaxation problem (3.3) instead of its dual, we can derive another SOCP relaxation problem. In this paper, we do not discuss this primal SOCP relaxation, since its feasible region is narrower than that of (3.3). When we denote the optimal value of the primal SOCP relaxation by OPT^P_{SOCP}, we can say $OPT^P_{SOCP} \leq OPT_{SDP}$. However, the relationship between OPT_{ED} and OPT^P_{SOCP} is unclear.*

3.2 Numerical Results

We report numerical result to compare the performance of the conic implementation with different approaches. The result is presented in Tables 3.2 and 3.3 for $N = 50$ and $N = 100$, respectively. All implementation was executed in a Windows PC with Core i7 3770K (3.5 GHz) and 32GB memory space. However, when the 32 GB memory space was insufficient, we used a Linux server with Opteron 4386 (3.10 GHz) and 128 GB memory space. The dataset were generated by the simulation `POPSIM` [32] or taken form `https://doi.org/10.5061/dryad.9pn5m` with the sizes of the test instances are $m = 2000, 1050, 2045, 5050, 10100,$ and 15222.

In the tables, the first column shows the algorithm of the conic relaxations. CR(LP) is the conic relaxation using LP approach 3.6 that is solved by `CPLEX 12.62`. CR(LP)-s does not solve the LP problem but obtain the same solution by sorting the values of $\boldsymbol{g}$ based on Remark 3.1.4. The SOCP relaxation denoted as CR(SOCP) and the SDP relaxation denoted as CR(SDP) are solved by `ECOS`[13] and `SDPT-3`[45], respectively. The second and the third columns are the number of genotype candidates m and the value of 2θ, respectively. We used different value of 2θ for each m. The last three columns are the obtained objective function $\boldsymbol{g}^T\boldsymbol{x}$, group coancestry $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$, and computation time in seconds. Note that the obtained objective function $\boldsymbol{g}^T\boldsymbol{x}$ shows the optimal value of each relaxation. More precisely, CR (LP) gives an optimal value for LP relaxation noted by OPT_{LP}; CR (SDP) and CR (SOCP) generate the optimal values OPT_{SDP} and OPT_{SOCP}, respectively.

Through Table 3.2 and 3.3, conic relaxation approaches somehow obtained infeasible solution due to constraint violation against the original EDP. Such results appear in the smallest problem $m = 200$; CR (LP) and CR (LP)-s generated $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} = 0.0574 \geq 0.0334 = 2\theta$ which violated the quadratic constraint $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} \leq 2\theta$. We can observe the same situation it for all m.

On the contrary, CR (SOCP) and CR (SDP) obtained the solutions, even if CR (SDP) required longer computation time as the problem size m increasing. Hence, the conic relaxation with SOCP implementation delivers to the efficiency. In addition, CR (SDP) obtained the objective value lower than that of CR (SOCP). For all $m \leq 5050$, the results always show $OPT_{SDP} \leq OPT_{SOCP} \leq OPT_{LP}$, which support the validity of Lemma 3.1.3. However, for large instances $m \geq 10100$, even if $\boldsymbol{g}^T\boldsymbol{x}$ of CR (SDP) is lower than CR (SOCP), the difference between both relaxations are too large. We observe that this inaccurate result is caused by a premature termination of `SDPT-3`, which induces the lack of interior-feasible points in the SDP relaxation problem (3.3) (see Remark 3.1.5). In contrast, the SOCP relaxation problem (3.7) provides high numerical stability, and this property can be regarded as an advantage of the SOCP relaxation approach.

Table 3.2: The comparison of the conic relaxation approaches ($N = 50$)

Algorithm	m	2θ	$\boldsymbol{g}^T\boldsymbol{x}$	$\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$	time (s)
CR (LP)	200	0.0334	28.068	0.0574	0.07
CR (LP)-s			28.068	0.0574	0.01
CR (SOCP)			26.156	0.0334	0.02
CR (SDP)			25.386	0.0321	1.29
CR (LP)	1050	0.0627	30.754	0.1362	0.37
CR (LP)-s			30.754	0.1362	0.01
CR (SOCP)			25.284	0.0627	0.08
CR (SDP)			24.938	0.0617	27.94
CR (LP)	2045	0.0711	504.217	0.4566	1.16
CR (LP)-s			504.217	0.4566	0.01
CR (SOCP)			439.353	0.0711	0.06
CR (SDP)			438.659	0.0706	145.57
CR (LP)	5050	0.1081	57.630	0.3672	10.17
CR (LP)-s			57.630	0.3672	0.01
CR (SOCP)			43.036	0.1081	0.21
CR (SDP)			42.786	0.0980	2221.22
CR (LP)	10100	0.0701	62.377	0.2368	46.84
CR (LP)-s			62.377	0.2368	0.01
CR (SOCP)			47.445	0.0701	0.54
CR (SDP)			21.265	0.0545	5577.80†
CR (LP)	15222	0.0388	603.783	0.4568	129.55
CR (LP)-s			603.783	0.4568	0.01
CR (SOCP)			468.367	0.0388	0.99
CR (SDP)			288.739	0.0195	17433.38†

† indicates numerical instability

Table 3.3: The comparison of the conic relaxation approaches ($N = 100$)

Algorithm	m	2θ	$\boldsymbol{g}^T\boldsymbol{x}$	$\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$	time (s)
CR (LP)	200	0.0258	24.654	0.0304	0.01
CR (LP)-s			24.654	0.0304	0.01
CR (SOCP)			24.015	0.0258	0.01
CR (SDP)			23.640	0.0255	0.82
CR (LP)	1050	0.0539	27.637	0.1214	0.39
CR (LP)-s			27.637	0.1214	0.01
CR (SOCP)			22.432	0.0539	0.08
CR (SDP)			22.358	0.0537	30.49
CR (LP)	2045	0.0628	478.114	0.4219	1.17
CR (LP)-s			478.114	0.4219	0.01
CR (SOCP)			421.696	0.0628	0.07
CR (SDP)			421.497	0.0627	165.33
CR (LP)	5050	0.0994	54.903	0.3355	10.35
CR (LP)-s			54.903	0.3355	0.01
CR (SOCP)			40.769	0.0995	0.25
CR (SDP)			40.711	0.0992	2164.49
CR (LP)	10100	0.0610	60.347	0.2245	46.71
CR (LP)-s			60.347	0.2245	0.01
CR (SOCP)			44.819	0.0610	0.70
CR (SDP)			21.374	0.0532	6750.06†
CR (LP)	15222	0.0300	575.227	0.4318	128.21
CR (LP)-s			575.227	0.4318	0.02
CR (SOCP)			444.730	0.0300	1.05
CR (SDP)			309.173	0.0228	19467.30†

† indicates numerical instability

4 Steepest-Ascent Method

We can obtain upper bounds of OPT_{ED} by the conic relaxation problems in the previous chapter, but these do not always output feasible solutions of the EDP (1.2). When SDP relaxation is employed, many methods rely on randomized algorithms to generate feasible solutions. However, the objective values obtained by a randomized algorithm is not so sharp (see Appendix 7.2), hence, we should develop a practical method that generates a feasible solution from the conic relaxations.

4.1 Primary Aspects of Steepest-Ascent Method

In contrast to mixed-integer linear programming (MI-LP) problems for which many solvers have been developed, a principal difficulty in the EDP (1.2) arises from the nonlinear constraint $\boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x} \leq 2\theta$. To obtain a sensible solution in a short time, we embed the violation against this constraint into the objective function as a penalty term using a penalty weight $\lambda \geq 0$ and focus on the following optimization problem

$$\begin{array}{lll} \max & : & f_\lambda(\boldsymbol{x}) := \boldsymbol{g}^T\boldsymbol{x} - \lambda \max\{\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} - 2\theta, 0\} \\ \text{subject to} & : & \boldsymbol{x} \in \hat{\mathcal{F}} \end{array} \tag{4.1}$$

where $\hat{\mathcal{F}} := \left\{\boldsymbol{x} \in \mathbb{R}^Z : \boldsymbol{e}^T\boldsymbol{x} = 1, \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}, x_1, \ldots, x_Z \in \left\{0, \frac{1}{N}\right\}\right\}$.

We give a validity of (4.1) by the next lemma which shows that if we take a large λ, this optimization problem with a penalty term (4.1) is equivalent to the original problem (1.2) .

Lemma 4.1.1. *Let $\boldsymbol{x}(\lambda) \in \mathbb{R}^Z$ be an optimal solution of (4.1). There exists a $\hat{\lambda} > 0$ such that $\boldsymbol{x}(\lambda)$ is an optimal solution of (1.2) for $\forall \lambda \geq \hat{\lambda}$.*

Proof. Let $\hat{\phi}$ be the optimal value of the following optimization problem:

$$\begin{array}{lllll} \hat{\phi} & := & \min & : & \max\{\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} - 2\theta, 0\} \\ & & \text{subject to} & : & \boldsymbol{x} \in \hat{\mathcal{F}}. \end{array} \tag{4.2}$$

From this definition, ϕ can take either zero or a positive number.

If $\hat{\phi} = 0$, the quadratic constraint $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} \leq 2\theta$ holds for $\forall \boldsymbol{x} \in \hat{\mathcal{F}}$. Therefore, this constraint vanishes from (1.2) and the penalty term in (4.1) has no effect. Hence, the two problems (1.2) and (4.1) are equivalent for any $\lambda \geq 0$.

For the case $\hat{\phi} > 0$, since $\hat{\mathcal{F}}$ is composed of a finite number of points, the reciprocal number of $\hat{\phi}$ is a finite number. Therefore, we can take $\hat{\lambda} = \frac{\max\{g_i : i=1,\ldots,Z\} - \min\{g_i : i=1,\ldots,Z\} + 1}{\hat{\phi}}$. To show this, we assume that $\boldsymbol{x}(\lambda)^T\boldsymbol{A}\boldsymbol{x}(\lambda) \leq 2\theta$ does not hold for $\lambda \geq \hat{\lambda}$. We then have

$$\boldsymbol{g}^T\boldsymbol{x}(\lambda) - \lambda\left(\boldsymbol{x}(\lambda)^T\boldsymbol{A}\boldsymbol{x}(\lambda) - 2\theta\right) \leq \max\{g_i : i = 1, \ldots, Z\} - \hat{\lambda}\hat{\phi} < \min\{g_i : i = 1, \ldots, Z\}.$$

Here, we used $\boldsymbol{e}^T\boldsymbol{x}(\lambda) = 1$ and $\boldsymbol{x}(\lambda) \geq \boldsymbol{0}$, since $\boldsymbol{x}(\lambda) \in \hat{\mathcal{F}}$. On the contrary, from the assumption that (1.2) has a feasible point, the objective value of (4.1) at this feasible point is at least $\min\{g_i : i = 1, \ldots, Z\}$. This indicates that $\boldsymbol{x}(\lambda)$ cannot be an optimal solution of (4.1) if $\boldsymbol{x}(\lambda)^T\boldsymbol{A}\boldsymbol{x}(\lambda) > 2\theta$. Therefore, we can restrict the feasible region of (4.1) to the set $\{\boldsymbol{x} \in \mathbb{R}^Z : \boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} \leq 2\theta\} \cap \hat{\mathcal{F}}$, and the objective function of (4.1) is reduced to $\boldsymbol{g}^T\boldsymbol{x}$. Consequently, the optimal solution of (4.1) is also optimal for (1.2). □ □

Since the computation for $\hat{\phi}$ is almost as hard as the original EDP, it is not practical to compute $\hat{\phi}$. In addition, when we maximize $f_\lambda(\boldsymbol{x})$, extremely large λ makes the computation numerically unstable. As an appropriate value for the penalty weight λ, we make the use of the Lagrangian multiplier λ_0 developed in Meuwissen [28];

$$\lambda_0 := \sqrt{\frac{(\boldsymbol{g}^T\boldsymbol{A}^{-1}\boldsymbol{g})(\boldsymbol{e}^T\boldsymbol{A}^{-1}\boldsymbol{e}) - (\boldsymbol{g}^T\boldsymbol{A}^{-1}\boldsymbol{e})^2}{8\theta(\boldsymbol{e}^T\boldsymbol{A}^{-1}\boldsymbol{e}) - 4}}.$$

This λ_0 corresponds to the Lagrangian multiplier of the constraint $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} = 2\theta$ in the following optimization problem;

$$\begin{array}{lcl} \max & : & \boldsymbol{g}^T\boldsymbol{x} \\ \text{subject to} & : & \boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} = 2\theta, \\ & & \boldsymbol{e}^T\boldsymbol{x} = 1. \end{array} \tag{4.3}$$

The approach of [28] first solves (4.3), then applies some heuristic method to obtain a solution of (1.2). Therefore, a maximization of $\boldsymbol{g}^T\boldsymbol{x} - \lambda_0(\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} - 2\theta)$ over $\boldsymbol{e}^T\boldsymbol{x} = 1$ is a natural derivation when we consider (4.3). For $f_\lambda(\boldsymbol{x})$, we often employ λ such that $\lambda \geq \lambda_0$, since we put a strong emphasis on the violation with respect to $\max\{\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} - 2\theta, 0\}$.

We now discuss (4.1) from the viewpoint of convex functions. The function $-f_\lambda(\boldsymbol{x})$ is a convex function in the continuous space $\mathbb{R}^Z$, since $\boldsymbol{A} \succeq \boldsymbol{O}$ and $\lambda \geq 0$. Hence, the problem (4.1) can be cast as a minimization of a convex function over a discrete feasible set.

