Simplex type algorithm for second-order cone programs via semi-infinite programming reformulation*

Yoshihiko Ito and Shunsuke Hayashi†

Abstract

The (linear) second-order cone program (SOCP) is to minimize a linear function over the intersection between a polyhedral set and an affine transformation of Cartesian product of second-order cones. For solving such a problem, the primal-dual interior-point method has been studied extensively so far and said to be the most efficient method by many researchers. On the other hand, the simplex type method for SOCP is much less spotlighted, while it still keeps an important position for linear programming (LP) problems. Actually, some researchers have tried to apply such a method to the SOCPs. However, in those existing studies, the proposed algorithms were not implemented practically, or could be applied only to some restricted class of problems.

In this paper, we apply the dual-simplex primal-exchange (DSPE) method, which was originally developed for solving linear semi-infinite programs (LSIP), to the SOCP by reformulating the second-order cone constraint as an infinite number of linear inequality constraints. Especially, by means of some numerical experiments, we observe that such a simplex type method can be more efficient than the existing interior-point based method, when we solve multiple SOCPs having similar data structures successively applying the so-called "hot start" technique.

1 Introduction

In this paper, we focus on the second-order cone program (SOCP) [1] of the form

$$\begin{aligned} \underset{x\in\mathbb{R}^m}{\text{minimize}} \quad & c^\top x \\ \text{subject to} \quad & Ax + b \in \mathcal{K}, \end{aligned} \tag{1.1}$$

where $A \in \mathbb{R}^{n\times m}$, $b \in \mathbb{R}^n$ and $c \in \mathbb{R}^m$ are given matrix and vectors, and $\mathcal{K} \subseteq \mathbb{R}^n$ is given as

$$\mathcal{K} = \mathcal{K}^{n_1} \times \mathcal{K}^{n_2} \times \cdots \times \mathcal{K}^{n_p}$$

with an n_i-dimensional second-order cone (SOC)

$$\mathcal{K}^{n_i} := \begin{cases} \left\{ u = (u_1, \overline{u}) \in \mathbb{R} \times \mathbb{R}^{n_i-1} \,\middle|\, u_1 \geq \|\overline{u}\|, u_1 \in \mathbb{R}, \overline{u} \in \mathbb{R}^{n_i-1} \right\} & (n_i \geq 2) \\ \mathbb{R}_+ = \{ u \in \mathbb{R} \mid u \geq 0 \} & (n_i = 1). \end{cases}$$

Here, $n = n_1 + n_2 + \cdots + n_p$ and $\|\cdot\|$ denotes the Euclidean norm. Throughout the paper, we write $\overline{u} := (u_2, u_3, \ldots, u_n)^\top \in \mathbb{R}^{n-1}$ for a given n-dimensional vector $u = (u_1, u_2, \ldots, u_n)^\top \in \mathbb{R}^n$. Also, we often identify $(u_1, \overline{u}) \in \mathbb{R} \times \mathbb{R}^{n-1}$ with $\binom{u_1}{\overline{u}} \in \mathbb{R}^n$.

The SOCP has a wide class of applications such as the antenna array weight design problem, the finite impulse response (FIR) design problem, the portfolio optimization with loss risk constraints [5],

*Technical report 2011-015, September 23, 2011. This research was supported in part by Grant-in-Aid for Young Scientists (B) from Japan Society for the Promotion of Science.

†Department of Applied Mathematics and Phisics, Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan. shunhaya@amp.i.kyoto-u.ac.jp (corresponding author)

the magnetic shield design problem for maglev trains [10], and so on. Since the nonnegative orthant is identical with the Cartesian product of one-dimensional SOCs, i.e., $\mathbb{R}^n_+ = \mathcal{K}^1 \times \mathcal{K}^1 \times \cdots \times \mathcal{K}^1$, the linear program (LP) can be regarded as a subclass of SOCP. Moreover, the quadratic program (QP) and a certain class of robust optimization problems can be reformulated as an SOCP. On the contrary, the SOCP is involved in the semidefinite program (SDP) as a subclass. However, it is not reasonable to solve the SOCP as an SDP [14] since the SDP has matrix variables and therefore the computational cost tends to be much higher than that of SOCP.

