Journal of the Operations Research Society of Japan
Vol. 62, No. 4, October 2019, pp. 133–151

POLYHEDRAL-BASED METHODS FOR MIXED-INTEGER SOCP IN TREE BREEDING

Sena Safarina
Tokyo Institute of Technology

Tim J. Mullin
The Swedish Forestry Research Institute (Skogforsk)

Makoto Yamashita
Tokyo Institute of Technology

(Received November 15, 2018; Revised April 29, 2019)

Abstract Optimal contribution selection (OCS) is a mathematical optimization problem that aims to maximize the total benefit from selecting a group of individuals under a constraint on genetic diversity. We are specifically focused on OCS as applied to forest tree breeding, where selected individuals will contribute equally to the gene pool. Since the diversity constraint in OCS can be described with a second-order cone, equal deployment in OCS can be mathematically modeled as mixed-integer second-order cone programming (MI-SOCP). However, if we apply a general solver for MI-SOCP, non-linearity embedded in OCS requires a heavy computation cost. To address this problem, we propose an implementation of lifted polyhedral programming (LPP) relaxation and a cone-decomposition method (CDM) by generating effective linear approximations for OCS. Furthermore, to enhance the performance of CDM, we utilize the sparsity structure that can be discovered in OCS. Through numerical experiments, we verified CDM with the sparse structure successfully solves OCS problems much faster than generic approaches for MI-SOCP.

Keywords: Optimization; Second-order cone programming; Conic relaxation; Mixed-integer conic programming; Equal deployment problem; Geometric cut; Tree Breeding; Optimal selection

1. Introduction

As in other types of breeding, 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) [4, 20, 21, 25] 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.

A mathematical optimization formulation for UDP is given by Meuwissen [13] as follows:

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

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 $\boldsymbol{g} = (g_1, g_2, \dots, g_m)^T$ contains the estimated breeding values (EBVs) [27] representing the genetic value of candidates in $\boldsymbol{x}$. In this paper, 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) 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 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 [14], 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 [24] 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. [20] reported that the computation cost of the SDP approach was very high, even when using a parallel SDP solver [15,31]. To reduce the heavy computation burden, Yamashita et al. [4] proposed an efficient numerical method based on second-order cone programming (SOCP) [8].

The current research is mainly concerned with EDP of form:

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

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 paper, we assume that (2) is feasible.

The OCS problem has been widely solved through a software package `GENCONT` developed by Meuwissen [25]. 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` [26], was proposed by Mullin and Belotti [21]. `dsOpt` is an implementation of the branch-and-bound method combined with an outer approximation method [11]. 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 [21, 25], this paper is focused on the fact that the crucial quadratic constraint $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} \leq 2\theta$ in (1) and (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 paper, 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 OCS problem (2) into an MI-SOCP (mixed-integer second-order cone programming) formulation:

$$\begin{array}{lll} \text{maximize} & : & \frac{\boldsymbol{g}^T\boldsymbol{y}}{N} \\ \text{subject to} & : & \boldsymbol{e}^T\boldsymbol{y} = N,\ y_i \in \{0,1\}\ (i = 1,\ldots,m),\ \left(\sqrt{2\theta}N, \boldsymbol{U}\boldsymbol{y}\right) \in \mathcal{K}^m. \end{array} \tag{3}$$

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 MI-SOCP formulations [22, 29, 30, 32], but these cannot directly exploit the structure of the matrices $\boldsymbol{A}$ or $\boldsymbol{U}$.

In this paper, we examine three approaches. The first one is a lifted polyhedral programming relaxation with active constraint selection method (LPP-ACSM) that removes the non-linearity, exploiting an extension of polyhedral programming relaxation for the second-order cone programming problem [16, 17, 23]. The second approach is a cone decomposition method (CDM) that is based on a geometric cut in a combination with a Lagrangian multiplier method. A cutting plane is a geometric cut if the plane is computed with an orthogonal projection [7]. Cone decomposition itself has already been used in `CPLEX`, but there depends on an outer approximation, therefore, the proposed CDM generates a different linear approximation. The last approach to reduce heavy computation burden is utilizing the sparsity found in the inverse of Wright's numerator relationship matrix $\boldsymbol{A}^{-1}$. This sparsity exploitation is the key property developed in [4] (See also Section 4). This sparsity and the geometric cuts are combined into a sparse linear approximation, and it can strongly enhance the performance of CDM.

In addition, we prove that the Lagrangian multiplier method in the framework of CDM gives the analytical form for the geometric cut, therefore, the proposed CDM and and its sparse variant generate the linear cuts without relying on iterative methods.