An M-convex function [33] is a discrete convex function defined on a set in which the sum of the elements of a feasible point is constant. A steepest-descent method for M-convex functions was developed in [34]. When an M-convex function $f^M(\boldsymbol{x})$ with a feasible set $\mathcal{F}^M$ is given, the steepest-descent method starts from an initial point $\boldsymbol{x}^0 \in \mathcal{F}^M$, and finds the next point $\boldsymbol{x}^1$ from a neighborhood $\mathcal{N}(\boldsymbol{x}^0) \subset \mathcal{F}^M$ that decreases the objective function $f^M(\boldsymbol{x})$ with the largest margin. Here, $\mathcal{N}(\boldsymbol{x}^0) := \{\boldsymbol{x} + \boldsymbol{e}_i - \boldsymbol{e}_j \in \mathcal{F}^M : i, j = 1, \ldots, n\}$. In other words, $\boldsymbol{x}^1$ is chosen so that $f^M(\boldsymbol{x}^1) \leq f^M(\boldsymbol{x})$ for any $\boldsymbol{x} \in \mathcal{N}(\boldsymbol{x}^0)$. The steepest-descent method continues the search in neighbors, and it eventually can find a global minimizer since any local minimizer is a global minimizer when the objective function f^M is an M-convex function.

When we applied the steepest-descent method to $-f_\lambda(\boldsymbol{x})$ starting from multiple randomly-generated initial points $\boldsymbol{x}^0$, we numerically found multiple local minimizers that were not always global minimizers. Though $-f_\lambda(\boldsymbol{x})$ is not an M-convex function, the optimization problem with the penalty term (4.1) resembles the minimization of an M-convex function. In particular, the feasible set $\hat{\mathcal{F}}$ satisfies $\sum_{i=1}^m x_i = 1$ and the function $-f_\lambda(\boldsymbol{x})$ is a convex function in the continuous space $\mathbb{R}^m$. Therefore, we can expect that the steepest-descent method for M-convex functions will give good direction to solve (4.1). Furthermore, we can exploit the solution obtained by the conic relaxation problems in Chapter 3 to generate a starting point $\boldsymbol{x}^0$.

When we adjust the steepest-descent method implemented in the software package ODICON [47] to solve (4.1), we obtain Algorithm 4.1.2. Since (4.1) is a maximization problem, Algorithm 4.1.2 is a steepest-ascent method.

Algorithm 4.1.2. A steepest-ascent method with a conic relaxation problem for the optimization problem with the penalty term arising from the EDP:

Step 1 Solve a conic relaxation problem (3.3), (3.6) or (3.7). If (3.7) is solved, let $\boldsymbol{x}^*$ be its optimal solution. For (3.6) and (3.3), let $\boldsymbol{y}^*_V$ be its optimal solution and set $\boldsymbol{x}^*$ by $\boldsymbol{x}^*_V := \boldsymbol{y}^*_V$ and $\boldsymbol{x}^*_F := \boldsymbol{c}_F$.

Step 2 By sorting $\boldsymbol{x}^*$, separate V into the two disjoint set $V_{\frac{1}{N}}$ and V_0 such that $x^*_i \geq x^*_j$ for $i \in V_{\frac{1}{N}}, j \in V_0$ and that $|V_{\frac{1}{N}}| = N - p$ (ties are broken arbitrarily). Set the initial point $\boldsymbol{x}^0 \in \mathbb{R}^m$ by $x^0_i := \frac{1}{N}$ for $i \in V_{\frac{1}{N}}$, $x^0_j := 0$ for $j \in V_0$, and $\boldsymbol{x}^0_F := \boldsymbol{c}_F$. Set the iteration counter $h := 0$.

Step 3 Select the steepest swap $i^h \in V_{\frac{1}{N}}$ and $j^h \in V_0$ such that

$$f_\lambda(\boldsymbol{x}^h - \frac{1}{N}\boldsymbol{e}_{i^h} + \frac{1}{N}\boldsymbol{e}_{j^h}) \geq f_\lambda(\boldsymbol{x}^h - \frac{1}{N}\boldsymbol{e}_i + \frac{1}{N}\boldsymbol{e}_j) \text{ for } i \in V_{\frac{1}{N}}, j \in V_0.$$

Step 4 If there is no improvement, that is $f_\lambda(\boldsymbol{x}^h - \frac{1}{N}\boldsymbol{e}_{i^h} + \frac{1}{N}\boldsymbol{e}_{j^h}) \leq f_\lambda(\boldsymbol{x}^h)$, output $\boldsymbol{x}^h$ as a solution and stop.

Step 5 Set $\boldsymbol{x}^{h+1} := \boldsymbol{x}^h - \frac{1}{N}\boldsymbol{e}_{i^h} + \frac{1}{N}\boldsymbol{e}_{j^h}$. Swap i^h and j^h by $V_{\frac{1}{N}} := V_{\frac{1}{N}} \cup \{j^h\}\backslash\{i^h\}$ and $V_0 := V_0 \cup \{i^h\}\backslash\{j^h\}$. Set $h := h + 1$ and return to **Step 3**.

In Step 2, the number of $\frac{1}{N}$ in $\boldsymbol{x}^0$ is exactly N. Due to Step 5, this property is kept through the iterations in the algorithm, hence, the number of $\frac{1}{N}$ in $\boldsymbol{x}^h$ is also exactly N for any $h \geq 1$. When no improvement can be found, the algorithm stops by Step 4.

Most of the computation cost for each iteration in Algorithm 4.1.2 is consumed at the evaluations of f_λ in Step 3. The number of the evaluations is determined by the size of neighbor around $\boldsymbol{x}^h$, that is, $|V_{\frac{1}{N}}| \times |V_0| = (N - p) \times (|V| - (N - p))$. Therefore, the case $N - p = \frac{|V|}{2}$ requires the heaviest computation cost. Furthermore, to reduce the computation cost, we focus the evaluation of $(\boldsymbol{x}^h - \frac{1}{N}\boldsymbol{e}_{i^h} + \frac{1}{N}\boldsymbol{e}_{j^h})^T\boldsymbol{A}(\boldsymbol{x}^h - \frac{1}{N}\boldsymbol{e}_{i^h} + \frac{1}{N}\boldsymbol{e}_{j^h})$. Since ODICON was designed to handle general functions, it accessed all the elements of $\boldsymbol{A}$ for each i^h and j^h. By expanding the part as $(\boldsymbol{x}^h)^T\boldsymbol{A}(\boldsymbol{x}^h) + \frac{2}{N}\left(-\boldsymbol{e}_{i^h} + \boldsymbol{e}_{j^h}\right)^T(\boldsymbol{A}\boldsymbol{x}^h) + \frac{1}{N^2}\left(A_{i^hi^h} + A_{j^hj^h} - 2A_{i^hj^h}\right)$, we evaluate $(\boldsymbol{x}^h)^T\boldsymbol{A}(\boldsymbol{x}^h)$ and $(\boldsymbol{A}\boldsymbol{x}^h)$ only once for each iteration of Algorithm 4.1.2. This saves 95% of the computation time for Step 3 compared to ODICON.

4.2 Numerical Results

This section presents numerical results of steepest-ascent (SA) method using different conic relaxation approaches compared to the existing solvers `GENCONT` and `dsOpt` embedded in `OPSEL`, and `CPLEX`. We executed all methods using the same platform and the data as mentioned in Chapter 3. For the SA method, we set the penalty weight $\lambda = 2\lambda_0$ in the function $f_\lambda(x)$ of Section 4.1.

As in the first column of Tables 4.1 and 4.2, the steepest-ascent algorithm uses conic relaxation to generate an initial point. More precisely, SA(LP) is the result of Algorithm 4.1.2 that starts from the solution of CR(LP) as its initial solution. While the same rule is applied to SA(SOCP) and SA(SDP), SA(LP)-s is a result of applying Remark 3.1.4 to (3.6) and generating its solution by sorting $\boldsymbol{g}_v$. The sixth column indicates the number of iterations for Algorithm 4.1.2. The seventh is the value of $f_\lambda(\boldsymbol{x}^h)$ with h being the iteration number in the sixth column. The last column is the SA computation time in seconds which is the total time in Algorithm 4.1.2.

In contrast to the conic relaxation, the steepest-ascent method obtains a feasible solution of the EDP (1.2) since it satisfies all required constraints including the quadratic constraint $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} \leq 2\theta$.

Table 4.1: The comparison of the steepest-ascent approaches ($N = 50$)

Algorithm	m	2θ	$\boldsymbol{g}^T\boldsymbol{x}$	$\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$	iter (h)	$f_\lambda(\boldsymbol{x}^h)$	time (s)
SA (LP)	200	0.0334	25.029	0.0334	21	25.029	0.10
SA (LP)-s			25.029	0.0334	21	25.029	0.04
SA (SOCP)			25.090	0.0334	13	25.090	0.06
SA (SDP)			25.207	0.0334	4	25.207	1.30
SA (LP)	1050	0.0627	22.707	0.0627	23	22.707	0.51
SA (LP)-s			22.707	0.0627	23	22.707	0.15
SA (SOCP)			24.831	0.0627	2	24.831	0.09
SA (SDP)			24.846	0.0627	2	24.846	27.96
SA (LP)	2045	0.0711	414.591	0.0710	32	414.591	1.47
SA (LP)-s			414.591	0.0710	32	414.591	0.32
SA (SOCP)			438.386	0.0710	2	438.386	0.09
SA (SDP)			438.457	0.0710	1	438.457	145.59
SA (LP)	5050	0.1081	38.696	0.1080	23	38.696	11.17
SA (LP)-s			38.696	0.1080	23	38.696	0.98
SA (SOCP)			42.691	0.1080	3	42.691	0.37
SA (SDP)			42.431	0.1080	3	42.431	2221.40
SA (LP)	10100	0.0701	41.284	0.0701	32	41.284	49.87
SA (LP)-s			41.284	0.0701	32	41.284	3.29
SA (SOCP)			46.568	0.0701	2	46.568	0.87
SA (SDP)			44.662	0.0701	45	44.662	5582.46†
SA (LP)	15222	0.0388	438.791	0.0388	42	438.791	139.03
SA (LP)-s			438.791	0.0388	42	438.791	6.45
SA (SOCP)			460.769	0.0388	9	460.769	2.56
SA (SDP)			460.409	0.0388	43	460.409	17441.93†

† indicates numerical instability

The smallest case $m = 200, N = 50$ of the SA(SDP) attains a remarkable result and it holds $OPT_{ED} \leq OPT_{SDP}$ from Lemma 3.1.3. Hence, we can guarantee that 25.207 (SA(SDP))$\leq$ $OPT_{ED} \leq$ 25.386 (CR(SDP)); and the optimal value of EDP is up to an error 0.710%.

The next viewpoint is that the violations of the solutions generated by the SA methods against $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$ are remarkably small. This is mainly because the value λ based on the Lagrangian multiplier $lambda_0$ is appropriate. Therefore, the maximization of the function with the penalty term (4.1) can provide a suitable solution for the EDP (1.2).

A comparison of SA results for $m \leq 5050$ starting with the conic relaxations indicates that $\boldsymbol{g}^T\boldsymbol{x}$ of SA(SDP) and SA(SOCP) are close to each other. This implies that the SOCP and SDP relaxation problems can provide eligible starting points for the proposed SA method. This point is also reflected in the numbers of iterations for the SA method; those of SA(SDP) and SA(SOCP) are much less than that of SA (SOCP). For example, in the case of $m = 2045, N = 50$, while SA(SOCP) and SA(SDP) required only two and one iteration, respectively, SA(LP) required 32 iterations. Therefore, we can infer that the solution of the SOCP and SDP relaxation problems are close to the local maximizer of $f_\lambda(x^h)$.

Regarding the computation time, the computation time of SA(SOCP) is much shorter than SA(SDP). This was mainly because we can aggressively exploit the structure of the inverse of the Wright's numerator matrix $\boldsymbol{A}$. Since the objective values $\boldsymbol{g}^T\boldsymbol{x}$ of SA(SOCP) are competitive, SA(SOCP) can be considered as the most efficient approach among SA(LP), SA(SOCP), and SA(SDP).

Referring the above results, we chose SA(SOCP) in a comparison with the existing solvers,

Table 4.2: The comparison of the steepest-ascent approaches ($N = 100$)

Algorithm	m	2θ	$\boldsymbol{g}^T\boldsymbol{x}$	$\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$	iter (h)	$f_\lambda(\boldsymbol{x}^h)$	time (s)
SA (LP)	200	0.0258	23.355	0.0258	18	23.355	0.07
SA (LP)-s			23.355	0.0258	18	23.355	0.06
SA (SOCP)			23.412	0.0258	13	23.412	0.08
SA (SDP)			23.521	0.0258	5	23.521	0.84
SA (LP)	1050	0.0539	19.805	0.0539	42	19.805	0.98
SA (LP)-s			19.805	0.0539	42	19.805	0.590
SA (SOCP)			22.321	0.0539	5	22.321	0.15
SA (SDP)			22.324	0.0539	3	22.324	30.54
SA (LP)	2045	0.0628	406.348	0.0628	65	406.348	5.00
SA (LP)-s			406.348	0.0628	65	406.348	3.78
SA (SOCP)			421.113	0.0627	3	421.113	0.25
SA (SDP)			421.425	0.0628	2	421.425	165.46
SA (LP)	5050	0.0994	36.509	0.0994	50	36.509	17.44
SA (LP)-s			36.509	0.0994	50	36.509	7.03
SA (SOCP)			40.629	0.0995	3	40.629	0.71
SA (SDP)			40.690	0.0994	2	40.690	2164.85
SA (LP)	10100	0.0610	39.911	0.0610	62	39.911	68.54
SA (LP)-s			39.911	0.0610	62	39.911	21.51
SA (SOCP)			44.522	0.0610	7	44.522	3.12
SA (SDP)			42.810	0.0610	66	42.810	6773.05†
SA (LP)	15222	0.0300	408.725	0.0300	90	408.725	185.33
SA (LP)-s			408.725	0.0300	90	408.725	49.14
SA (SOCP)			441.438	0.0300	6	441.438	4.72
SA (SDP)			406.266	0.0300	92	406.266	19525.52†

† indicates numerical instability

GENCONT, dsOpt embedded in OPSEL, and CPLEX. Since textttdsOpt and CPLEX utilized the branch-and-bound framework, we can expect their solution to be close to the real optimal solution of the EDP (1.2).