Currently, the most popular method for solving the SOCP is the primal-dual interior-point method [5, 6, 13], which was originally developed for solving LPs and has been extended to SOCP and SDP. This method is known to be quite efficient both theoretically and practically, and several software packages [12, 11, 15] have been developed. On the other hand, there are only a few studies on the simplex method for SOCP, though it is still important and popular for LP. For example, Muramatsu [7, 8] defined a "dictionary" with respect to the simplex method and proposed an implementable simplex type algorithm for SOCP with some restricted structure. Pataki [9] extended the simplex method for LP to SDP in a theoretical manner. However, his approach has not been implemented to practical problems.

The main reason why the simplex method for SOCP has not been studied so much is that the feasible region of an SOCP has infinitely many extreme points unlike LP. However, the simplex type method has an advantage when we need to solve multiple problems whose structures are similar to each other. In such a case, the simplex type method can be accelerated by using the "hot start" technique, which inherits the information of the bases of the problem solved in the previous step.

In this paper, we first reformulate the SOCP as a linear semi-infinite program (LSIP), and then apply to it the dual-simplex primal-exchange (DSPE) method [4, Chapter 12]. Since any (linear) SOCP can be reformulated as an LSIP, we do not need to restrict the structure of SOCPs. Moreover, in the step of finding the "most violated index" in the DSPE method, we have only to substitute an obtained vector into an explicit formula. This advantage is due to the special structure of the SOC. Indeed, such an index is usually obtained by solving a nonconvex optimization problem when we apply the DSPE method to general LSIPs.

The paper is organized as follows. In Section 2, we give some fundamental background on SOCP and LSIP, and reformulate the SOCP as an LSIP. In Section 3, we introduce the DSPE method for solving such an LSIP, and mention some properties important for implementation. In Section 4, we report some numerical results. Especially, we compare the simplex type algorithm with the existing primal-dual interior-point method, and observe that the simplex type algorithm is often more efficient in solving multiple problems with similar structures successively. In Section 5, we conclude the paper with some remarks.

2 Preliminaries

2.1 Linear semi-infinite program and its duality

We first study the duality theory for LSIP and provide the condition under which the strong duality holds. The LSIP is described as follows:

$$\begin{array}{ll} \underset{x\in\mathbb{R}^m}{\text{minimize}} & c^\top x \\ \text{subject to} & a_t^\top x \geq b_t \ \ (\forall t \in T), \end{array} \tag{2.1}$$

where T is a given compact metric space, $c \in \mathbb{R}^m$ is a given vector, and $a_t = a(t) = (a_1(t), \ldots, a_m(t))^\top$ and $b_t = b(t)$ are continuous mappings from T to $\mathbb{R}^m$ and $\mathbb{R}$, respectively. Originally, the dual problem

of LSIP (2.1) is written as

$$\begin{aligned} \text{maximize} \quad & \int_T b(t)\mu(dt) \\ \text{subject to} \quad & \int_T a(t)\mu(dt) = c \\ & \mu \in W,\ \mu \geq 0, \end{aligned} \tag{2.2}$$

where the decision variable μ is a Borel measure, W is a set of Borel measures on T, and $\mu \geq 0$ denotes that $\mu(T') \geq 0$ for any Borel set $T' \subseteq T$. However, this dual problem is quite difficult to be handled since the variables are not vectors or matrices but a measure. We therefore introduce another type of dual problem called Haar's dual problem.

Let $\mathbb{R}^{(T)}$ and $\mathbb{R}_+^{(T)}$ be the sets of functions from T to $\mathbb{R}$ and $\mathbb{R}_+$, respectively, such that there exist at most finitely many arguments with nonzero functional value, i.e.,

$$\begin{aligned} \mathbb{R}^{(T)} &:= \left\{ \lambda : T \to \mathbb{R} \;\middle|\; |\operatorname{supp} \lambda| < \infty \right\}, \\ \mathbb{R}_+^{(T)} &:= \left\{ \lambda : T \to \mathbb{R}_+ \;\middle|\; |\operatorname{supp} \lambda| < \infty \right\}, \end{aligned}$$

where

$$\operatorname{supp} \lambda := \{\, t \in T \mid \lambda_t \neq 0 \,\}.$$

Then, Haar's dual problem is given as follows:

$$\begin{aligned} \text{maximize} \quad & \sum_{t \in T} \lambda_t b_t \\ \text{subject to} \quad & \lambda \in \mathbb{R}_+^{(T)},\ \sum_{t \in T} \lambda_t a_t = c \end{aligned} \tag{2.3}$$

where $\sum_{t \in T}$ denotes that the sum of all t with $t \in \operatorname{supp} \lambda$. Although the dual problem (2.3) is more restrictive than (2.2), it is known that both optimal value coincide under a general condition [4, Chapter 8]. In what follows, we only focus on Haar's dual problem as a dual of LSIP.