The remainder of this paper is organized as follows. In Section 2, we briefly review LPP, then propose its enhancement LPP-ACSM. Section 3 proposes the framework of CDM and demonstrates that the geometric cut in CDM has an analytical form. Section 4 is focused on CDM with the enhancement by exploiting the sparsity structure in the inverse matrix. The numerical results will be presented in Section 5. Finally, in Section 6, we give some conclusions and discuss future studies.

2. Lifted Polyhedral Programming Relaxation

Lifted polyhedral programming (LPP) relaxation [16,23] 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 $\mathcal{P}_\epsilon^m$ to approximate $\mathcal{K}^m$. Here, $\epsilon > 0$ is a parameter to control the tightness of LPP relaxation. Vielma et al. [16] gave the detailed formulation of $\mathcal{P}_\epsilon^m$. They first decompose the $(m+1)$-dimensional second-order cone $\mathcal{K}^m$ into multiple 2-dimensional second-order cones $\mathcal{K}^2$ and linear constraints:

$$\begin{aligned}\mathcal{K}^m := \{ &(v_0, \boldsymbol{v}) \in \mathbb{R}_+ \times \mathbb{R}^m : \exists (\delta^j)_{j=0}^J \in \mathbb{R}^{T(m)} \text{ such that} \\ &v_0 = \delta_1^J, \\ &\delta_i^0 = v_i \ (i = 1, \cdots, m), \\ &\left(\delta_{2i-1}^j, \delta_{2i}^j, \delta_i^{j+1}\right) \in \mathcal{K}^2 \ \left(i = 1, \cdots, \left\lfloor \frac{t_j}{2} \right\rfloor, \ j = 0, \cdots, J-1\right), \\ &\delta_{t_j}^j = \delta_{\lceil t_j/2 \rceil}^{j+1} \text{ for } j = 0, \cdots, J-1 \text{ such that } t_j \text{ is odd}\}\end{aligned} \tag{4}$$

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, $\mathcal{K}^4$ 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 $T(4) = \lfloor \frac{4}{2} \rfloor + \lfloor \frac{2}{2} \rfloor = 3$ cones of $\mathcal{K}^2$; $v_0^2 \geq (\delta_1^1)^2 + (\delta_2^1)^2$, $(\delta_1^1)^2 \geq v_1^2 + v_2^2$ and $(\delta_2^1)^2 \geq v_3^2 + v_4^2$.

Then, a replacement of $\mathcal{K}^2$ in (4) by $\mathcal{W}_j(\epsilon)$ defined below generates $\mathcal{P}_\epsilon^m$:

$$\begin{aligned}\mathcal{W}_j(\epsilon) := \Big\{ &(v_0, v_1, v_2) \in \mathbb{R}_+ \times \mathbb{R}^2 : \exists (\alpha, \beta) \in \mathbb{R}^{2 s_j(\epsilon)} \text{ such that} \\ &v_0 = \alpha_{s_j(\epsilon)} \cos\left(\frac{\pi}{2^{s_j(\epsilon)}}\right) + \beta_{s_j(\epsilon)} \sin\left(\frac{\pi}{2^{s_j(\epsilon)}}\right), \ \alpha_1 = v_1 \cos(\pi) + v_2 \sin(\pi), \\ &\beta_1 \geq |v_2 \cos(\pi) - v_1 \sin(\pi)|, \ \alpha_{i+1} = \alpha_i \cos\left(\frac{\pi}{2^i}\right) + \beta_i \sin\left(\frac{\pi}{2^i}\right), \\ &\beta_{i+1} \geq \left|\beta_i \cos\left(\frac{\pi}{2^i}\right) - \alpha_i \sin\left(\frac{\pi}{2^i}\right)\right|, \ \text{ for } i \in \{1, \ldots, s_j(\epsilon) - 1\}\Big\}\end{aligned} \tag{5}$$

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 [17].

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.

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

$$\beta_1 \geq -v_2 \cos(\pi) + v_1 \sin(\pi) \tag{6}$$

in (5) 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 (6).

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 (3).

Algorithm 2.1. [LPP relaxation with active constraint selection method (LPP-ACSM)]

1. Build an MI-LP problem by replacing $\mathcal{K}^m$ in (3) with $\mathcal{P}^m_\epsilon$.
2. Replace the inequality constraints of form $\beta_1 \geq |v_2\cos(\pi) - v_1\sin(\pi)|$ in (5) with the corresponding equality constraints $\beta_1 = -v_2\cos(\pi) + v_1\sin(\pi)$.
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 (4). 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.

3. Cone Decomposition Method

In this section, we propose a cone decomposition method (CDM) for EDP (3). The basic concept of the cone decomposition method also draws on the properties of second-order cones. The above LPP approach decomposes the $(m+1)$-dimensional second-order cone $\mathcal{K}^m$ into multiple two-dimensional second-order cones $\mathcal{K}^2$ in a recursive style as shown in (4). In contrast, the proposed CDM makes use of different decomposition, based on the following theorem from [6].