Tables 4.3 and 4.4 present the comparison of the proposed Algorithm 4.1.2 (SA (SOCP)), GENCONT and dsOpt, CPLEX for $N = 50$ and $N = 100$ (the numbers of selected genotypes). We tried to execute GENCONT2, a new version of GENCONT, but its binary file did not work on our computational environment. Thus, we used GENCONT1 for the comparison.

In the tables, $f_\lambda(\boldsymbol{x})$ is evaluated in seventh column at the obtained solution of each algorithm; and the eighth column reports the number of selected genotypes $\#N_s$ for $|\{i : x_i > 0\}|$. For dsOpt and CPLEX, we limit the computation time to three hours and set the tolerance gap as 1%. When dsOpt and CPLEX reached the time limit, we obtained a feasible solution from dsOpt, but we could not extract sensible solutions from CPLEX. GENCONT could not solve large instances ($m = 10100$ and $m = 15222$) due to out of memory.

Table 4.3: The comparison of the existing methods and the proposed algorithm ($N = 50$)

Algorithm	m	2θ	$\boldsymbol{g}^T\boldsymbol{x}$	$\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$	$f_\lambda(\boldsymbol{x})$	$\#N_s$	time (s)
GENCONT	200	0.0334	25.290	0.0342	20.087	50	0.06
dsOpt			25.191	0.0334	25.191	50	1779.13
CPLEX			25.190	0.0334	25.190	50	4270.77
SA (SOCP)			25.090	0.0334	25.090	50	0.06
GENCONT	1050	0.0627	24.983	0.0627	24.983	48	7.91
dsOpt			24.858	0.0627	24.858	50	> 10800
CPLEX			Cannot obtain a sensible solution in 3 hours				> 10800
SA (SOCP)			24.831	0.0627	24.831	50	0.09
GENCONT	2045	0.0711	437.049	0.0694	437.049	50	88.46
dsOpt			435.826	0.0692	435.826	50	16.08
CPLEX			436.213	0.0680	436.212	50	0.37
SA (SOCP)			438.386	0.0710	438.386	50	0.09
GENCONT	5050	0.1081	42.780	0.1089	-306.701	50	1769.72
dsOpt			42.702	0.1081	42.702	50	> 10800
CPLEX			42.456	0.1066	42.456	50	2.02
SA (SOCP)			42.691	0.1080	42.691	50	0.37
GENCONT	10100	0.0701	Out of memory				
dsOpt			46.252	0.0700	46.252	50	> 10800
CPLEX			Cannot obtain a sensible solution in 3 hours				> 10800
SA (SOCP)			46.568	0.0701	46.568	50	0.87
GENCONT	15222	0.0388	Out of memory				
dsOpt			459.040	0.0388	459.040	50	> 10800
CPLEX			459.135	0.0386	459.135	50	39.20
SA (SOCP)			460.769	0.0388	460.769	50	2.56

From the numerical results in Tables 4.3 and 4.4, the objective values $\boldsymbol{g}^T\boldsymbol{x}$ of SA(SOCP) are close to those of GENCONT, dsOpt, and CPLEX. Since the cost vector $\boldsymbol{g}$ in the objective function is usually generated from a statistical procedure, the difference in the objective values makes little difference for practical use. However, GENCONT sometimes failed to satisfy the constraints; the quadratic constraint $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} \leq 2\theta$ was violated in the $m = 200$ or $m = 5050$ cases, and the number of chosen genotypes did not match the input N. Therefore, the quality of SA(SOCP) was superior to that of GENCONT.

The computation time shows that SA(SOCP) is much faster than GENCONT. As an example, for $m = 5050$, SA(SOCP) used less than one second, while GENCONT required 1700 seconds

Table 4.4: The comparison of the existing methods and the proposed algorithm ($N = 100$)

Algorithm	m	2θ	$\boldsymbol{g}^T\boldsymbol{x}$	$\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$	$f_\lambda(\boldsymbol{x})$	$\#N_s$	time (s)
GENCONT	200	0.0258	23.640	0.0261	21.253	100	0.07
dsOpt			23.551	0.0258	23.551	100	> 10800
CPLEX			23.508	0.0258	23.508	100	1.42
SA (SOCP)			23.412	0.0258	23.412	100	0.08
GENCONT	1050	0.0539	22.749	0.0539	22.749	91	9.63
dsOpt			22.275	0.0539	22.275	100	304.89
CPLEX			Cannot obtain a sensible solution in 3 hours				> 10800
SA (SOCP)			22.321	0.0539	22.321	100	0.15
GENCONT	2045	0.0628	421.005	0.0632	392.893	100	105.40
dsOpt			419.600	0.0613	419.600	100	6.85
CPLEX			420.748	0.0619	420.748	100	0.41
SA (SOCP)			421.113	0.0627	421.113	100	0.25
GENCONT	5050	0.0995	40.692	0.0997	40.692	100	1940.43
dsOpt			40.468	0.0994	40.468	100	50.46
CPLEX			Cannot obtain a sensible solution in 3 hours				> 10800
SA (SOCP)			40.629	0.0995	40.629	100	0.71
GENCONT	10100	0.0610	Out of memory				
dsOpt			44.467	0.0696	44.467	100	> 10800
CPLEX			Cannot obtain a sensible solution in 3 hours				> 10800
SA (SOCP)			44.522	0.0610	44.522	100	3.12
GENCONT	15222	0.0300	Out of memory				
dsOpt			441.770	0.0300	441.770	100	> 10800
CPLEX			440.996	0.0290	440.99640	100	7.45
SA (SOCP)			441.438	0.0300	441.438	100	4.72

(almost half an hour). SA(SOCP) returned a solution in 5 seconds for each instance, even for the largest instance $m = 15222$. The computation times of the branch-and-bound framework were unpredictable. In $N = 50$, dsOpt and CPLEX required longer computation for a small problem $m = 200$ than for a large problem $m = 2045$. It is very difficult to estimate the computation time required by dsOpt and CPLEX in advance due to a nature of the branch-and-bound framework. In contrast, SA(SOCP) consumed longer computation time for larger problems and this property is favorable for practical use.

5 Lifted Polyhedral Programming

The previous chapter gives the fact that SOCP relaxation contributes to the efficiency when solving the EDP. Since the previous methods are only the iterative method that (somehow) cannot guarantee the feasible solution, this chapter explain a lifted polyhedral programming (LPP) relaxation based on the SOCP properties. In addition, we proposed an improvement of LPP that utilize an active constraint selection method to removes the non-linearity generated by LPP.

5.1 Lifted Polyhedral Programming Relaxation

Lifted polyhedral programming (LPP) relaxation [8, 48] is an approach to solve MI-SOCP problems by employing a polyhedral relaxation. Instead of an MI-SOCP problem that involves difficult non-linear constraints, we solve a mixed-integer *linear* programming problem (MI-LP) as the resultant problem.

In LPP relaxation, many hyper-planes are generated for constructing a polyhedral cone $\mathscr{P}_\epsilon^m$ to approximate $\mathscr{K}^m$. Here, $\epsilon > 0$ is a parameter to control the tightness of LPP relaxation. Vielma et al. [48] gave the detailed formulation of $\mathscr{P}_\epsilon^m$. They first decompose the $(m+1)$-dimensional second-order cone $\mathscr{K}^m$ into multiple 2-dimensional second-order cones $\mathscr{K}^2$ and linear constraints:

$$
\begin{aligned}
\mathscr{K}^m := \{ & (v_0, \boldsymbol{v}) \in \mathbb{R}_+ \times \mathbb{R}^m : \exists (\delta^j)_{j=0}^J \in \mathbb{R}^{T(m)} \text{ such that} \\
& v_0 = \delta_1^J, \\
& \delta_i^0 = v_i \ (i = 1, \cdots, m), \\
& \left(\delta_{2i-1}^j, \delta_{2i}^j, \delta_i^{j+1} \right) \in \mathscr{K}^2 \ \left(i = 1, \cdots, \left\lfloor \frac{t_j}{2} \right\rfloor, \ j = 0, \cdots, J-1 \right), \\
& \delta_{t_j}^j = \delta_{\lceil t_j/2 \rceil}^{j+1} \text{ for } j = 0, \cdots, J-1 \text{ such that } t_j \text{ is odd} \}
\end{aligned}
\tag{5.1}
$$

with $J = \lceil \log_2 m \rceil$, and $\{t_j\}_{j=0}^J$ is defined recursively as $t_0 = m$ and $t_{j+1} = \left\lceil \frac{t_j}{2} \right\rceil$ for $j = 0, \ldots, J-1$ so that $T(m) = \sum_{j=0}^{J-1} \left\lfloor \frac{t_j}{2} \right\rfloor$. For example, $\mathscr{K}^4$ is determined by a quadratic constraint $v_0^2 \geq v_1^2 + v_2^2 + v_3^2 + v_4^2$. Based on the above decomposition, this constraint is decomposed into $T(4) = \left\lfloor \frac{4}{2} \right\rfloor + \left\lfloor \frac{2}{2} \right\rfloor = 3$ cones of $\mathscr{K}^2$; $v_0^2 \geq (\delta_1^1)^2 + (\delta_2^1)^2$, $(\delta_1^1)^2 \geq v_1^2 + v_2^2$ and $(\delta_2^1)^2 \geq v_3^2 + v_4^2$.

Then, a replacement of $\mathcal{K}^2$ in (5.1) by $\mathcal{W}_j(\epsilon)$ defined below generates $\mathcal{P}_\epsilon^m$:

$$\begin{aligned}\mathcal{W}_j(\epsilon) := \Big\{ & (v_0, v_1, v_2) \in \mathbb{R}_+ \times \mathbb{R}^2 : \exists (\alpha, \beta) \in \mathbb{R}^{2s_j(\epsilon)} \text{ such that} \\ & v_0 = \alpha_{s_j(\epsilon)} \cos\left(\frac{\pi}{2^{s_j(\epsilon)}}\right) + \beta_{s_j(\epsilon)} \sin\left(\frac{\pi}{2^{s_j(\epsilon)}}\right),\ \alpha_1 = v_1 \cos(\pi) + v_2 \sin(\pi), \\ & \beta_1 \geq |v_2 \cos(\pi) - v_1 \sin(\pi)|,\ \alpha_{i+1} = \alpha_i \cos\left(\frac{\pi}{2^i}\right) + \beta_i \sin\left(\frac{\pi}{2^i}\right), \\ & \beta_{i+1} \geq \left|\beta_i \cos\left(\frac{\pi}{2^i}\right) - \alpha_i \sin\left(\frac{\pi}{2^i}\right)\right|,\ \text{for } i \in \{1, \ldots, s_j(\epsilon) - 1\}\Big\}\end{aligned} \tag{5.2}$$

where

$$s_j(\epsilon) = \left\lceil \frac{j+1}{2} \right\rceil - \left\lceil \log_4 \left(\frac{16}{9} \pi^{-2} \log(1+\epsilon) \right) \right\rceil \text{ for } j \in \{0, \ldots, J-1\}.$$

The polyhedral cone $\mathcal{P}_\epsilon^m$ is a good approximation of $\mathcal{K}^m$ in the sense that $\mathcal{P}_\epsilon^m$ is wedged between $\mathcal{K}^m$ and $\mathcal{K}_\epsilon^m$, where $\mathcal{K}_\epsilon^m$ is an ϵ extension of $\mathcal{K}^m$ defined by $\mathcal{K}_\epsilon^m = \{(v_0, \boldsymbol{v}) \in \mathbb{R}_+ \times \mathbb{R}^m : ||\boldsymbol{v}||_2 \leq (1+\epsilon) v_0\}$. More precisely, $\mathcal{P}_\epsilon^m$ satisfies $\mathcal{K}^m \subsetneq \mathcal{P}_\epsilon^m \subsetneq \mathcal{K}_\epsilon^m$ as proven in [6].

Naturally, when we take smaller ϵ, the relaxation $\mathcal{P}_\epsilon^m$ becomes tighter, but we require more hyper planes to build $\mathcal{P}_\epsilon^m$. In particular, the number of linear constraints in $\mathcal{W}_j(\epsilon)$ is $4 + 3(s_j(\epsilon) - 1)$, when we divide each linear inequality that involves absolute values into two linear inequalities. The number of linear inequalities will be larger as we set a tighter ϵ. Therefore, another implementation is needed to reduce the larger number of added linear inequalities.