The following proposition provides an optimality condition for LSIP (2.1) and its dual (2.3), and is deeply related to the terminate condition and the optimality of obtained solution for DSPE method mentioned in Section 3.

Proposition 2.1. *Let $x \in \mathbb{R}^n$ and $\lambda \in \mathbb{R}_+^{(T)}$ be feasible points of LSIP (2.1) and its dual (2.3), respectively. Then, they are optimal if*

$$\lambda_t (a_t^\top x - b_t) = 0 \ (\forall t \in T). \tag{2.4}$$

Proof. We readily obtain the result from [4, Theorem 7.1 (i)] and [4, Theorem 7.6 (iii)]. □

2.2 Semi-infinite reformulation of second-order cone program

We next reformulate the SOCP as an LSIP, by using the infinite linear inequality expression of SOC. The following proposition claims that the n-dimensional SOC $\mathcal{K}^n$ can be rewritten by using an infinite number of linear inequalities equivalently.

Proposition 2.2. *Let $\tilde{T} := \{t \in \mathbb{R}^{n-1} \mid \|t\| \leq 1\}$ be an $(n-1)$-dimensional unit ball. Then, $(u_1, \overline{u}) \in \mathcal{K}^n$ if and only if*

$$u_1 \geq t^\top \overline{u} \quad (\forall t \in \tilde{T}). \tag{2.5}$$

Proof. We first show "only if" part by means of contraposition. Suppose that there exists $(u_1, \overline{u}) \in \mathbb{R}^n$ not satisfying (2.5). Then, there exists a $t \in T$ such that $u_1 < t^\top \overline{u}$. We therefore have $u_1 < t^\top \overline{u} \leq \|t^\top \overline{u}\| \leq \|t\| \|\overline{u}\| \leq \|\overline{u}\|$, which implies $(u_1, \overline{u}) \notin \mathcal{K}^n$.

We next show "if" part. Let $(u_1, \overline{u})$ be an arbitrary vector satisfying (2.5). Then, letting $t' := \overline{u}/\|\overline{u}\|$, we have

$$u_1 \geq (t')^\top \overline{u} = \left(\frac{\overline{u}}{\|\overline{u}\|}\right)^\top \overline{u} = \|\overline{u}\|$$

due to $t' \in \tilde{T}$. Hence, $(u_1, \overline{u}) \in \mathcal{K}^n$. □

Now, let us decompose matrix A and vector b in SOCP (1.1) corresponding to the Cartesian structure of $\mathcal{K}$, that is,

$$A = \begin{pmatrix} \begin{pmatrix} (a^1)^\top \\ (A^1)^\top \end{pmatrix} \\ \vdots \\ \begin{pmatrix} (a^p)^\top \\ (A^p)^\top \end{pmatrix} \end{pmatrix}, \qquad b = \begin{pmatrix} \begin{pmatrix} b_1^1 \\ \overline{b}^1 \end{pmatrix} \\ \vdots \\ \begin{pmatrix} b_1^p \\ \overline{b}^p \end{pmatrix} \end{pmatrix}, \tag{2.6}$$

where $a^i \in \mathbb{R}^m$, $A^i \in \mathbb{R}^{m \times (n_i - 1)}$, and $(b_1^i, \overline{b}^i) \in \mathbb{R} \times \mathbb{R}^{n_i - 1}$ $(i = 1, \ldots, p)$. Then we have $Ax + b \in \mathcal{K}$ if and only if

$$\begin{pmatrix} (a^i)^\top x + b_1^i \\ (A^i)^\top x + \overline{b}^i \end{pmatrix} \in \mathcal{K}^{n_i} \qquad (i = 1, \ldots, p).$$