Theorem 3.1. [6] *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.*

Theorem 3.1 gives another decomposition of $\mathcal{K}^m$ by using an auxiliary vector $\boldsymbol{w} \in \mathbb{R}^m$.

Corollary 3.2. *A second-order cone* $\mathcal{K}^m$ *can be also written as*

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

The utilization of Corollary 3.2 leads to another reformulation of our OCS (3) as follows:

$$\begin{array}{lll} \text{maximize} & : & \frac{\boldsymbol{g}^T\boldsymbol{y}}{N} \\ \text{subject to} & : & \boldsymbol{e}^T\boldsymbol{y} = N, \boldsymbol{z} = \boldsymbol{U}\boldsymbol{y}, \\ & & z_i^2 \leq w_i c_0 \ (i = 1, \ldots, m), \ \sum_{i=1}^m w_i \leq c_0, \ y_i \in \{0, 1\} \ (i = 1, \ldots, m) \end{array} \tag{7}$$

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 $\boldsymbol{w}$.

The nonlinear constraint in (7) is only the quadratic constraint $z_i^2 \leq w_i c_0$ 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 [7].

The framework of the proposed CDM is given as Algorithm 3.3.

Algorithm 3.3. [Cone decomposition method (CDM)]

Step 1 Let P^0 be an MI-LP problem that is generated from an optimization problem (7) by omitting the quadratic constraints $z_i^2 \leq w_i c_0$ $(i = 1, \dots, m)$. However, this constraint implicitly guarantees the nonnegativity of w_i. To make the MI-LP relaxation tighter, we explicitly add $w_i \geq 0$ $(i = 1, \dots, m)$. Apply an MI-LP solver to P^0, and let its optimal solution be $\left(\hat{\boldsymbol{y}}^0, \hat{\boldsymbol{z}}^0, \hat{\boldsymbol{w}}^0\right)$. Let $k = 0$.

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

Step 3 For each $i = 1, \dots, m$, 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 $\mathcal{C}^k$ 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 $\mathcal{C}^k$ is empty, output $\hat{\boldsymbol{y}}^k$ as the solution and terminate.

Step 5 Build a new MI-LP P^{k+1} by adding $\mathcal{C}^k$ 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 3.3, 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 [9] proposed some numerical method, but this is an iterative method. Another iterative method is also proposed by [5] 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 3.2 enables us to derive this theorem.

Theorem 3.4. *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 3.4, we make use of Cardano's Formula [18] to obtain a root of a cubic function analytically.

Theorem 3.5. [Cardano's Formula [28]] *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 3.4)

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

$$\begin{array}{ll} \text{minimize} & : \frac{1}{2}(z-\hat{z})^2 + \frac{1}{2}(w-\hat{w})^2 \\ \text{subject to} & : z^2 \leq wc_0. \end{array} \tag{8}$$

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 (8) is equivalent to the following optimization problem:

$$\begin{array}{ll} \text{minimize} & : \frac{1}{2}(z-\hat{z})^2 + \frac{1}{2}(w-\hat{w})^2 \\ \text{subject to} & : z^2 = wc_0. \end{array} \tag{9}$$

To apply the Lagrangian multiplier method, a Lagrangian function of (9) with a Lagrangian multiplier $\lambda \in \mathbb{R}$ is given by:

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

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

$$\nabla_z \mathcal{L} = z - \hat{z} + 2\lambda z = 0, \tag{10}$$
$$\nabla_w \mathcal{L} = w - \hat{w} - \lambda c_0 = 0, \tag{11}$$
$$\nabla_\lambda \mathcal{L} = -c_0 w + z^2 = 0. \tag{12}$$

Substituting (10) and (11) into (12) leads to a cubic function with respect to λ:

$$4c_0^2\lambda^3 + (4c_0^2 + 4c_0\hat{w})\lambda^2 + (c_0^2 + 4c_0\hat{w})\lambda + (c_0\hat{w} - \hat{z}^2) = 0. \tag{13}$$

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 3.5 to obtain λ. In Theorem 3.5, 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

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

we can prove that $Q^3 + R^2 \neq 0$ by showing the numerator is nonzero. Substituting a, b, c, 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 $w_i \geq 0$ $(i = 1, \dots, m)$, 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 (10) and (11). Therefore, the optimal solution $(\bar{z}, \bar{w})$ of (8) has an analytical form. $\square$

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

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

$$C_3 = \frac{2\theta N + \sqrt{2\theta}\hat{w}}{3\theta N}, \text{ and}$$

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

Note that the analytical solution of (8) can be also proved via solving a one-dimensional unconstrained optimization problem. Rewriting problem (8) 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 (8) 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 3.6. *Algorithm 3.3 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 $y_i \in \{0, 1\}$ $(i = 1, \dots, m)$. In the kth iteration, the generated cuts in $\mathcal{C}^k$ 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. $\square$

4. 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 [4] to reduce the computation time for solving UDP (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.