5.2 LPP with Active Constraint Selection Method

We propose a conversion of active linear constraints at the solution into equality constraints. In an optimization problem P, inequality constraints $\boldsymbol{a}_i^T \boldsymbol{x} \leq b_i$ $(i = 1, \ldots, p)$ with $\boldsymbol{a}_i \in \mathbb{R}^q$ and $b_i \in \mathbb{R}$ $(i = 1, \ldots, p)$ are called active constraints at an optimal solution $\boldsymbol{x}^*$ if $\boldsymbol{a}_i^T \boldsymbol{x}^* = b_i$. For the EDPs, we conducted preliminary experiments changing the parameter ϵ as 0.04, 0.05, and 0.08 and found that the constraints of form

$$\beta_1 \geq -v_2 \cos(\pi) + v_1 \sin(\pi) \tag{5.3}$$

in (5.2) were always active at $\boldsymbol{x}^*(\epsilon)$, where $\boldsymbol{x}^*(\epsilon)$ is the optimal solution for the LPP relaxation with ϵ. Therefore, we numerically checked the activeness of the inequality (5.3).

Based on this observation, we propose an LPP relaxation with an active constraint selection method (LPP-ACSM) to solve a restricted formulation of the EDP (1.3).

Algorithm 5.2.1. [LPP relaxation with active constraint selection method (LPP-ACSM)]

Step 1 Build an MI-LP problem by replacing $\mathcal{K}^m$ in (1.3) with $\mathcal{P}_\epsilon^m$.

Step 2 Replace the inequality constraints of form $\beta_1 \geq |v_2 \cos(\pi) - v_1 \sin(\pi)|$ in (5.2) with the corresponding equality constraints $\beta_1 = -v_2 \cos(\pi) + v_1 \sin(\pi)$.

Step 3 Solve the MI-LP problem generated in **Step 2.**

Note that $\mathcal{K}^m$ is decomposed into $T(m)$ cones of $\mathcal{K}^2$ in (5.1). In each $\mathcal{K}^2$, we reduce one linear constraint by applying LPP-ACSM. Therefore, the number of constraints reduced in the LPP-ACSM is $T(m)$ and we could expect a reduction in computation time.

5.3 Numerical Results

We conduct a numerical experiment to compare the LPP and the proposed LPP-ACSM implementations for both $N = 50$ and $N = 100$. Here, the same data and parameter set with the previous Chapters 3 and 4 were used to observe the performance. Here, we implemented them using a 64-bit Windows 10 PC with Xeon CPU E3-1231 (3.40 GHz) and 8 GB memory space, with `CPLEX 12.7.1` as a solver for the MI-LP problems.

Table 5.1: Numerical comparison between LPP and LPP-ACSM ($N = 50$)

method	m	2θ	$(1+\epsilon)2\theta$	gap	$\boldsymbol{g}^T\boldsymbol{x}$	$\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$	time (sec)
LPP	200	0.0334	0.03373	5%	24.83	0.03340	26.81
LPP-ACSM					24.84	0.03340	47.75
LPP				1%	25.11	0.03340	3691.26
LPP-ACSM					25.15	0.03340	2587.16
LPP	1050	0.0627	0.06333	5%	24.54	0.06129	976.54
LPP-ACSM					24.72	0.06215	1230.01
LPP				1%	24.89	0.06291	10063.39
LPP-ACSM					24.89	0.06291	1634.58

Table 5.2: Numerical comparison between LPP and LPP-ACSM ($N = 100$)

method	m	2θ	$(1+\epsilon)2\theta$	gap	$\boldsymbol{g}^T\boldsymbol{x}$	$\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$	time (sec)
LPP	200	0.0258	0.026059	5%	23.38	0.02585	10.60
LPP-ACSM					23.24	0.02580	11.35
LPP				1%	23.53	0.02585	2289.65
LPP-ACSM					23.58	0.02585	4003.32
LPP	1050	0.0539	0.054440	5%	22.35	0.05401	1047.81
LPP-ACSM					22.34	0.05392	1059.46
LPP				1%	22.35	0.05401	1088.16
LPP-ACSM							OOM

In Tables 5.1 and 5.2, the fourth column shows the relaxed 2θ by the use of $\epsilon > 0$. For this experiment, we fixed $\epsilon = 0.005$ for generating $\mathcal{P}_\epsilon^m$, so that these two methods output feasible solutions.

We expected earlier that LPP-ACSM could reduce the number of constraints. Actually, in Table 5.1, LPP-ASCM reduce 6 times of LPP computation in $m = 1050$ and gap $= 1\%$. However, excepting this result, the two tables indicate that LPP-ACSM turn out to be slower than LPP itself. LPP-ACSM even failed to obtain the feasible solution due to out of memory (OOM) which shown in Table 5.2 for $m = 1050$ and gap $= 1\%$. The insufficiency of the memory space was not observed only for this case but also for $m \geq 2045$. Here, we presented only the results for $m = 200, 1045$ because of the out of memeory in LPP-ACSM.

A difficulty in implementing LPP and LPP-ACSM is to choose the right ϵ, since a very tight ϵ leads to a larger number of constraints. Therefore, we propose other approaches that also exploit SOCP properties to solve EDP 1.2 in the next chapter.

6 Cone Decomposition Method

This chapter explains our proposed cone decomposition method (CDM) for solving EDP (1.2). We verified in Chapter 4 that the implementation with SOCP relaxation is more efficient compared to others. Using this fact, we utilized the properties of second-order cones to derive the cone decomposition method. The steep-ascent method in Chapter 4 is a heuristic method that outputs a solution quickly. However, this heuristic method does not guarantee the optimality of the output solution. The cone decomposition method is designed to obtain a solution that attains the optimal value in a prescribed tolerance.

6.1 Primary Properties Cone Decomposition Method

The proposed cone decomposition method is based on the following theorem:
Theorem 6.1.1. [49] *Let*

$$\hat{\boldsymbol{H}}^m := \left\{ (v_0, \boldsymbol{v}, \boldsymbol{w}) \in \mathbb{R}^{2m+1} : v_j^2 \leq w_j v_0 \ (j = 1, \ldots, m),\ \sum_{j=1}^{m} w_j \leq v_0,\ v_0 \geq 0 \right\},$$

then $\mathcal{K}^m = Proj_{(v_0, \boldsymbol{v})}(\hat{\boldsymbol{H}}^d)$*, where* $Proj_{(v_0, \boldsymbol{v})}$ *is the orthogonal projection onto the space of* $(v_0, \boldsymbol{v})$ *variables.*

Utilizing an auxiliary vector $\boldsymbol{w} \in \mathbb{R}^m$ in Theorem 6.1.1, $\mathcal{K}^m$ has another decomposition as:
Corollary 6.1.2. *A second-order cone* $\mathcal{K}^m$ *can be also written as*

$$\mathcal{K}^m := \left\{ (v_0, \boldsymbol{v}) \in \mathbb{R}^{m+1} : \exists \boldsymbol{w} \in \mathbb{R}^m \text{ such that } v_j^2 \leq w_j v_0 \ (j = 1, \ldots, m),\ \sum_{j=1}^{m} w_j \leq v_0,\ v_0 \geq 0 \right\}.$$

This new decomposition of $\mathcal{K}^m$ in the above corollary results in another reformulation of EDP (1.2).

$$\begin{array}{lll} \text{maximize} & : & \dfrac{\boldsymbol{g}^T \boldsymbol{y}}{N} \\ \text{subject to} & : & \boldsymbol{e}^T \boldsymbol{y} = N, \boldsymbol{z} = \boldsymbol{U}\boldsymbol{y}, \\ & & z_i^2 \leq w_i c_0 \ (i = 1, \ldots, m),\ \sum_{i=1}^{m} w_i \leq c_0,\ y_i \in \{0, 1\} \ (i = 1, \ldots, m) \end{array} \tag{6.1}$$

where z_i is the ith element of $\boldsymbol{z}$ and $c_0 = \sqrt{2\theta}N$. In this new formulation, the decision variables are $\boldsymbol{y}, \boldsymbol{z}$, and $\boldsymbol{w}$.

We should emphasize that , our nonlinear constraint in (6.1) is only the quadratic constraints $z_i^2 \leq w_i c_0$ with two variables z_i and w_i. In the proposed CDM, we generate cutting planes to these quadratic cones. Particularly, we use the geometric cuts as cutting planes, that is, we generate the cutting planes employing orthogonal projections [9].

The framework of the proposed CDM is given as Algorithm 6.1.3.

Algorithm 6.1.3. [Cone decomposition method (CDM)]

Step 1 Let P^0 be an MI-LP problem that is generated from an optimization problem (6.1) by omitting the quadratic constraints $z_i^2 \leq w_i c_0$ $(i = 1, \ldots, m)$. However, this constraint implicitly guarantees the nonnegativity of w_i. To make the MI-LP relaxation tighter, we explicitly add $w_i \geq 0$ $(i = 1, \ldots, m)$. Apply an MI-LP solver to P^0, and let its optimal solution be $(\hat{\boldsymbol{y}}^0, \hat{\boldsymbol{z}}^0, \hat{\boldsymbol{w}}^0)$. Let $k = 0$.

Step 2 Let a set of generated cuts $\mathscr{C}^k = \emptyset$.

Step 3 For each $i = 1, \ldots, m$, if $(\hat{z}_i^k)^2 \leq \hat{w}_i^k c_0$ is violated, apply the following steps.

Step 3-1 Compute the orthogonal projection of $(\hat{z}_i^k, \hat{w}_i^k)$ onto $z_i^2 \leq w_i c_0$ by solving the following sub-problem with the Lagrangian multiplier method;

$$\begin{array}{ll} \text{minimize} & : \frac{1}{2}\left(\bar{z} - \hat{z}_i^k\right)^2 + \frac{1}{2}\left(\bar{w} - \hat{w}_i^k\right)^2 \\ \text{subject to} & : \bar{z}^2 \leq \bar{w} c_0. \end{array}$$

Let $(\bar{z}_i^k, \bar{w}_i^k)$ be the solution of this subproblem.

Step 3-2 Add to $\mathscr{C}^k$ the following linear constraint

$$\begin{pmatrix} \hat{z}_i^k - \bar{z}_i^k \\ \hat{w}_i^k - \bar{w}_i^k \end{pmatrix}^T \begin{pmatrix} z_i - \bar{z}_i^k \\ w_i - \bar{w}_i^k \end{pmatrix} \leq 0.$$

Step 4 If $\mathscr{C}^k$ is empty, output $\hat{\boldsymbol{y}}^k$ as the solution and terminate.

Step 5 Build a new MI-LP P^{k+1} by adding $\mathscr{C}^k$ to P^k. Let the optimal solution of P^{k+1} be $\left(\hat{\boldsymbol{y}}^{k+1}, \hat{\boldsymbol{z}}^{k+1}, \hat{\boldsymbol{w}}^{k+1}\right)$. Return to **Step 2** with $k \leftarrow k + 1$.

In Step 3-1 of Algorithm 6.1.3, we compute the orthogonal projection. It would be desirable to compute the orthogonal projection on the original quadratic constraint $\boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x} \leq 2\theta$, but such orthogonal projection does not have an analytic form. Kiseliov [22] proposed some numerical method, but this is an iterative method. Another iterative method is also proposed by [16] to solve different case of second-order cones. In contrast, the orthogonal projection in Step 3-1 is onto a specific cone $\bar{z}^2 \leq \bar{w} c_0$. We can derive its analytical form, as proven in the next theorem. Note that the decomposition in Corollary 6.1.2 enables us to derive this theorem.

Theorem 6.1.4. *Assume that $(\hat{z}, \hat{w}) \in \mathbb{R}^2$ violates $\hat{z}^2 \leq \hat{w} c_0$ and $0 \leq \hat{w} \leq c_0$. Let $(\bar{z}, \bar{w}) \in \mathbb{R}^2$ be the orthogonal projection of $(\hat{z}, \hat{w})$ onto $z^2 \leq w c_0$. Then, $(\bar{z}, \bar{w})$ can be given by an analytical form.*

In the proof of Theorem 6.1.4, we make use of Cardano's Formula [50] to obtain a root of a cubic function analytically.