Therefore, by Proposition 2.2, SOCP (1.1) can be rewritten as an LSIP as follows:

$$\begin{array}{ll} \underset{x \in \mathbb{R}^m}{\text{minimize}} & c^\top x \\ \text{subject to} & (a^1 - A^1 t^1)^\top x \geq (t^1)^\top \overline{b}^1 - b_1^1 \ \ (\forall t^1 \in \tilde{T}^1), \\ & \qquad \vdots \\ & (a^p - A^p t^p)^\top x \geq (t^p)^\top \overline{b}^p - b_1^p \ \ (\forall t^p \in \tilde{T}^p), \end{array} \tag{2.7}$$

where

$$\tilde{T}^i := \left\{ t^i \in \mathbb{R}^{n_i - 1} \,\middle|\, \|t^i\| \leq 1 \right\} \tag{2.8}$$

for each $i = 1, 2, \ldots, p$. Moreover, LSIP (2.7) can be written of the form LSIP (2.1) by letting

$$T := \tilde{T}^1 \oplus \tilde{T}^2 \oplus \cdots \oplus \tilde{T}^p, \tag{2.9}$$

$$a_t := a^i - A^i t^i, \tag{2.10}$$

$$b_t := (t^i)^\top \overline{b}^i - b_1^i, \tag{2.11}$$

$$t := (i, t^i), \tag{2.12}$$

where $\oplus$ denotes the direct sum. Notice that any element $t \in T$ is expressed as (2.12) for some $i \in \{1, \ldots, p\}$ and $t^i \in \tilde{T}^i$.

Next we focus on the dual problems of LSIP (2.7) and SOCP (1.1), and study their strong dualities. Note that the dual of SOCP (1.1) can be expressed as

$$\begin{array}{ll} \underset{y \in \mathbb{R}^n}{\text{maximize}} & -b^\top y \\ \text{subject to} & A^\top y = c, \ y \in \mathcal{K}. \end{array} \tag{2.13}$$

Moreover, the optimality condition for SOCP (1.1) and its dual can be given as follows.

Proposition 2.3. *Let $x \in \mathbb{R}^m$ and $y \in \mathbb{R}^n$ be the feasible points for SOCP* (1.1) *and its dual* (2.13)*, respectively. Then, they are optimal if*

$$(Ax + b)^\top y = 0. \tag{2.14}$$

Proof. We readily obtain the proposition by the weak duality theory for SOCP [1, Lemma 12]. □

On the other hand, the dual problem of LSIP (2.7) can be written as

$$\begin{array}{ll} \underset{\lambda^1,\lambda^2,\ldots,\lambda^p}{\text{maximize}} & \displaystyle\sum_{i=1}^{p} \sum_{t^i \in \tilde{T}^i} \lambda^i_{t^i}((t^i)^\top \overline{b}^i - b^i_1) \\ \text{subject to} & \displaystyle\sum_{i=1}^{p} \sum_{t^i \in \tilde{T}^i} \lambda^i_{t^i}(a^i - A^i t^i) = c \\ & \lambda^i \in \mathbb{R}_+^{(\tilde{T}^i)} \ (i = 1, \ldots, p), \end{array} \tag{2.15}$$

which is of the form (2.3) by using (2.9) – (2.12) and

$$\lambda := (\lambda^1, \ldots, \lambda^p) \in \mathbb{R}^{(\tilde{T}^1)} \times \cdots \times \mathbb{R}^{(\tilde{T}^p)} = \mathbb{R}^{(T)}.$$

Moreover, by Proposition 2.1, the optimality condition for LSIP (2.7) and its dual (2.15) can be given as

$$\lambda^i_{t^i} \left\{ (a^i - A^i t^i)^\top x - ((t^i)^\top \overline{b}^i - b^i_1) \right\} = 0 \quad (i = 1, \ldots, p). \tag{2.16}$$

Although SOCP (1.1) and LSIP (2.7) have the same feasible regions and decision variables, their dual problems (2.13) and (2.15) look quite different. In what follows, we study the relation between both feasible points $\lambda = (\lambda^i)_{i=1}^p$ and $y \in \mathbb{R}^n$, which plays an important role in evaluation the accuracy of the algorithm in the subsequent numerical experiments.