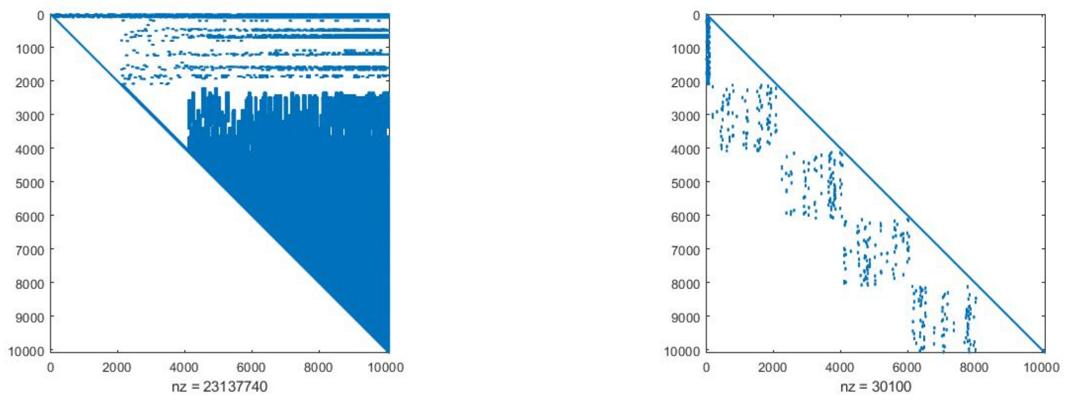
As indicated earlier, Wright's numerator relationship matrix $\boldsymbol{A}$ describes the additive genetic similarity between individuals. Each element of $\boldsymbol{A}$ is determined by pedigree information [14]. When $p(i)$ and $q(i)$ are the parents of genotype i (we can assume $1 \leq p(i) < q(i) < i \leq m$ without loss of generality), the element $\boldsymbol{A}_{ij}$ is given by $\boldsymbol{A}_{ij} = \frac{\boldsymbol{A}_{j,p(i)}+\boldsymbol{A}_{j,q(i)}}{2}$. $\boldsymbol{A}$ is important for explaining genetic (co)variances and has numerous applications in the field of quantitative genetics, including the methods for estimation of breeding values that require inversion of $\boldsymbol{A}$. Henderson [3] outlined a method for deriving $\boldsymbol{A}^{-1}$, based on the root-free factorization of $\boldsymbol{A}$ and showed the high sparsity of the inverse triangular factor of $\boldsymbol{A}$. An efficient use of this sparsity then allows direct computation of $\boldsymbol{A}^{-1}$ as a sum of individual contributions based on a chronological reading of the pedigree.

The resulting sparse structure of $\boldsymbol{A}^{-1}$ was the key property in [4] to reduce the computation time for solving UDP (1). We also 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 [3]. 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 [4]. If we use $\boldsymbol{b}_i^T$ to denote the ith row of $\boldsymbol{B}$, $\boldsymbol{b}_i^T$ is given by

$$\boldsymbol{b}_i^T = \begin{cases} \sqrt{\alpha_i}\boldsymbol{e}_i^T & \text{for} \quad i \in \mathcal{P}_0, \\ \sqrt{\alpha_i}\left(\boldsymbol{e}_i - \frac{1}{2}\boldsymbol{e}_{p(i)}\right)^T & \text{for} \quad i \in \mathcal{P}_1, \\ \sqrt{\alpha_i}\left(\boldsymbol{e}_i - \frac{1}{2}\boldsymbol{e}_{p(i)} - \frac{1}{2}\boldsymbol{e}_{q(i)}\right)^T & \text{for} \quad i \in \mathcal{P}_2. \end{cases} \tag{14}$$

Here, $\alpha_i \in \mathbb{R}$ $(i = 1,\dots,m)$ is a constant computed using Quaas's algorithm [2], and $\boldsymbol{e}_i \in \mathbb{R}^m$ is the ith unit vector. From the actual value of $\alpha_i \in \mathbb{R}$ $(i = 1,\dots,m)$ obtained by Quaas's algorithm, it holds that $\boldsymbol{b}_i^T\boldsymbol{b}_i \leq 2$. The set of genotype $\{1,\dots,m\}$ 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).