Theorem 6.1.5. [Cardano's Formula [10]] *Let $F(\lambda)$ be a cubic function $F(\lambda) = a\lambda^3 + b\lambda^2 + c\lambda + d$ with $a \neq 0$. Then $F(\lambda) = 0$ has three solutions*

$$\left\{ \begin{array}{lcl} \lambda_1 & = & S + T - \frac{b}{3a}, \\ \lambda_2 & = & -\frac{S+T}{2} - \frac{b}{3a} + \frac{\sqrt{-3}}{2}(S - T), \\ \lambda_3 & = & -\frac{S+T}{2} - \frac{b}{3a} - \frac{\sqrt{-3}}{2}(S - T), \end{array} \right.$$

where

$$S = \sqrt[3]{R + \sqrt{Q^3 + R^2}}, \quad T = \sqrt[3]{R - \sqrt{Q^3 + R^2}}, \quad Q = \frac{3ac - b^2}{9a^2}, \text{ and } R = \frac{9abc - 27a^2 d - 2b^3}{54a^3}.$$

Proof. (for Theorem 6.1.4)

The orthogonal projection $(\bar{z}, \bar{w}) \in \mathbb{R}^2$ is the optimal solution of the following subproblem.

$$\begin{array}{ll} \text{minimize} & : \frac{1}{2}(z-\hat{z})^2 + \frac{1}{2}(w-\hat{w})^2 \\ \text{subject to} & : z^2 \leq wc_0. \end{array} \tag{6.2}$$

This problem has a convex closed feasible region and its objective function is strongly convex, therefore, this problem has a unique solution. Since $(\hat{z}, \hat{w})$ is outside of the region $z^2 \leq wc_0$, the projection exists on the boundary of the region. We can replace $z^2 \leq wc_0$ with $z^2 = wc_0$, and (6.2) is equivalent to the following optimization problem:

$$\begin{array}{ll} \text{minimize} & : \frac{1}{2}(z-\hat{z})^2 + \frac{1}{2}(w-\hat{w})^2 \\ \text{subject to} & : z^2 = wc_0. \end{array} \tag{6.3}$$

To apply the Lagrangian multiplier method, a Lagrangian function of (6.3) with a Lagrangian multiplier $\lambda \in \mathbb{R}$ is given by:

$$\mathscr{L}(z, w, \lambda) = \frac{1}{2}(z-\hat{z})^2 + \frac{1}{2}(w-\hat{w})^2 - \lambda(wc_0 - z^2).$$

Setting $\nabla\mathscr{L} = 0$, we have

$$\nabla_z \mathscr{L} = z - \hat{z} + 2\lambda z = 0, \tag{6.4}$$

$$\nabla_w \mathscr{L} = w - \hat{w} - \lambda c_0 = 0, \tag{6.5}$$

$$\nabla_\lambda \mathscr{L} = -c_0 w + z^2 = 0. \tag{6.6}$$

Substituting (6.4) and (6.5) into (6.6) leads to a cubic function with respect to λ:

$$4c_0^2\lambda^3 + (4c_0^2 + 4c_0\hat{w})\lambda^2 + (c_0^2 + 4c_0\hat{w})\lambda + (c_0\hat{w} - \hat{z}^2) = 0. \tag{6.7}$$

Defining $a = 4c_0^2,\ b = 4c_0^2 + 4c_0\hat{w},\ c = c_0^2 + 4c_0\hat{w},$ and $d = c_0\hat{w} - \hat{z}^2$, we apply Theorem 6.1.5 to obtain λ. In Theorem 6.1.5, we have three solutions $\lambda_1, \lambda_2, \lambda_3$. Among the three solutions, only λ_1 can generate the analytical solution since the other two contain nonzero imaginary parts. To prove that λ_2 and λ_3 are complex numbers, it is enough to show $S \neq T$, and this is equivalent to show $Q^3 + R^2 \neq 0$. Computing

$$Q^3 + R^2 = \left(\frac{3ac - b^2}{9a^2}\right)^3 + \left(\frac{9abc - 27a^2d - 2b^3}{54a^3}\right)^2 = \frac{27a^2d^2 - 18abcd + 4ac^3 + 4b^3d - b^2c^2}{108a^4},$$

we can prove that $Q^3 + R^2 \neq 0$ by showing the numerator is nonzero. Substituting a, b, c and d, we derive

$$27a^2d^2 - 18abcd + 4ac^3 + 4b^3d - b^2c^2 = 32c_0^3\hat{z}^2(c_0 - 2\hat{w})^3 + 432c_0^4\hat{z}^4$$

where the right-hand side part can be transformed as follows:

$$\begin{aligned} 32c_0^3\hat{z}^2(c_0 - 2\hat{w})^3 + 432c_0^4\hat{z}^4 &= 16c_0^3\hat{z}^2(2c_0^3 - 12c_0^2\hat{w} + 24c_0\hat{w}^2 + 27c_0\hat{z}^2 - 16\hat{w}^3) \\ &> 16c_0^3\hat{z}^2(2c_0^3 - 12c_0^2\hat{w} + 24c_0\hat{w}^2 + 27c_0^2\hat{w} - 16\hat{w}^3) \\ &= 16c_0^3\hat{z}^2(2c_0^3 + 15c_0^2\hat{w} + 8c_0\hat{w}^2 + 16(c_0 - \hat{w})\hat{w}^2). \end{aligned}$$

Here, the inequality is due to $\hat{z}^2 > \hat{w}c_0$. When we solve an MILP for obtaining the $(\hat{z}, \hat{w})$, we add $w_i \geq 0$ $(i = 1, \ldots, m)$, therefore its solution satisfies $0 \leq \hat{w} \leq c_0$. Since $c_0 = \sqrt{2\theta N} \neq 0$, we know that the numerator is nonzero. Therefore, we proved that λ_2 and λ_3 have nonzero imaginary parts.

Thus, we only have the analytical solution by λ_1. After we obtain λ as λ_1, it is easy to compute z and w by (6.4) and (6.5). Therefore, the optimal solution $(\bar{z}, \bar{w})$ of (6.2) has an analytical form. □

Through the analytical form of the optimal solution $(\bar{z}, \bar{w})$, we can also derive an analytical form

$$\begin{pmatrix} \hat{z} - \bar{z} \\ \hat{w} - \bar{w} \end{pmatrix}^T \begin{pmatrix} z - \bar{z} \\ w - \bar{w} \end{pmatrix} \leq 0.$$

The substitution of $\bar{z} = \frac{\hat{z}}{1+2\lambda}$ and $\bar{w} = \lambda c_0 + \hat{w}$ results in an analytical form of the geometric cut as

$$\left(z - \frac{\hat{z}}{C_4 + 1}\right)\left(\hat{z} - \frac{\hat{z}}{C_4 + 1}\right) + \sqrt{\theta/2}NC_4\left(\hat{w} - w + \sqrt{\theta/2}NC_4\right) \leq 0,$$

where

$$C_1 = \frac{54\theta N\hat{z}^2 + \sqrt{2\theta}\left(\sqrt{2\theta}N - 2\hat{w}\right)^3}{864N^3\theta^2},\ C_2 = \left(\frac{\left(\sqrt{2\theta}N - 2\hat{w}\right)^2}{72N^2\theta}\right)^3,\ C_3 = \frac{2\theta N + \sqrt{2\theta}\hat{w}}{3\theta N},$$

and $C_4 = 2\left(C_1 - (C_1^2 - C_2)^{\frac{1}{2}}\right)^{\frac{1}{3}} + 2\left((C_1^2 - C_2)^{\frac{1}{2}} + C_1\right)^{\frac{1}{3}} - C_3$.

Note that the analytical solution of (6.2) can be also proved via solving a one-dimensional unconstrained optimization problem. Rewriting problem (6.2) as

$$\text{minimize } \tfrac{1}{2}(z - \hat{z})^2 + \tfrac{1}{2}\left(\tfrac{z^2}{c_0} - \hat{w}\right)^2$$

and utilizing Cardano's Formula, we can also obtain the optimal solution $(\bar{z}, \bar{w})$ of (6.2) in an analytical form. There, we again need a discussion that only one root of the corresponding cubic function is a real number.

The termination of the proposed method is guaranteed by the following theorem.

Theorem 6.1.6. *Algorithm 6.1.3 terminates in a finite number of iterations.*

Proof. The number of points we are interested for $\boldsymbol{y}$ is at most 2^m, where m is the number of candidate genotypes, due to the binary constraints $y_i \in \{0, 1\}$ $(i = 1, \ldots, m)$. In the kth iteration, the generated cuts in $\mathscr{C}^k$ remove $\hat{\boldsymbol{z}}^k, \hat{\boldsymbol{w}}^k$. Since $\hat{\boldsymbol{y}}^k$ is directly connected to $\hat{\boldsymbol{z}}^k$ by the constraint $\boldsymbol{z} = \boldsymbol{U}\boldsymbol{y}$ and $\boldsymbol{U}$ is invertible, $\hat{\boldsymbol{y}}^k$ is not feasible in P^{k+1}. At least one solution will be infeasible in each iteration, therefore, the number of iterations is also at most 2^m. □

6.2 Cone Decomposition Method with a Sparse Matrix

A sparsity structure in $\boldsymbol{A}^{-1}$ (the inverse of Wright's numerator relationship matrix) was the key property in [55] to reduce the computation time for solving UDP (1.1). We exploit this sparsity structure to improve the performance of CDM. In particular, we will discuss that MI-LP problems which need to be solved in CDM will have fewer nonzero elements by the sparsity, and this will lead to a shorter computation time.

The sparsity structure in $\boldsymbol{A}^{-1}$ can be derived from an efficient formula due to Henderson [20]. We can decompose the positive definite matrix $\boldsymbol{A}^{-1}$ into $\boldsymbol{A}^{-1} = \boldsymbol{B}^T\boldsymbol{B}$, where $\boldsymbol{B} \in \mathbb{R}^{m\times m}$ is the matrix defined in [55]. If we use $\boldsymbol{b}_i^T$ to denote the ith row of $\boldsymbol{B}$, $\boldsymbol{b}_i^T$ is given by

$$\boldsymbol{b}_i^T = \begin{cases} \sqrt{\alpha_i}\boldsymbol{e}_i^T & \text{for} \quad i \in \mathscr{P}_0, \\ \sqrt{\alpha_i}\left(\boldsymbol{e}_i - \frac{1}{2}\boldsymbol{e}_{p(i)}\right)^T & \text{for} \quad i \in \mathscr{P}_1, \\ \sqrt{\alpha_i}\left(\boldsymbol{e}_i - \frac{1}{2}\boldsymbol{e}_{p(i)} - \frac{1}{2}\boldsymbol{e}_{q(i)}\right)^T & \text{for} \quad i \in \mathscr{P}_2. \end{cases} \tag{6.8}$$

Here, $\alpha_i \in \mathbb{R}$ $(i = 1, \ldots, m)$ is a constant computed using Quaas's algorithm [37], and $\boldsymbol{e}_i \in \mathbb{R}^m$ is the ith unit vector. From the actual value of $\alpha_i \in \mathbb{R}$ $(i = 1, \ldots, m)$ obtained by Quaas's algorithm, it holds that $\boldsymbol{b}_i^T \boldsymbol{b}_i \leq 2$. As described in Section 2.1.4, the set of genotypes $\{1, \ldots, m\}$ is classified by the pedigree information into the disjoint three sets $\mathcal{P}_0$ (no parents are known), $\mathcal{P}_1$ (only one parent $p(i)$ is known), and $\mathcal{P}_2$ (both parents $p(i)$ and $q(i)$ are known).

Matrix $\boldsymbol{U}$ Matrix $\boldsymbol{B}$

Figure 6.1: Non-zero element structure of matrix $\boldsymbol{U}$ and $\boldsymbol{B}$ for $m = 10100$

The important property in (6.8) is that the number of the nonzero elements in each $\boldsymbol{b}_i$ is at most three. For the case $\mathcal{P}_2$, the nonzero elements appear at only the ith, $p(i)$th and $q(i)$th positions.

Figure 6.1 illustrates the structure of nonzero elements in the matrices $\boldsymbol{U}$ and $\boldsymbol{B}$ for an EDP of size $m = 10100$. The matrix $\boldsymbol{U}$ contains 23137740 nonzero elements, while $\boldsymbol{B}$ has only 30100 elements, thus we observe that matrix $\boldsymbol{B}$ has nonzero elements almost 769 times fewer than $\boldsymbol{U}$.

We exploit the sparsity in the framework of CDM. Let us recall that the original quadratic constraint in (1.3) is equivalent to a quadratic constraint $\boldsymbol{y}^T \boldsymbol{A}\boldsymbol{y} \leq 2\theta N^2$. We then introduce a new variable $\boldsymbol{v} \in \mathbb{R}^m$ by a relation $\boldsymbol{v} := \boldsymbol{A}\boldsymbol{y}$; thus the quadratic constraint can be replaced by $\boldsymbol{v}^T \boldsymbol{A}^{-1} \boldsymbol{v} \leq 2\theta N^2$. Since $\boldsymbol{A}^{-1} = \boldsymbol{B}^T \boldsymbol{B}$ and $\boldsymbol{b}_i^T$ is the ith row of $\boldsymbol{B}$, the quadratic constraint $\boldsymbol{v}^T \boldsymbol{A}^{-1} \boldsymbol{v} \leq 2\theta N^2$ is further equivalent to

$$(\sqrt{2\theta}N, (\boldsymbol{b}_1^T \boldsymbol{v}, \boldsymbol{b}_2^T \boldsymbol{v}, \ldots, \boldsymbol{b}_m^T \boldsymbol{v})^T) \in \mathcal{K}^m. \tag{6.9}$$

Thus, using Corollary 6.1.2 and $c_0 = \sqrt{2\theta}N$, (6.9) can be decomposed into

$$\sum_{i=1}^{m} w_i \leq c_0 \text{ and } (\boldsymbol{b}_i^T \boldsymbol{v})^2 \leq w_i c_0 \ (i = 1, \ldots, m). \tag{6.10}$$

If $(\hat{\boldsymbol{v}}, \hat{w}_i) \in \mathbb{R}^m \times \mathbb{R}_+$ violates the inequality $(\boldsymbol{b}_i^T \boldsymbol{v})^2 \leq w_i c_0$, we can again consider a geometric cut, but this time we utilize the orthogonal projection of $(\hat{\boldsymbol{v}}, \hat{w}_i)$ onto $(\boldsymbol{b}_i^T \boldsymbol{v})^2 \leq w_i c_0$. We use $d_i^w w_i + (\boldsymbol{d}_i^v)^T \boldsymbol{v} + d_i^0 \leq 0$ to denote the geometric cut as a cutting plane that separates the point $(\hat{\boldsymbol{v}}, \hat{w}_i)$ from the cone $(\boldsymbol{b}_i^T \boldsymbol{v})^2 \leq w_i c_0$. This cutting plane has an advantage for computation efficiency as described in the next theorem.