Proposition 2.4. *Let $\lambda \in \mathbb{R}^{(T)}$ be any feasible point of dual LSIP* (2.15)*. Then, the vector $y \in \mathbb{R}^n$ defined by*

$$y := \begin{pmatrix} \begin{pmatrix} y^1_1 \\ \overline{y}^1 \end{pmatrix} \\ \begin{pmatrix} y^2_1 \\ \overline{y}^2 \end{pmatrix} \\ \vdots \\ \begin{pmatrix} y^p_1 \\ \overline{y}^p \end{pmatrix} \end{pmatrix} := \begin{pmatrix} \begin{pmatrix} \sum_{t^1 \in \tilde{T}^1} \lambda^1_{t^1} \\ -\sum_{t^1 \in \tilde{T}^1} \lambda^1_{t^1} t^1 \end{pmatrix} \\ \begin{pmatrix} \sum_{t^2 \in \tilde{T}^2} \lambda^2_{t^2} \\ -\sum_{t^2 \in \tilde{T}^2} \lambda^2_{t^2} t^2 \end{pmatrix} \\ \vdots \\ \begin{pmatrix} \sum_{t^p \in \tilde{T}^p} \lambda^p_{t^p} \\ -\sum_{t^p \in \tilde{T}^p} \lambda^p_{t^p} t^p \end{pmatrix} \end{pmatrix} \tag{2.17}$$

is feasible for problem (2.13)*.*

Proof. Let Λ^D and $\lambda \in \Lambda^D$ be a feasible set of (2.15) and its arbitrary feasible point, respectively. Then, the vector $y = (y^i_1, \overline{y}^i)_{i=1}^p$ defined by (2.17) satisfies

$$y^i_1 - \|\overline{y}^i\| = \sum_{t^i \in \tilde{T}^i} \lambda^i_{t^i} - \left\| - \sum_{t^i \in \tilde{T}^i} \lambda^i_{t^i} t^i \right\| \geq \sum_{t^i \in \tilde{T}^i} \lambda^i_{t^i} - \sum_{t^i \in \tilde{T}^i} \lambda^i_{t^i} \|t^i\| \geq 0$$

for each $i = 1, \ldots, p$, where the first inequality is due to the triangle inequality and $\lambda^i_{t^i} \geq 0$, and the last inequality follows from $\|t^i\| \leq 1$. We thus have $y \in \mathcal{K}$. Moreover, the equality condition of

problem (2.15) together with (2.17) yields

$$\begin{aligned} c &= \sum_{i=1}^{p} a^i \sum_{t^i \in \tilde{T}^i} \lambda^i_{t^i} - \sum_{i=1}^{p} A^i \sum_{t^i \in \tilde{T}^i} \lambda^i_{t^i} t^i \\ &= \sum_{i=1}^{p} \left(a^i y^i_1 + A^i \overline{y}^i \right) \;=\; A^\top y. \end{aligned}$$

Therefore, the vector y defined by (2.17) is feasible for (2.13). □

Finally, we show that the vector y with (2.17) is optimal for the dual SOCP (2.13) when x and λ are optimal for (2.7) and (2.15), respectively.

Proposition 2.5. *Suppose that x and λ are feasible for* (2.7) *and* (2.15)*, respectively, and that they satisfy the optimality condition* (2.16)*. Then, x and y defined by* (2.17) *satisfy the SOCP optimality condition* (2.14)*, that is, x and y are optimal for SOCP* (1.1) *and its dual* (2.13)*, respectively.*

Proof. Let x and λ be the feasible solutions of (2.7) and (2.15) satisfying (2.16). Then, we can easily see that x and y defined by (2.17) are feasible for SOCP (1.1) and its dual (2.13), respectively. Moreover, by (2.16), we have

$$((a^i)^\top x + b^i_1)\lambda^i_{t^i} + ((A^i)^\top x + \overline{b}^i)^\top (-\lambda^i_{t^i} t^i) = 0$$

for any $t^i \in T^i$ $(i = 1, 2, \ldots, p)$. Therefore, by (2.17), we obtain

$$((a^i)^\top x + b^i_1) y^i_1 + ((A^i)^\top x + \overline{b}^i)^\top \overline{y}^i = 0,$$

which implies

$$(Ax + b)^\top y = 0.$$

Hence, by Proposition 2.3, x and y are optimal for SOCP (1.1) and its dual (2.13), respectively. □

3 Simplex type algorithm for second-order cone programs

In this section, we study the simplex type algorithm for SOCP (1.1) by applying the DSPE method to LSIP (2.7).

3.1 Dual-simplex primal-exchange method

The DSPE method is based on the simplex method for LPs, where the pivoting in the dual space corresponds to the exchanging of active constraints in the primal space. In the initial step, the DSPE method requires the feasible extreme point for the dual LSIP (2.15). For a given convex set $C \subseteq \mathbb{R}^{(T)}$, the extreme point $c \in C$ is defined as follows[1].