(a) Matrix $\boldsymbol{U}$ (b) Matrix $\boldsymbol{B}$

Figure 1: Non-zero element structure of matrix $\boldsymbol{U}$ and $\boldsymbol{B}$ for $m = 10100$

The important property in (14) 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 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 (3) is equivalent to a quadratic constraint $\boldsymbol{y}^T\boldsymbol{A}\boldsymbol{y} \le 2\theta N^2$. 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 $\boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{v} \le 2\theta N^2$. Since $\boldsymbol{A}^{-1} = \boldsymbol{B}^T\boldsymbol{B}$ and $\boldsymbol{b}_i^T$ is the ith row of $\boldsymbol{B}$, the quadratic constraint $\boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{v} \le 2\theta N^2$ is further equivalent to

$$(\sqrt{2\theta}N, (\boldsymbol{b}_1^T\boldsymbol{v}, \boldsymbol{b}_2^T\boldsymbol{v}, \ldots, \boldsymbol{b}_m^T\boldsymbol{v})^T) \in \mathcal{K}^m. \tag{15}$$

Thus, using Corollary 3.2 and $c_0 = \sqrt{2\theta}N$, (15) can be decomposed into

$$\sum_{i=1}^{m} w_i \le c_0 \text{ and } (\boldsymbol{b}_i^T\boldsymbol{v})^2 \le w_i c_0 \ (i = 1, \ldots, m). \tag{16}$$

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

Theorem 4.1. *Assume that $(\hat{\boldsymbol{v}}, \hat{w})$ violates $(\boldsymbol{b}^T\boldsymbol{v})^2 \le wc_0$. Then, the geometric cut $d^w w + (\boldsymbol{d}^v)^T\boldsymbol{v} + d^0 \le 0$ by the orthogonal projection onto $(\boldsymbol{b}^T\boldsymbol{v})^2 \le wc_0$ 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. In contrast, the cutting plane used in `dsOpt` [26] is derived from the first-order Taylor series. For an iterate $\boldsymbol{y}^k$ that violates the quadratic constraint $\boldsymbol{y}^T\boldsymbol{A}\boldsymbol{y} \le 2\theta N^2$, the cut in `dsOpt` is of form $(\boldsymbol{A}\boldsymbol{y}^k)^T\boldsymbol{y} \le \sqrt{2\theta N^2}\sqrt{(\boldsymbol{y}^k)^T\boldsymbol{A}\boldsymbol{y}^k}$. 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 4.1.

Proof. In a similar way to Theorem 3.4, let $(\bar{\boldsymbol{v}}, \bar{w})$ be the orthogonal projection of $(\hat{\boldsymbol{v}}, \hat{w})$ onto $(\boldsymbol{b}^T\boldsymbol{v})^2 \le wc_0$. Then, $(\bar{\boldsymbol{v}}, \bar{w})$ can be obtained by the following sub-problem:

$$\begin{array}{ll} \text{minimize} & : \frac{1}{2}||\boldsymbol{v} - \hat{\boldsymbol{v}}||^2 + \frac{1}{2}(w - \hat{w})^2 \\ \text{subject to} & : (\boldsymbol{b}^T\boldsymbol{v})^2 = wc_0. \end{array} \tag{17}$$

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

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

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 3.4 is an application of the Sherman-Morrison-Woodbury formula [1]. The use of this formula leads to

$$\boldsymbol{v} = (\boldsymbol{I} + 2\lambda \boldsymbol{b}\boldsymbol{b}^T)^{-1}\hat{\boldsymbol{v}} = \hat{\boldsymbol{v}} - \frac{2\lambda(\boldsymbol{b}^T\hat{\boldsymbol{v}})\boldsymbol{b}}{1 + 2\lambda \boldsymbol{b}^T\boldsymbol{b}}. \tag{18}$$

The substitution of $w = \hat{w} + \lambda c_0$ and (18) into $wc_0 = (\boldsymbol{b}^T\boldsymbol{v})^2$ results in the following cubic function

$$(4c_0^2(\boldsymbol{b}^T\boldsymbol{b})^2)\lambda^3 + (4c_0^2\boldsymbol{b}^T\boldsymbol{b} + 4\hat{w}c_0(\boldsymbol{b}^T\boldsymbol{b})^2)\lambda^2 + (c_0^2 + 4\hat{w}c_0(\boldsymbol{b}^T\boldsymbol{b}))\lambda + (\hat{w}c_0 - (\boldsymbol{b}^T\hat{\boldsymbol{v}})^2) = 0. \tag{19}$$

We can use a similar step to Theorem 3.4 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 3.5 guarantees an analytical form on $\bar{\lambda}$. We put the coefficients of (19) 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 $\boldsymbol{b}$. $\square$

With the geometric cut for the cone $(\boldsymbol{b}_i^T\boldsymbol{v})^2 \leq w_ic_0$, we develop Algorithm 4.2.