Theorem 6.2.1. *Assume that $(\hat{\boldsymbol{v}}, \hat{w})$ violates $(\boldsymbol{b}^T \boldsymbol{v})^2 \leq w c_0$. Then, the geometric cut $d^w w + (\boldsymbol{d}^v)^T \boldsymbol{v} + d^0 \leq 0$ by the orthogonal projection onto $(\boldsymbol{b}^T \boldsymbol{v})^2 \leq w c_0$ can be given by an analytical form. In particular, the number of the nonzero elements in $\boldsymbol{d}^v$ is at most that of $\boldsymbol{b}$.*

Since the number of the nonzero elements in each $\boldsymbol{b}_i$ is at most three, this geometric cut is a sparse linear constraint. This property is remarkably effective when we solve MI-LP problems iteratively. As already mention in Chapter 1, the cutting plane used in `dsOpt` [30] is derived from the first-order Taylor series. For an iterate $\boldsymbol{y}^k$ that violates the quadratic constraint $\boldsymbol{y}^T \boldsymbol{A}\boldsymbol{y} \leq 2\theta N^2$, the

cut in `dsOpt` is of form $(\boldsymbol{A}\boldsymbol{y}^k)^T\boldsymbol{y} \leq \sqrt{2\theta N^2}\sqrt{(\boldsymbol{y}^k)^T\boldsymbol{A}\boldsymbol{y}^k}$. Since this cut involves the dense matrix $\boldsymbol{A}$, it usually involves m nonzero elements, thus it is much denser than the geometric cut generated from Theorem 6.2.1.

Proof. In a similar way to Theorem 6.1.4, let $(\bar{\boldsymbol{v}}, \bar{w})$ be the orthogonal projection of $(\hat{\boldsymbol{v}}, \hat{w})$ onto $(\boldsymbol{b}^T\boldsymbol{v})^2 \leq wc_0$. Then, $(\bar{\boldsymbol{v}}, \bar{w})$ can be obtained by the following sub-problem:

$$\begin{array}{ll} \text{minimize} & : \frac{1}{2}||\boldsymbol{v} - \hat{\boldsymbol{v}}||^2 + \frac{1}{2}(w - \hat{w})^2 \\ \text{subject to} & : (\boldsymbol{b}^T\boldsymbol{v})^2 = wc_0. \end{array} \tag{6.11}$$

A Lagrangian function is defined with a Lagrangian multiplier $\lambda \in \mathbb{R}$:

$$\mathscr{L}(\boldsymbol{v}, w, \lambda) = \frac{1}{2}||\boldsymbol{v} - \hat{\boldsymbol{v}}||^2 + \frac{1}{2}(w - \hat{w})^2 - \lambda(wc_0 - (\boldsymbol{b}^T\boldsymbol{v})^2).$$

By setting the differentials to zero, we obtain

$$\boldsymbol{v} - \hat{\boldsymbol{v}} + 2\lambda\boldsymbol{b}\boldsymbol{b}^T\boldsymbol{v} = \boldsymbol{0}, \quad w - \hat{w} = \lambda c_0, \quad wc_0 = (\boldsymbol{b}^T\boldsymbol{v})^2.$$

A distinguished difference from Theorem 6.1.4 is an application of the Sherman-Morrison-Woodbury formula [42]. The use of this formula leads to

$$\boldsymbol{v} = (\boldsymbol{I} + 2\lambda\boldsymbol{b}\boldsymbol{b}^T)^{-1}\hat{\boldsymbol{v}} = \hat{\boldsymbol{v}} - \frac{2\lambda(\boldsymbol{b}^T\hat{\boldsymbol{v}})\boldsymbol{b}}{1 + 2\lambda\boldsymbol{b}^T\boldsymbol{b}}. \tag{6.12}$$

The substitution of $w = \hat{w} + \lambda c_0$ and (6.12) into $wc_0 = (\boldsymbol{b}^T\boldsymbol{v})^2$ results in the following cubic function

$$(4c_0^2(\boldsymbol{b}^T\boldsymbol{b})^2)\lambda^3 + (4c_0^2\boldsymbol{b}^T\boldsymbol{b} + 4\hat{w}c_0(\boldsymbol{b}^T\boldsymbol{b})^2)\lambda^2 + (c_0^2 + 4\hat{w}c_0(\boldsymbol{b}^T\boldsymbol{b}))\lambda + (\hat{w}c_0 - (\boldsymbol{b}^T\hat{\boldsymbol{v}})^2) = 0. \tag{6.13}$$

We can use a similar step to Theorem 6.1.4 to show that this cubic function again has only one real root, which we will denote by $\bar{\lambda}$. In particular, if we can show $Q^3 + R^2 \neq 0$, Theorem 6.1.5 guarantees an analytical form on $\bar{\lambda}$. We put the coefficients of (6.13) to derive the positivity of $Q^3 + R^2$:

$$\begin{aligned} & \text{the numerator of } \ Q^3 + R^2 \\ = \ & 16(\boldsymbol{b}^T\boldsymbol{b})^3(\boldsymbol{b}^T\hat{\boldsymbol{v}})^2c_0^3\left(-16(\boldsymbol{b}^T\boldsymbol{b})^3\hat{w}^3 + 24(\boldsymbol{b}^T\boldsymbol{b})^2c_0\hat{w}^2 + 27(\boldsymbol{b}^T\boldsymbol{b})(\boldsymbol{b}^T\hat{\boldsymbol{v}})^2c_0 - 12(\boldsymbol{b}^T\boldsymbol{b})c_0^2\hat{w} + 2c_0^3\right) \\ = \ & 16(\boldsymbol{b}^T\boldsymbol{b})^3(\boldsymbol{b}^T\hat{\boldsymbol{v}})^2c_0^3\left(2\left(c_0 - 2(\boldsymbol{b}^T\boldsymbol{b})\hat{w}\right)^3 + 27(\boldsymbol{b}^T\boldsymbol{b})(\boldsymbol{b}^T\hat{\boldsymbol{v}})^2c_0\right) \\ > \ & 16(\boldsymbol{b}^T\boldsymbol{b})^3(\boldsymbol{b}^T\hat{\boldsymbol{v}})^2c_0^3\left(2\left(c_0 - 2(\boldsymbol{b}^T\boldsymbol{b})\hat{w}\right)^3 + 27(\boldsymbol{b}^T\boldsymbol{b})\hat{w}c_0^2\right) \\ = \ & 16(\boldsymbol{b}^T\boldsymbol{b})^3(\boldsymbol{b}^T\hat{\boldsymbol{v}})^2c_0^3\left(4(\boldsymbol{b}^T\boldsymbol{b})\hat{w} + c_0\right)^2\left(2c_0 - (\boldsymbol{b}^T\boldsymbol{b})\hat{w}\right) \\ \geq \ & 0. \end{aligned}$$

The first inequality comes from $(\boldsymbol{b}^T\hat{\boldsymbol{v}})^2 > \hat{w}c_0 \geq 0$ and the last inequality from $\boldsymbol{b}^T\boldsymbol{b} \leq 2$, which is guaranteed by the actual value of α from Quaas's algorithm, and $0 \leq \hat{w} \leq c_0$.

Using $\bar{\lambda}$, we obtain $\bar{\boldsymbol{v}} = \hat{\boldsymbol{v}} - \frac{2\bar{\lambda}(\boldsymbol{b}^T\hat{\boldsymbol{v}})}{1+2\bar{\lambda}\boldsymbol{b}^T\boldsymbol{b}}\boldsymbol{b}$ and $\bar{w} = \hat{w} + \bar{\lambda}c_0$. The geometric cut between $(\hat{\boldsymbol{v}}, \hat{w})$ and $(\boldsymbol{b}^T\boldsymbol{v})^2 \leq wc_0$ at $(\bar{\boldsymbol{v}}, \bar{w})$ can be given by

$$\begin{pmatrix} \hat{\boldsymbol{v}} - \bar{\boldsymbol{v}} \\ \hat{w} - \bar{w} \end{pmatrix}^T \begin{pmatrix} \boldsymbol{v} - \bar{\boldsymbol{v}} \\ w - \bar{w} \end{pmatrix} \leq 0,$$

therefore, this cut can be rewritten as $d^w w + (\boldsymbol{d}^v)^T \boldsymbol{v} + d^0 \leq 0$ with

$$\begin{aligned} d^w &= -\bar{\lambda} c_0, \quad \boldsymbol{d}^v = \frac{2\bar{\lambda}(\boldsymbol{b}^T \hat{\boldsymbol{v}})\boldsymbol{b}}{1 + 2\bar{\lambda}\boldsymbol{b}^T}\boldsymbol{b}, \\ d^0 &= -\left(\frac{2\bar{\lambda}(\boldsymbol{b}^T \hat{\boldsymbol{v}})\boldsymbol{b}}{1 + 2\bar{\lambda}\boldsymbol{b}^T\boldsymbol{b}}\right)^T \left(\hat{\boldsymbol{v}} - \frac{2\bar{\lambda}(\boldsymbol{b}^T \hat{\boldsymbol{v}})\boldsymbol{b}}{1 + 2\bar{\lambda}\boldsymbol{b}^T\boldsymbol{b}}\right) + \bar{\lambda} c_0 (\hat{w} + \bar{\lambda} c_0). \end{aligned}$$

Consequently, we obtain an analytical form for the geometric cut. In addition, since $\frac{2\bar{\lambda}(\boldsymbol{b}^T \hat{\boldsymbol{v}})}{1+2\bar{\lambda}\boldsymbol{b}^T\boldsymbol{b}} \in \mathbb{R}$, the number of nonzero elements in $\boldsymbol{d}^v$ is at most that of $\boldsymbol{b}$. □

With the geometric cut for the cone $(\boldsymbol{b}_i^T \boldsymbol{v})^2 \leq w_i c_0$, we develop Algorithm 6.2.2.

Algorithm 6.2.2. [Cone decomposition method with the sparsity in the inverse of the numerator relationship matrix (CDM-B)]

Step 1 Let P^0 be an MI-LP problem that is generated from an optimization problem by omitting the quadratic constraints $(\boldsymbol{b}_i^T \boldsymbol{v})^2 \leq w_i c_0$ $(i = 1, \ldots, m)$. Apply an MI-LP solver to P^0, and let its optimal solution be $(\hat{\boldsymbol{y}}^0, \hat{\boldsymbol{v}}^0, \hat{\boldsymbol{w}}^0)$. Let $k = 0$.

Step 2 Let a set of generated cuts $\mathscr{C}^k = \emptyset$.

Step 3 For each $i = 1, \ldots, m$, if $(\boldsymbol{b}_i^T \hat{\boldsymbol{v}}^k)^2 \leq \hat{w}_i^k c_0$ is violated, add the geometric cut obtained by Theorem 6.2.1 to $\mathscr{C}^k$.

Step 4 If $\mathscr{C}^k$ is empty, output $\hat{\boldsymbol{y}}^k$ as the solution and terminate.

Step 5 Build a new MI-LP P^{k+1} by adding $\mathscr{C}^k$ to P^k. Let the optimal solution of P^{k+1} be $\left(\hat{\boldsymbol{y}}^{k+1}, \hat{\boldsymbol{v}}^{k+1}, \hat{\boldsymbol{w}}^{k+1}\right)$. Return to **Step 2** with $k \leftarrow k + 1$.

Since Algorithm 6.2.2 is a variant of CDM that exploits the sparsity of the matrix $\boldsymbol{B}$, it will be referred as CDM-B. This algorithm also terminates in a finite number of iterations.

Theorem 6.2.3. *Algorithm 6.2.2 terminates in a finite number of iterations.*

Proof. In a similar way to Algorithm 3.3, we again rely on the upper bound 2^m for the number of points for feasible $\boldsymbol{y}$. In each iteration of Algorithm 4.2, we remove $\hat{\boldsymbol{v}}^k$ and $\hat{\boldsymbol{w}}^k$. Therefore, we employ the constraint $y_i = \boldsymbol{b}_i^T \boldsymbol{v}$ for $i = 1, \ldots, m$ to show that $\hat{\boldsymbol{y}}^k$ such that $\hat{y}_i^k = \boldsymbol{b}_i^T \hat{\boldsymbol{v}}^k$ is removed in the kth iteration. Therefore, the number of iterations in Algorithm 4.2 is at most 2^m. □

6.3 Numerical Results

This section provides numerical results of our proposed approaches (CDM, and CDM-B) comparing to existing methods. Both proposed approaches were implemented using `Matlab R2017b` with `CPLEX` as the MI-LP solver. Similarly to the previous chapters, we compared the proposed approaches with `GENCONT`, `dsOpt` (implemented in `OPSEL`, and `CPLEX`. The numerical experiment in this chapter was done using a 64-bit Windows 10 PC with Xeon CPU E3-1231 (3.40 GHz) and 8 GB memory space. The data source and most parameter setting are same.

We begin with the `GENCONT` results presented in Table 6.1. Remind that $\boldsymbol{x}$ is a feasible solution if the constraint $\boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x} \leq 2\theta$ is satisfied and the selected candidate is equal to N. The result shows different performance compared to that in the previous chapter due to an instability of the execution of GENCONT (in particular, we used different CPUs). Here, even the result satisfies

Table 6.1: Numerical results on GENCONT

m	$N = 50$					$N = 100$				
	2θ	$\boldsymbol{g}^T\boldsymbol{x}$	$\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$	time (sec)	$\#N_s$	2θ	$\boldsymbol{g}^T\boldsymbol{x}$	$\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$	time (sec)	$\#N_s$
200	0.0334	11.472	0.03340	3.54	64	0.0258	8.89	0.02580	0.48	93
1050	0.0627	25.91	0.06270	7.20	81	0.0539	24.07	0.05390	4.77	94
2045	0.0711	438.36	0.07109	111.52	71	0.0628	432.75	0.06279	106.48	74
5050	0.1081	43.44	0.10810	1561.43	78	0.0994	42.08	0.09940	1533.31	81

$\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} \leq 2\theta$ for all m, GENCONT failed to output feasible solution since the number of selected candidates $\#N_s$ does not match N.