Definition 3.1. *Let $C \subseteq \mathbb{R}^{(T)}$ be a convex set and c be an arbitrary element in C. Then c is said to be an extreme point of C if there* do not *exist $c_1, c_2 \in C \setminus \{c\}$ and $\mu \in (0, 1)$ such that $c = (1-\mu)c_1 + \mu c_2$.*

Denote the feasible set of the dual problem (2.15) by

$$\begin{aligned} \Lambda &:= \left\{ \lambda \in \mathbb{R}^{(T)}_+ \middle| \textstyle\sum_{t \in T} \lambda_t a_t = c \right\} \\ &= \left\{ (\lambda^i)^p_{i=1} \in \mathbb{R}^{(\tilde{T}^1)}_+ \times \cdots \times \mathbb{R}^{(\tilde{T}^p)}_+ \middle| \textstyle\sum_{i=1}^{p} \sum_{t^i \in \tilde{T}^i} \lambda^i_{t^i} (a^i - A^i t^i) = c \right\}. \end{aligned}$$

Then we have the following two properties on the extreme points of Λ.

[1] This definition is a natural extension of the extreme point in $\mathbb{R}^n$.

- $\lambda \in \Lambda$ is an extreme point if and only if all vectors a_t with $t \in \operatorname{supp}\lambda$ are linearly independent [3, Theorem 3.1].
- If an extreme point $\lambda \in \Lambda$ satisfies $|\operatorname{supp}\lambda| = m$, then it is called non-degenerate.

We therefore choose $\lambda^0 \in \mathbb{R}_+^{(T)}$ in the initial step so that the vectors a_t with $t \in \operatorname{supp}\lambda^0$ are linearly independent. Moreover, all the points $\lambda^r \in \mathbb{R}_+^{(T)}$ $(r = 0, 1, 2, \ldots)$ generated by the DSPE method are guaranteed to be extreme points of Λ as will be mentioned in Theorem 3.1. Similarly to the LP simplex method, the pivoting procedure of the DSPE method utilizes the *basic set* derived from $\lambda^r \in \mathbb{R}_+^{(T)}$ in each iteration. Here, $\mathcal{B} \subseteq T$ is called a basic set of an extreme point of $\lambda \in \Lambda$ if it satisfies the following two conditions:

(i) $\mathcal{B} \supseteq \operatorname{supp}\lambda$;

(ii) $\{a_t \,|\, t \in \mathcal{B}\}$ becomes the basis of $\mathbb{R}^m$.

We immediately have $|\mathcal{B}| = m$ from (ii). Generally, the basic set $\mathcal{B}$ is not determined uniquely for a given extreme point $\lambda \in \Lambda$. However, if λ is non-degenerate in addition to (i) and (ii), then $\mathcal{B}$ is determined uniquely and given as $\mathcal{B} = \operatorname{supp}\lambda$. Indeed, as well as the LP case, it is guaranteed that the optimal value decreases strictly for each iteration if the generated extreme points λ^r $(r = 1, 2, \ldots)$ are non-degenerate. This will be shown in Theorem 3.1, later.

Based on the above arguments, we mention the detailed steps of the DSPE method for LSIP (2.7). We assume[2] that λ^r is non-degenerate for $r = 0, 1, 2, \ldots$, and the dimension of $\operatorname{span}\{a_t \,|\, t \in \operatorname{supp}\lambda^r\}$ is always m. We also assume $c \neq 0$ and $\Lambda \neq \emptyset$ for the original problem.

Algorithm 1. (Dual-simplex primal-exchange (DSPE) method)

Step 0 *Choose a feasible extreme point $\lambda^0 \in \Lambda$. Let $\mathcal{B}_0 \subseteq T$ be a basic set of λ^0, and $\mathcal{N}_0 := T \setminus \mathcal{B}_0$ be the corresponding non-basic set. Set $r := 0$.*

Step 1 *Solve the linear equation $a_t^\top x = b_t\,(t \in \mathcal{B}_r)$, and let x_r be its (unique) solution.*

Step 2 *Let $t_{\mathrm{in}}^r := \operatorname{argmin}_{t\in T}(a_t^\top x^r - b_t)$. If $a_{t_{\mathrm{in}}^r}^\top x^r - b_{t_{\mathrm{in}}^r} \geq 0$, then terminate. Otherwise, go to the next step.*