Algorithm 4.2. [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 $(\boldsymbol{b}_i^T\boldsymbol{v})^2 \leq w_ic_0$ $(i = 1, \ldots, m)$. Apply an MI-LP solver to P^0, and let its optimal solution be $\left(\hat{\boldsymbol{y}}^0, \hat{\boldsymbol{v}}^0, \hat{\boldsymbol{w}}^0\right)$. Let $k = 0$.

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

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 4.1 to $\mathcal{C}^k$.

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

Step 5 Build a new MI-LP P^{k+1} by adding $\mathcal{C}^k$ 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 4.2 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 4.3. *Algorithm 4.2 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. □

5. Numerical Results

Numerical experiments were conducted to compare the performance of the three proposed methods (LPP-ACSM, CDM, and CDM-B) with existing software (`dsOpt` as implemented in `OPSEL`, `GENCONT`, a general MI-SOCP solver `CPLEX`, and existing LPP). The proposed methods were implemented using Matlab R2017b by setting `CPLEX` as the MI-LP solver. All methods were executed on a 64-bit Windows 10 PC with Xeon CPU E3-1231 (3.40 GHz) and 8 GB memory space. The data were taken from `https://doi.org/10.5061/dryad.9pn5m` or generated by the simulation POPSIM [10]. The sizes of the test instances are $m = 200, 1050, 2045, 5050, 10100$, and 15222. We set parameter $N = 50, 100$, and as a stopping criterion for `CPLEX`, we used gap $= 1\%, 5\%$. The computation time was limited to 3 hours for each execution.

We are first focused on the results from the OCS solver `GENCONT` in Table 1. In this table, the first column is m, the number of candidates, while the second column is 2θ. The columns "$\boldsymbol{g}^T\boldsymbol{x}$" and "$\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$" are the obtained objective values and group coancestry, respectively. The fifth column shows the computation time, and the last column the number of chosen candidates by `GENCONT`. In particular, if $\boldsymbol{x}$ is a feasible solution, it should hold $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} \leq 2\theta$ and the number of chosen candidate should be exactly N. We only show the solution for $m \leq 5050$, since the results with $m = 10100$ and 15222 were not obtained due to out of memory.

From Table 1, we observe that the number of chosen candidates did not match the given parameter N. This indicates that `GENCONT` failed to output feasible solutions.

The results for other methods where $N = 50$ and $N = 100$ are presented in Tables 2 and 3, respectively. In contrast to `GENCONT`, the other methods output the solution that match N. The first column indicates methods conducted in the numerical experiments. In the tables, while `CPLEX-default` is used to solve (3) using the default setting; `CPLEX-LPrelax` is used for solving (7) by setting `mip.strategy.search=2` in `CPLEX` to use LP relaxation forcibly.

For the LPP relaxation and its modification (LPP-ACSM), we fixed $\epsilon = 0.005$ for generating $\mathcal{P}_\epsilon^m$, so that these two methods output feasible solutions. In addition, since only LPP and LPP-ACSM require the parameter ϵ, we show the value of ϵ for only these two methods in the column $(1+\epsilon)2\theta$. The other methods do not need the parameter ϵ, and this is indicated by "*" in the tables.

Table 1: Numerical results on `GENCONT`

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

When the computation could not finish within the time limit of 3 hours, it is indicated as '> 3 hours' and the best objective values up to that point are shown in the tables. We indicate out of memory by "OOM."

Table 2: Numerical comparison for EDPs ($N = 50$)

Method	m	2θ	$(1+\epsilon)2\theta$	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-default			*	24.99	0.03340	1.06	25.19	0.03340	8735.24
CPLEX-LPrelax			*	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
LPP	200	0.0334	0.03373	24.83	0.03340	26.81	25.11	0.03340	3691.26
LPP-ACSM			0.03373	24.84	0.03340	47.75	25.15	0.03340	2587.16
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-default			*	24.97	0.06267	3.56	24.97	0.06267	6.64
CPLEX-LPrelax			*	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
LPP	1050	0.0627	0.06333	24.54	0.06129	976.54	24.89	0.06291	10063.39
LPP-ACSM			0.06333	24.72	0.06215	1230.01	24.89	0.06291	1634.58
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-default			*	437.21	0.07100	3.95	437.21	0.07100	3.83
CPLEX-LPrelax			*	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
LPP	2045	0.0711	0.07181			OOM			OOM
LPP-ACSM			0.07181			OOM			OOM
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-default			*	41.90	0.10776	73.16	42.57	0.10781	> 3 hours
CPLEX-LPrelax			*	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
LPP	5050	0.1081	0.109184			OOM			OOM
LPP-ACSM			0.109184			OOM			OOM
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-default			*	44.89	0.06931	> 3 hours	44.89	0.06931	> 3 hours
CPLEX-LPrelax			*	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
LPP	10100	0.0701	0.070803			OOM			OOM
LPP-ACSM			0.070803			OOM			OOM
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-default			*	118.33	0.03840	> 3 hours	107.56	0.03280	> 3 hours
CPLEX-LPrelax			*	454.07	0.03860	350.14	458.85	0.03880	1080.17
dsOpt			*			OOM			OOM
LPP	15222	0.0388	0.039189			OOM			OOM
LPP-ACSM			0.039189			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