Unlike GENCONT, other results provided in Tables 6.2 and 6.3 match $N = 50$ and $N = 100$, respectively. Both tables show the comparison of our proposed method with the rest of existing methods.

Through the first column, we could see all methods for this numerical experiment. Here, we have different settings for direct implementation using CPLEX. CPLEX-default is used to solve (1.3) using the default setting; and CPLEX-LP is used for solving (6.1) by setting mip.strategy.search=2 in CPLEX to use LP relaxation forcibly. For computation time as in columns six and nine, we limited it into 3 hours. Therefore, if the computation is not finished within 3 hours, we indicated by "$>$ 3 hours" and the best objective values up to the point are shown in the tables. Furthermore, we noted "OOM" for the result cannot be obtained due to out of memory.

The results show in both tables that only dsOpt failed to handle the largest problem $m = 15222$ due to insufficient memory. dsOpt also required longer computation time, even for the smallest problem, compared to other existing methods.

CPLEX-default gives better computation time performance for $m \leq 2045$ except with gap $= 1\%$. However, as the size is getting larger and the gap is tighter, this implementation failed to complete the computation within 3 hours. In the case $m = 10100$ and gap $= 5\%$, while CPLEX-LP and dsOpt obtained the objective values $\boldsymbol{g}^T\boldsymbol{x} = 45.91$ and 46.00, respectively, CPLEX-default only reached 44.89.

Through the existing methods, CPLEX-LP gives better performance by solving the problem within the time limit. This implementation is even more efficient than our CDM excluding the largest size. Hence, from the difference between the CPLEX-LP and CPLEX-default results, the default setting of CPLEX cannot solve EDPs efficiently. We have to explicitly let CPLEX know that LP relaxation is effective for EDPs.

According to Tables 6.2 and 6.3, CDM-B gives better performance among all methods including CDM itself. This method even reduce the computation time into $\frac{1}{40}$ times that of CDM when $m = 10100$ and gap $= 1\%$. In addition, CDM is the most effective for both gap $= 5\%$ and 1%.

In addition, Table 6.4 presents an evidence of CDM and CDM-B efficiency. The column "nnz" shows the number of nonzero elements in MI-LP problems solved at the last iteration of both methods. The table verifies that nnz of CDM-B is smaller than that of CDM, and that leads to reductions in both memory size and computation time. For example, with $N = 50, m = 10100$, and gap $= 5\%$, CDM generates 164574 nnz but CDM generates 34570 nnz, thus CDM is 5 times lesser than CDM. Therefore, it is reflected in Table 6.2 and 6.3 that the computation time of CDM-B is much faster than CDM.

The last table compares the result of all proposed approaches. From Section 4, we get that SA (SOCP) is the fastest relaxation among the SA approaches. Therefore, in Table 6.5, we only put the result of SA (SOCP), and compare it with the other approaches. The result in this table shows that the computation time of SA (SOCP) is very competitive compared to the CDM-B.

Table 6.2: Numerical comparison for EDPs ($N = 50$)

Method	m	2θ	gap = 5%			gap = 1%		
			$\boldsymbol{g}^T\boldsymbol{x}$	$\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$	time (sec)	$\boldsymbol{g}^T\boldsymbol{x}$	$\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$	time (sec)
CPLEX-def	200	0.0334	24.99	0.03340	1.06	25.19	0.03340	8735.24
CPLEX-LP			25.16	0.03340	1.96	25.19	0.03340	3.96
dsOpt			25.12	0.03340	5.32	25.18	0.03340	606.94
CDM			25.02	0.03340	1.69	25.15	0.03340	1.72
CDM-B			25.02	0.03340	1.64	25.04	0.03340	1.84
CPLEX-def	1050	0.0627	24.97	0.06267	3.56	24.97	0.06267	6.64
CPLEX-LP			24.94	0.06265	4.27	24.94	0.06265	4.64
dsOpt			24.97	0.06169	5.19	24.85	0.06268	$>$ 3 hours
CDM			24.65	0.06118	9.41	24.96	0.06238	12.11
CDM-B			24.66	0.06138	2.98	24.95	0.06264	4.98
CPLEX-def	2045	0.0711	437.21	0.07100	3.95	437.21	0.07100	3.83
CPLEX-LP			438.07	0.07060	2.97	438.08	0.07060	3.52
dsOpt			432.94	0.06700	7.09	435.87	0.07020	14.42
CDM			434.26	0.06760	1.76	437.38	0.06960	2.52
CDM-B			432.59	0.06640	1.68	436.16	0.07060	1.86
CPLEX-def	5050	0.1081	41.90	0.10776	73.16	42.57	0.10781	$>$ 3 hours
CPLEX-LP			42.46	0.10658	11.42	42.46	0.10658	15.19
dsOpt			41.57	0.10471	236.70	42.67	0.10807	$>$ 3 hours
CDM			42.56	0.10742	187.24	42.56	0.10742	182.10
CDM-B			42.04	0.10507	5.37	42.54	0.10719	6.36
CPLEX-def	10100	0.0701	44.89	0.06931	$>$ 3 hours	44.89	0.06931	$>$ 3 hours
CPLEX-LP			45.91	0.06789	104.44	46.48	0.07008	200.55
dsOpt			46.00	0.07005	4509.83	46.21	0.06975	8787.37
CDM			45.27	0.06896	1003.67	46.43	0.07005	1204.47
CDM-B			45.88	0.06939	17.93	46.54	0.06989	28.93
CPLEX-def	15222	0.0388	118.33	0.03840	$>$ 3 hours	107.56	0.03280	$>$ 3 hours
CPLEX-LP			454.07	0.03860	350.14	458.85	0.03880	1080.17
dsOpt					OOM			OOM
CDM			452.57	0.03880	450.84	461.83	0.03880	547.02
CDM-B			449.01	0.03860	50.66	461.83	0.03880	112.58

Though SA (SOCP) is the fastest approach, SA (SOCP) cannot guarantee the optimal solution since it is only a heuristic method. Thus, excluding the SA (SOCP), CDM-B can be referred as the efficient approach for solving EDP.

Table 6.3: Numerical comparison for EDPs ($N = 100$)

Method	m	2θ	gap = 5%			gap = 1%		
			$\boldsymbol{g}^T\boldsymbol{x}$	$\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$	time (sec)	$\boldsymbol{g}^T\boldsymbol{x}$	$\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$	time (sec)
CPLEX-def	200	0.0258	23.19	0.02580	4.31	23.49	0.02580	13.14
CPLEX-LP			23.52	0.02580	3.08	23.52	0.02580	3.54
dsOpt			23.14	0.02575	1.30	23.54	0.02580	566.89
CDM			23.53	0.02580	1.78	23.55	0.02580	2.03
CDM-B			23.53	0.02580	1.57	23.52	0.02580	1.73
CPLEX-def	1050	0.0539	22.53	0.05389	6.68	22.53	0.05389	3.64
CPLEX-LP			22.55	0.05371	8.10	22.55	0.05371	5.61
dsOpt			21.79	0.05358	6.07	22.25	0.05382	193.08
CDM			22.49	0.05339	17.02	22.49	0.05339	15.23
CDM-B			22.49	0.05369	3.12	22.49	0.05369	3.19
CPLEX-def	2045	0.0628	420.04	0.06100	3.21	420.04	0.06100	3.08
CPLEX-LP			420.79	0.06190	4.28	420.79	0.06190	3.08
dsOpt			419.53	0.06155	7.93	419.53	0.06155	7.96
CDM			418.67	0.06010	2.56	418.67	0.06010	2.43
CDM-B			420.06	0.06165	2.43	420.06	0.06165	2.41
CPLEX-def	5050	0.0994	40.63	0.09932	58.37	40.63	0.09932	54.43
CPLEX-LP			40.56	0.09868	27.22	40.56	0.09868	19.23
dsOpt			40.13	0.09860	134.55	40.47	0.09936	367.29
CDM			40.28	0.09821	183.56	40.35	0.09742	197.38
CDM-B			40.02	0.09639	6.40	40.67	0.09943	7.87
CPLEX-def	10100	0.0610	43.79	0.06059	2720.18	44.34	0.06070	$>$ 3 hours
CPLEX-LP			44.43	0.06061	197.51	44.42	0.06061	216.66
dsOpt			43.36	0.06018	584.77	44.44	0.06100	7538.99
CDM			43.86	0.06095	948.07	44.53	0.06092	1282.72
CDM-B			44.26	0.05994	28.24	44.47	0.06081	29.23
CPLEX-def	15222	0.0300	436.92	0.02990	5084.69	436.92	0.02990	$>$ 3 hours
CPLEX-LP			423.75	0.02985	603.78	438.96	0.03000	710.21
dsOpt					OOM			OOM
CDM			432.13	0.02865	632.34	439.88	0.02960	448.82
CDM-B			433.19	0.02865	27.59	439.97	0.02950	47.51

Table 6.4: The number of nonzero elements in MI-LP problems arising from CDM and CDM-B

N	m	gap = 5%		gap = 1%	
		nnz		nnz	
		CDM	CDM-B	CDM	CDM-B
50	200	1700	1700	1702	1688
	1050	21290	7633	21460	8035
	2045	5960	5953	6035	6006
	5050	74601	17539	74601	17778
	10100	164574	34570	164948	35110
	15222	51123	41985	51397	41431
100	200	1842	1906	1864	1824
	1050	21948	8084	21948	8084
	2045	6820	6656	6820	6656
	5050	76357	18619	76391	18619
	10100	165850	35883	166357	35842
	15222	51719	41290	51917	42642

Table 6.5: Numerical comparison for all proposed approaches ($N = 50$, gap $= 5\%$)

Method	m	2θ	$2\theta(1+\epsilon)$	$\boldsymbol{g}^T\boldsymbol{x}$	$\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$	time (sec)
CPLEX-LP			*	25.16	0.03340	1.96
SA (SOCP)			*	25.09	0.03340	2.65
LPP-ACSM	200	0.0334	0.03373	24.84	0.03340	47.75
CDM			*	25.02	0.03340	1.69
CDM-B			*	25.02	0.03340	1.64
CPLEX-LP			*	24.94	0.06265	4.27
SA (SOCP)			*	24.83	0.06270	0.09
LPP-ACSM	1050	0.0627	0.06333	24.72	0.06215	1230.01
CDM			*	24.65	0.06118	9.41
CDM-B			*	24.66	0.06138	2.98
CPLEX-LP			*	438.07	0.07060	2.97
SA (SOCP)			*	438.39	0.07100	0.09
LPP-ACSM	2045	0.0711	0.07181			OOM
CDM			*	434.26	0.06760	1.76
CDM-B			*	432.59	0.06640	1.68
CPLEX-LP			*	42.46	0.10658	11.42
SA (SOCP)			*	42.69	0.10802	0.61
LPP-ACSM	5050	0.1081	0.109184			OOM
CDM			*	42.56	0.10742	187.24
CDM-B			*	42.04	0.10507	5.37
CPLEX-LP			*	45.91	0.06789	104.44
SA (SOCP)			*	46.57	0.07009	1.39
LPP-ACSM	10100	0.0701	0.070803			OOM
CDM			*	45.27	0.06896	1003.67
CDM-B			*	45.88	0.06939	17.93
CPLEX-LP			*	454.07	0.03860	350.14
SA (SOCP)			*	460.77	0.03880	12.46
LPP-ACSM	15222	0.0388	0.039189			OOM
CDM			*	452.57	0.03880	450.84
CDM-B			*	449.01	0.03860	50.66

7 Conclusion

7.1 Summary of the project

In this thesis, we studied efficient numerical methods for solving equally deployment problem (EDP) and MI-SOCP problem arising from tree breeding.

In Chapter 3, we discussed conic relaxation approaches based on LP, SOCP, and SDP. Among three relaxations, the quality of solutions generated through LP relaxation was not desirable, since the quadratic constraint on the genetic diversity was violated. In contrast, SOCP and SDP relaxations generate solutions that satisfy the quadratic constraint. We also showed that the objective value of SDP relaxation is slightly closer to the original EDP than that of SOCP relaxation, but the two objective values are competitive. In the viewpoint of computation time, SOCP relaxation is more efficient than SDP relaxation. In addition, we found from the numerical result that SOCP relaxation is more numerically stable.

In Chapter 4, we developed the steepest-ascent method to acquire a suitable solution for practical usage. We moved the quadratic constraint to the objective function as a penalty term, and we proved that this modification does not change the optimality of the solution if the penalty term is large enough. The numerical results show that SOCP relaxation used as the starting point of the steepest-ascent method was the most effective among the three conic relaxation approaches, and that this outperformed existing solvers, in particular, in the viewpoint of computation time.

In Chapter 5, we examined lifted-polyhedral programming (LPP) approach for obtaining the exact solution and proposed the modification of LPP by an active constraint selection method. We expected that such kind of modification would give better improvement by reducing the linear constraints embedded in LPP. However, the numerical experiments with very tight epsilon, as a parameter in LPP, indicates that both implementations are able to generate the solution for only small-scale problems. This kind of polyhedral relaxation demands a huge number of constraints that leads to a heavy computation.

In Chapter 6, we proposed cone decomposition methods (CDMs) to obtain a solution guaranteeing the optimality. We decomposed an m-dimensional second-order cones into m 3-dimensional second-order cones, and generated a geometric cut for each 3-dimensional second-order cone. We proved that this new geometric cut has an analytical form, thus it does not require iterative methods. We further enhanced the performance of the CDM by exploiting the sparsity of the Wright's numerator relationship matrix. We showed that the sparsity of the geometric cuts in the CDM successfully inherits that of the inverse the numerator matrix. Through the comparison among all methods, the proposed CDM with the use of sparsity efficiently obtained the optimal solution of EDPs, while other methods encountered insufficient memory space or time limitation.