Step 3 *Let $g^r \in \mathbb{R}^{(T)}$ be such that $\operatorname{supp} g^r \subseteq \mathcal{B}_r$ and $\sum_{t\in\mathcal{B}_r} g_t^r a_t = a_{t_{\mathrm{in}}^r}$.*

Step 4 *If $-g^r \in \mathbb{R}_+^{(T)}$, then terminate since the objective function is unbounded. Otherwise, let*

$$\mu_r := \min_{t\in\mathcal{B}_r, g_t^r>0} \{\lambda_t^r / g_t^r\},$$
$$t_{\mathrm{out}}^r := \operatorname*{argmin}_{t\in\mathcal{B}_r, g_t^r>0} \{\lambda_t^r / g_t^r\}.$$

Step 5 *Define λ^{r+1}, $\mathcal{B}_{r+1}$ and $\mathcal{N}_{r+1}$ by*

$$\lambda_t^{r+1} := \begin{cases} \lambda_t^r - \mu_r g_t^r & (t \in \mathcal{B}_r \setminus \{t_{\mathrm{out}}^r\}) \\ 0 & (t \in \mathcal{N}_r \setminus \{t_{\mathrm{in}}^r\}) \\ 0 & (t = t_{\mathrm{out}}^r) \\ \mu_r & (t = t_{\mathrm{in}}^r) \end{cases}$$
$$\mathcal{B}_{r+1} := \mathcal{B}_r \cup \{t_{\mathrm{in}}^r\} \setminus \{t_{\mathrm{out}}^r\},$$
$$\mathcal{N}_{r+1} := \mathcal{N}_r \cup \{t_{\mathrm{out}}^r\} \setminus \{t_{\mathrm{in}}^r\}.$$

Let $r := r + 1$ and return to Step 1.

[2]This assumption is necessary for the existence of a non-generate extreme point.

In Steps 1 and 3, x^r and g^r can be calculated by means of a classical technique such as the Gaussian elimination under the non-degeneracy assumption. In Step 2, it is not easy to find the "most violated index" $t^r_{\rm in} \in T$ when we try to solve a general LSIP. However, in case of LSIP (2.7), we can easily obtain such an index explicitly by using the special structure of $\tilde{T}^i$. Since $\tilde{T}^i$ is an (n^i-1)-dimensional unit ball, we easily have

$$\begin{aligned} t^{r,i}_{\rm min} &:= \operatorname*{argmin}_{t^i \in \tilde{T}^i} \left\{ (a^i - A^i t^i)^\top x^r - (t^i)^\top \overline{b}^i + b^i_1 \right\} \\ &= \operatorname*{argmin}_{t^i \in \tilde{T}^i} \left\{ (t^i)^\top (-(A^i)^\top x^r - \overline{b}^i) + (a^i)^\top x^r + b^i_1 \right\} \\ &= \frac{(A^i)^\top x^r + \overline{b}^i}{\|(A^i)^\top x^r + \overline{b}^i\|}, \\ v^{r,i}_{\rm min} &:= \min_{t^i \in \tilde{T}^i} \left\{ (a^i - A^i t^i)^\top x^r - (t^i)^\top \overline{b}^i + b^i_1 \right\} \\ &= -\|(A^i)^\top x^r + \overline{b}^i\| + (a^i)^\top x^r + b^i_1 \end{aligned}$$

for each $i = 1, 2, \ldots, p$. Therefore, letting

$$\bar{i}_r := \operatorname{argmin} \left\{ v^{r,i}_{\rm min} \;\middle|\; i = 1, \ldots, p \right\} \tag{3.1}$$

$$t^r_{\rm in} := (\bar{i}_r, t^{r,\bar{i}_r}_{\rm min}) \tag{3.2}$$

we have $t^r_{\rm in} = \operatorname{argmin}_{t \in T} \{a_t^\top x^r - b_t\}$. If we have multiple candidates for $\bar{i}_r$ in (3.1), then we can choose the smallest index. Moreover, on the termination criterion in each iteration, we have the following theorem.