Table 3: Numerical comparison for EDPs ($N = 100$)

Method	m	2θ	$(1+\epsilon)2\theta$	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-default`			*	23.19	0.02580	4.31	23.49	0.02580	13.14
`CPLEX-LPrelax`			*	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
LPP	200	0.0258	0.026059	23.38	0.02585	10.60	23.53	0.02585	2289.65
LPP-ACSM			0.026059	23.24	0.02580	11.35	23.58	0.02585	4003.32
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-default`			*	22.53	0.05389	6.68	22.53	0.05389	3.64
`CPLEX-LPrelax`			*	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
LPP	1050	0.0539	0.054440	22.35	0.05401	1047.81	22.35	0.05401	1088.16
LPP-ACSM			0.054440	22.34	0.05392	1059.46			OOM
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-default`			*	420.04	0.06100	3.21	420.04	0.06100	3.08
`CPLEX-LPrelax`			*	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
LPP	2045	0.0628	0.063429			OOM			OOM
LPP-ACSM			0.063429			OOM			OOM
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-default`			*	40.63	0.09932	58.37	40.63	0.09932	54.43
`CPLEX-LPrelax`			*	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
LPP	5050	0.0994	0.100498			OOM			OOM
LPP-ACSM			0.100498			OOM			OOM
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-default`			*	43.79	0.06059	2720.18	44.34	0.06070	> 3 hours
`CPLEX-LPrelax`			*	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
LPP	10100	0.0610	0.061611			OOM			OOM
LPP-ACSM			0.061611			OOM			OOM
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-default`			*	436.92	0.02990	5084.69	436.92	0.02990	> 3 hours
`CPLEX-LPrelax`			*	423.75	0.02985	603.78	438.96	0.03000	710.21
`dsOpt`			*			OOM			OOM
LPP	15222	0.0300	0.030301			OOM			OOM
LPP-ACSM			0.030301			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

We expected that selecting active constraints could enhance the implementation of LPP. However, the result in Table 2 and 3 shows that both LPP and LPP-ACSM failed to obtain the solution due to OOM for large problems $m \geq 2045$. A relatively small $\epsilon = 0.005$ demands a huge number of linear constraints and this makes both methods slower as the problem size is getting larger. In addition, the improvement by LPP-ACSM was limited for the large problems.

Difficulty with memory consumption occurred not only in LPP and LPP-ACSM, but also in `dsOpt`; however, `dsOpt` only failed to handle the largest problem $m = 15222$ due to insufficient memory.

In contrast to the above methods, the other methods `CPLEX-default`, `CPLEX-LPrelax`, CDM, and CDM-B obtained the optimal solution without having such memory problems. In the case $m = 15222$, `CPLEX-default` could not complete its computation within the time limit (three hours), and the best objective values in the three hours were much worse than `CPLEX-LPrelax`, CDM, and CDM-B; for gap = 1%, while CDM and CDM-B obtained $\boldsymbol{g}^T\boldsymbol{x} = 461.83$, `CPLEX-default` only reached $\boldsymbol{g}^T\boldsymbol{x} = 107.56$.

`CPLEX-LPrelax` obtained the optimal solution less than three hours. This method is even more efficient than the proposed method CDM, excepting the largest problem. Therefore, from the difference between the result of `CPLEX-default` and those `CPLEX-LPrelax`, we can infer that the default setting of `CPLEX` cannot solve EDPs efficiently, and we have to explicitly let `CPLEX` know that LP relaxation is effective for EDPs.

Through Tables 2 and 3, CDM-B shows better performance among all methods including the proposed method CDM. CDM-B can even reduce the computation time into $\frac{1}{40}$ times that of CDM when $m = 10100$ and gap = 1%. For both gap = 5% and 1%, CDM-B is the most effective.

Table 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 4 presents an evidence supporting efficiency of CDM and CDM-B. The column "nnz" is the number of nonzero elements in MI-LP problems that are solved at the last iteration of CDM and CDM-B. The table shows that the number of nonzero elements of CDM-B is smaller than CDM, which leads to reductions in both computation time and memory size. For instance, in the column with $N = 50$, $m = 10100$, and gap = 5%, while CDM has 164574 nnz, CDM-B has only 34570 nnz. It means that the number of nonzero elements is CDM is almost 5 times CDM-B, and this is reflected in Table 2 that the computation time of CDM is much slower than CDM-B. Thus, the computation time is consistent with the number of nonzero elements.