7.2 Future Work

We suggest for future studies to consider sparse CDM combining with heuristic methods, for example, our proposed steepest-ascent method. In particular, we expect a good tentative value

would reduce the number of subproblems in the branch-and-bound step. Another direction is to consider another problem of OCS that involves not only simple binary constraints but also other types of integer constraints.

Bibliography

[1] J. Ahlinder, T. J. Mullin, and M. Yamashita. Using semidefinite programming to optimize unequal deployment of genotypes to a clonal seed orchard. *Tree Genetics & Genomes*, 10(1):27–34, 2014.

[2] A. A. Ahmadi and A. Majumbar. DSOS and SDSOS optimization: LP and SOCP-based alternatives to sum of squares optimization. In *Proceedings of the 48th Annual Conference on Information Sciences and Systems*, pages 1–5, 2014.

[3] F. Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM journal on Optimization*, 5(1):13–51, 1995.

[4] F. Alizadeh and D. Goldfarb. Second-order cone programming. *Mathematical programming*, 95(1):3–51, 2003.

[5] P. Belotti, J. C. Góez, I. Pólik, T. K. Ralphs, and T. Terlaky. A conic representation of the convex hull of disjunctive sets and conic cuts for integer second order cone optimization. In *Numerical Analysis and Optimization*, pages 1–35. Springer, 2015.

[6] A. Ben-Tal and A. Nemirovski. On polyhedral approximations of the second-order cone. *Mathematics of Operations Research*, 26(2):193–205, 2001.

[7] H. Y. Benson and Ü. Sa*u*glam. Mixed-integer second-order cone programming: A survey. In *Theory Driven by Influential Applications*, pages 13–36. INFORMS, 2013.

[8] D. P. Bertsekas. *Nonlinear programming*. Athena scientific, 1999.

[9] P. Białon. Some variants of projection methods for large nonlinear optimization problems. *Journal of Telecommunications and Information Technology*, pages 43–49, 2003.

[10] G. Cardano. *Ars magna or the rules of algebra*. Dover Publications, 1968.

[11] M. T. Cezik and G. Iyengar. Cuts for mixed 0-1 conic programming. *Mathematical Programming*, 104(1):179–202, 2005.

[12] C. C. Cockerham. Group inbreeding and coancestry. *Genetics*, 56(1):89, 1967.

[13] A. Domahidi, E. Chu, and S. Boyd. ECOS: An SOCP solver for embedded systems. In *Proceedings of European Control Conference*, pages 3071–3076, 2013.

[14] S. Drewes and S. Ulbrich. Subgradient based outer approximation for mixed integer second order cone programming. In *Mixed Integer Nonlinear Programming*, pages 41–59. Springer, 2012.

[15] M. A. Duran and I. E. Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical programming*, 36(3):307–339, 1986.

[16] O. Ferreira and S. Németh. How to project onto extended second order cones. *Journal of Global Optimization*, 70(4):707–718, 2018.

[17] B. Gärtner and J. Matousek. *Approximation algorithms and semidefinite programming*. Springer Science & Business Media, 2012.

[18] A. M. Geoffrion. Generalized benders decomposition. *Journal of optimization theory and applications*, 10(4):237–260, 1972.

[19] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995.

[20] C. R. Henderson. A simple method for computing the inverse of a numerator relationship matrix used in prediction of breeding values. *Biometrics*, 32(1):69–83, 1976.

[21] S. Kim and M. Kojima. Second order cone programming relaxation of nonconvex quadratic optimization problems. *Optimization Methods and Software*, 15(3-4):201–224, 2001.

[22] Y. N. Kiseliov. Algorithms of projection of a point onto an ellipsoid. *Lithuanian Mathematical Journal*, 34(2):141–159, 1994.

[23] M. Kojima, S. Shindoh, and S. Hara. Interior-point methods for the monotone semidefinite linear complementarity problem in symmetric matrices. *SIAM Journal on Optimization*, 7(1):86–125, 1997.

[24] A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica: Journal of the Econometric Society*, pages 497–520, 1960.

[25] J. D. Little, K. G. Murty, D. W. Sweeney, and C. Karel. An algorithm for the traveling salesman problem. *Operations research*, 11(6):972–989, 1963.

[26] M. Lynch and B. Walsh. *Genetics and analysis of quantitative traits*. Sinauer Sunderland, MA, 1998.

[27] T. H. Meuwissen. GENCONT: an operational tool for controlling inbreeding in selection and conservation schemes. In *Proceedings of the 7th Congress on Genetics Applied to Livestock Production*, pages 19–23, 2002.

[28] T. H. E. Meuwissen. Maximizing the response of selection with a predefined rate of inbreeding. *Journal of animal science*, 75:934–940, 1997.

[29] R. D. Monteiro. Primal–dual path-following algorithms for semidefinite programming. *SIAM Journal on Optimization*, 7(3):663–678, 1997.

[30] T. J. Mullin. OPSEL 2.0: A computer program for optimal selection in tree breeding. *Arbetsrapport från Skogforsk Nr 954-2017, Skogforsk, Uppsala, SE*, 2017.

[31] T. J. Mullin and P. Belotti. Using branch-and-bound algorithms to optimize selection of a fixed-size breeding population under a relatedness constraint. *Tree Genetics & Genomes*, 12(1):1–12, 2016.

[32] T. J. Mullin, J. Hallander, O. Rosvall, and B. Andersson. Using simulation to optimise tree breeding programmes in Europe: an introduction to POPSIM. Technical Report Nr. 711-2010, Arbetsrapport fran Skogforsk, 2010.

[33] K. Murota. *Discrete convex analysis*. SIAM, 2003.

[34] K. Murota. On steepest descent algorithms for discrete convex functions. *SIAM Journal on Optimization*, 14(3):699–707, 2004.

[35] Y. Nesterov and A. Nemirovski. *Interior-point polynomial algorithms in convex programming*, volume 13. SIAM, 1994.

[36] R. Pong-Wong and J. A. Woolliams. Optimisation of contribution of candidate parents to maximise genetic gain and restricting inbreeding using semidefinite programming (open access publication). *Genetics Selection Evolution*, 39(1):3, 2007.

[37] R. Quaas. Computing the diagonal elements and inverse of a large numerator relationship matrix. *Biometrics*, 32:949–953, 1976.

[38] S. Safarina, S. Moriguchi, T. J. Mullin, and M. Yamashita. Conic relaxation approaches for equal deployment problems. *To appear in Discrete Applied Mathematics*, 2019.

[39] S. Safarina, T. J. Mullin, and M. Yamashita. A cone decomposition method for optimal contribution selection in fprest tree management. 数理解析研究所講究録, (2108):14–21, 2019.

[40] S. Safarina, T. J. Mullin, and M. Yamashita. Polyhedral-based methods for mixed-integer socp in tree breeding. *To appear in Journal of the Operations Research Society of Japan*, 62(3), 2019.

[41] S. Safarina and M. Yamashita. An application of polyhedral relaxations to optimal contribution selection of tree breeding problem. 数理解析研究所講究録, (2069):62–73, 2018.

[42] J. Sherman and W. J. Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, 21(1):124–127, 1950.

[43] G. Sierksma and Y. Zwols. *Linear and Integer Optimization: Theory and Practice*. CRC Press, 2015.

[44] M. J. Todd. Semidefinite optimization. *Acta Numerica*, 10:515–560, 2001.

[45] K. C. Toh, M. J. Todd, and R. H. Tütüncü. SDPT3 – a MATLAB software package for semidefinite programming, version 1.3. *Optimization Methods and Software*, 11 & 12(1-4):545–581, 1999.

[46] P. Tseng. Further results on approximating nonconvex quadratic optimization by semidefinite programming relaxation. *SIAM Journal on Optimization*, 14(1):268–283, 2003.

[47] N. Tsuchimura, S. Moriguchi, and K. Murota. Discrete convex optimization solvers and demonstration softwares. *Transactions of the Japan Society for Industrial and Applied Mathematic*, 23(2):233–252, 2013. (In Japanese).

[48] J. P. Vielma, S. Ahmed, and G. L. Nemhauser. A lifted linear programming branch-and-bound algorithm for mixed-integer conic quadratic programs. *INFORMS Journal on Computing*, 20(3):438–450, 2008.

[49] J. P. Vielma, I. Dunning, J. Huchette, and M. Lubin. Extended formulations in mixed integer conic quadratic programming. *Mathematical Programming Computation*, 9(3):369–418, 2017.

[50] R. Wituła and D. Słota. Cardano's formula, square roots, chebyshev polynomials and radicals. *Journal of Mathematical Analysis and Applications*, 363(2):639–647, 2010.

[51] H. Wolkowicz, R. Saigal, and L. Vandenberghe. *Handbook of semidefinite programming: theory, algorithms, and applications*, volume 27. Springer Science & Business Media, 2012.

[52] S. Wright. Coefficients of inbreeding and relationship. *The American Naturalist*, 56(645):330–338, 1922.

[53] M. Yamashita, K. Fujisawa, M. Fukuda, K. Kobayashi, K. Nakta, and M. Nakata. Latest developments in the SDPA family for solving large-scale SDPs. In M. F. Anjos and J. B. Lasserre, editors, *Handbook on Semidefinite, Cone and Polynomial Optimization: Theory, Algorithms, Software and Applications*, chapter 24, pages 687–714. Springer, NY, USA, 2012.

[54] M. Yamashita, K. Fujisawa, and M. Kojima. Implementation and evaluation of SDPA 6.0 (SemiDefinite Programming Algorithm 6.0). *Optimization Methods and Software*, 18(4):491–505, 2003.

[55] M. Yamashita, T. J. Mullin, and S. Safarina. An efficient second-order cone programming approach for optimal selection in tree breeding. *Optimization Letters*, 12(7):1683–1697, 2018.

A Theoretical evaluation of the SDP relaxation problems with a randomized algorithm

The solutions obtained from the conic relaxation problem are not always feasible solutions of the ED problem since we ignored some constraints of the NP-hard problem to derive the conic relaxation problems. When SDP relaxation approaches are used, examination of randomized algorithms often follows to generate feasible solutions. In this section, we employ the result of Tseng [46] to give theoretical bounds on the expected objective value of a randomized algorithm. Corresponding to the ED problem, we analyze the performance of the SDP relaxation problem (3.3) by applying Theorem 2 of [46]. From the form of (3.3), the objective value at the optimal solution is $2\boldsymbol{g}_V^T\tilde{\boldsymbol{y}} + \bar{g}$. The following lemma provides a theoretical aspect of the expected value of this objective function in the randomized algorithm discussed in [46]. The lower bound is a direct consequence of Theorem 2 of [46], while the upper bound can be derived in a similar manner to [46] employing the evaluation in [19].

Lemma A.0.1. *For the ED problem, the expected objective value obtained through the randomized algorithm is bounded by*

$$\frac{2}{\pi}OPT_{SDP} + \left(1 - \frac{2}{\pi}\right)(-2\bar{\boldsymbol{g}}_V^T\boldsymbol{e}_V + \bar{g}) \leq E[2\bar{\boldsymbol{g}}_V^T\tilde{\boldsymbol{y}}_V + \bar{g}] \leq \alpha OPT_{SDP} + (1-\alpha)(2\bar{\boldsymbol{g}}_V^T\boldsymbol{e}_V + \bar{g}),$$

where $\alpha := \min\left\{\frac{2}{\pi}\frac{\theta}{1-\cos\theta} : 0 \leq \theta \leq \pi\right\} \approx 0.878.$

From a theoretical viewpoint, Lemma A.0.1 gives the bounds on the expected objective value $E[2\bar{\boldsymbol{g}}_V^T\tilde{\boldsymbol{y}}_V + \bar{g}]$ of the randomized algorithm. When we examined numerical tests, we observed that the interval between the lower and upper bounds are not so sharp, and the expected objective value. The dataset we used here is a subset of datasets in Chapter 4. The first column shows Z, the number of genotype candidates. We fix the number of chosen candidates to $N = 50$. The third and fifth columns are the lower and the upper bounds in Lemma A.0.1, respectively. The expected objective value is shown in the fourth column, and it is obtained by generating the random vector $\boldsymbol{v}$ thousand times and taking the average of the thousand trials. The sixth column is the optimal value of the SDP relaxation problem. For the smallest size $Z = 200$, the gap between the lower and upper bounds was not so large. However, when we tried the larger problems, the gap was getting worse. In particular, the ratio of the upper bound to the lower objective value for the case $Z = 5050$ goes beyond 139.

Another characteristic in the randomized algorithm is that the expected objective value is always larger than OPT_{SDP}. A reason for this is that the generated solution $\tilde{\boldsymbol{y}}_V$ is not guaranteed to satisfy the constraint $\boldsymbol{y}_V^T\boldsymbol{A}_{VV}\boldsymbol{y}_V + 2\bar{\boldsymbol{c}}_F^T\boldsymbol{y}_V \leq 2\bar{\theta}$ that corresponds to $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} \leq 2\theta$ of

Table A.1: Theoretical bounds on the expected values by the randomized algorithm

Z	2θ	lower bound	expected value	upper bound	OPT_{SDP}
200	0.0334	16.161	25.812	30.340	25.386
1050	0.0627	5.075	32.305	112.600	24.938
2045	0.0711	279.259	446.089	2007.212	438.659
5050	0.1081	5.775	284.965	806.205	42.786

Due to this weaker bounds reported in Table A.1, we seek, in Chapter 4, an optimization method that can obtain a reasonable solution for practical use.