6. Conclusion and Future Work

In this paper, we proposed LPP with an active constraint selection method (LPP-ACSM), cone decomposition method (CDM), and CDM with sparse matrix (CDM-B) to achieve optimal contribution selection in the context of tree breeding. We compared the efficiency of the proposed methods with those found in existing breeding selection software (`GENCONT` and `dsOpt`), the optimization solver `CPLEX`, and LPP. From the numerical results, we observed that LPP and LPP-ACSM failed to obtain solutions for problems with large m due to out of memory. Since we needed very tight ϵ, the number of constraints tended to be huge.

Our final proposed method, CDM-B, can efficiently obtain the optimal solution of EDP. For the largest problem $m = 15222$, while `CPLEX-default` could not find satisfactory solutions in three hours, CDM-B can efficiently obtain favourable feasible solutions in just two minutes.

In future studies, we will consider a combination of CDM-B with heuristic methods, for example, the method proposed in [19]. In particular, a feasible value obtained by the method of [19] would give a good tentative value in the framework of branch-and-bound for solving MI-LP (P^k). Another direction is that the decomposition in CDM-B and the generation of linear cuts can be used not only to solve the OCS problem in tree breeding, but can also be applied to other MI-SOCP problems. We will also consider another problem of OCS that involves not only simple binary constraints, but also other types of integer constraints.

7. Acknowledgement

Our work was partially supported by funding from JSPS KAKENHI (Grant-in-Aid for Scientific Research (C), 18K11176), the European Union Horizon 2020 Research and Innovation Program under Grant Agreement No. 676876348 (Project GenTree), and Föreningen Skogsträdsförädling (The Tree Breeding Association, Sweden).

References

[1] 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** (1950), 124–127.

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

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

[4] 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** (2018), 1683–1697.

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

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

[7] P. Białoń: Some variants of projection methods for large nonlinear optimization problems. *Journal of Telecommunications and Information Technology*, (2003), 43–49.

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

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

[10] T.J. Mullin: Popsim: a computer program for simulation of tree breeding programs over multiple generations. *Arbetsrapport från Skogforsk Nr. 984-2018, Skogforsk, Uppsala, SE*, (2018).

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

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

[13] T.H. Meuwissen: Maximizing the response of selection with a predefined rate of inbreeding. *Journal of animal science*, **75-4** (1997), 934–940.

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

[15] 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** (2003), 491–505.

[16] 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** (2008), 438–450.

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

[18] R. Witu la and D.S. lota: Cardano's formula, square roots, chebyshev polynomials and radicals. *Journal of Mathematical Analysis and Applications*, **363-2** (2010), 639–647.

[19] S. Safarina, S. Moriguchi, T.J. Mullin, and M. Yamashita: Conic relaxation approaches for equal deployment problems. *arXiv preprint arXiv:1703.03155*, (2017).

[20] 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** (2014), 27–34.

[21] 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** (2016), 4.

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

[23] D.P. Bertsekas: Nonlinear programming. *Journal of the Operational Research Society*, **48-3** (1997), 334–334.

[24] 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** (2007), 3.

[25] 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*, (2002), 19–23.

[26] 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).

[27] M. Lynch and B. Walsh: *Genetics and analysis of quantitative traits.* (Sinauer, Sunderland, 1998), 535–557.

[28] G. Cardano: *Ars magna or the rules of algebra* (Dover Publications, 1968).

[29] S. Drewes and S. Ulbrich: Subgradient based outer approximation for mixed integer second order cone programming. In J. Lee and S. Leyffer (eds.): *Mixed Integer Nonlinear Programming — The IMA Volumes in Mathematics and its Applications* (Springer, New York, 2012), 41–59.

[30] 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 M. Al-Baali, L. Grandinetti, and A. Purnama (eds.): *Numerical Analysis and Optimization — Springer Proceedings in Mathematics & Statistics* (Springer, Cham, 2015), 1–35.

[31] M. Yamashita, K. Fujisawa, M. Fukuda, K. Kobayashi, K. Nakata, and M. Nakata: Latest developments in the SDPA family for solving large-scale SDPs. In M. F. Anjos and J. B. Lasserre (ads.): *Handbook on Semidefinite, Conic and Polynomial Optimization — International Series in Operations Research & Management Science*, **166**, (Springer, Boston, MA, 2012), 687–713.

[32] H.Y. Benson and U. Sağlam: Mixed-Integer Second-Order Cone Programming: A Survey. *INFORMS TutORials in Operations Research*, (2013), 13–36. `https://doi.org/10.1287/educ.2013.0115`.

Sena Safarina
Department of Mathematical and Computing Science
Tokyo Institute of Technology, 2-12-1-W8-29
Tokyo 152-8552, Japan
E-mail: `safarina.s.aa@m.titech.ac.jp`