Theorem 3.1. [4, Theorem 12.2] *Let $x^r \in \mathbb{R}^m$ and $\lambda^r \in \mathbb{R}^{(T)}_+$ be the r-th iterative points for Algorithm 1. Then, the following three statements hold.*

(i) *If the iteration terminates in Step 2, then x^r and λ^r are optimal for (2.7) and (2.15), respectively.*

(ii) *If the iteration terminates in Step 4, then the dual problem (2.15) is unbounded and the primal problem (2.7) is infeasible.*

(iii) *If the termination criterions in Steps 2 and 4 do not hold at the r-th iteration, then λ^{r+1} is an extreme point of Λ satisfying*

$$\sum_{t \in T} \lambda^r_t b_t < \sum_{t \in T} \lambda^{r+1}_t b_t.$$

3.2 Two-phase method

In the simplex method for LP, it is well known that the two-phase method is available to find an initial feasible point [2]. The similar technique can be applied to the DSPE method in finding a feasible extreme point $\lambda_0 \in \Lambda$. Consider the following auxiliary problem for LSIP (2.15):

$$\begin{aligned} \underset{\lambda^1, \ldots, \lambda^p, \tilde{\lambda}}{\text{minimize}} \quad & \mathbf{1}^\top \tilde{\lambda} \\ \text{subject to} \quad & \sum_{i=1}^{p} \sum_{t^i \in \tilde{T}^i} \lambda^i_{t^i} (a^i - A^i t^i) + \tilde{\lambda} = c, \\ & \lambda^i \in \mathbb{R}^{(T_i)}_+ \ (i = 1, \ldots, p), \ \ \tilde{\lambda} \geq 0, \end{aligned} \tag{3.3}$$

where $\tilde{\lambda} \in \mathbb{R}^m$ is an additional variable and $\mathbf{1} := (1, 1, \ldots, 1)^\top \in \mathbb{R}^m$. Notice that we can assume $c \geq 0$ without loss of generality.[3] Therefore, $(\lambda^1, \ldots, \lambda^p, \tilde{\lambda}) = (0, \ldots, 0, c) \in \mathbb{R}^{(T_1)}_+ \times \cdots \times \mathbb{R}^{(T_p)}_+ \times \mathbb{R}^m_+$ becomes the self-evident feasible extreme point of problem (3.3). Moreover, we have the following theorem.

[3] If $c_k < 0$ for some k, then we have only to multiply the k-th equality constraint in (2.15) by -1.

Theorem 3.2. [3, Theorem 6.1] *Let v_a be the optimal value of problem* (3.3)*. Then,*

- $v_a \geq 0$ *if* (3.3) *is feasible;*
- (2.15) *is infeasible if $v_a > 0$;*
- *the basic set at the optimum of* (3.3) *becomes a basic set of the original problem* (2.15)*, if $v_a = 0$.*

Therefore, if the optimum $(\lambda^{1,*}, \ldots, \lambda^{p,*}, \tilde{\lambda}^*)$ of problem (3.3) satisfies $\tilde{\lambda}^* = 0$, then we can choose $(\lambda^{1,*}, \ldots, \lambda^{p,*}) \in \Lambda$ as a feasible extreme point of problem (2.15). As a result, the two-phase method can be summarized as follows.

Two-phase DSPE method

Step 1 Solve the auxiliary problem (3.3) by the DSPE method with the initial feasible extreme point $(0, \ldots, 0, c) \in \mathbb{R}_+^{(T_1)} \times \cdots \times \mathbb{R}_+^{(T_p)} \times \mathbb{R}_+^m$. If the optimal value is 0, then go to Step 2. Otherwise, terminate the algorithm since (2.15) is infeasible.

Step 2 Let $\lambda^0 := (\lambda^{1,*}, \ldots, \lambda^{p,*}) \in \mathbb{R}_+^{(T_1)} \times \cdots \times \mathbb{R}_+^{(T_p)} = \mathbb{R}_+^{(T)}$, where $(\lambda^{1,*}, \ldots, \lambda^{p,*}, \tilde{\lambda}^*) \in \mathbb{R}_+^{(T_1)} \times \cdots \times \mathbb{R}_+^{(T_p)} \times \mathbb{R}_+^m$ is the optimum of problem (3.3). Then, apply Algorithm 1 with $\lambda^0 \in \Lambda$ in Step 0.

4 Numerical results

In this section, we report some numerical results to see the efficiency of the SDPE method provided in Section 3. The following three statements apply to all experiments.

- We generate test problems of the form SOCP (1.1) as follows. We first generate matrix $A \in \mathbb{R}^{n \times m}$ and vectors $\tilde{b}^i \in \mathbb{R}^{n_i}$ $(i = 1, \ldots, p)$ and $c \in \mathbb{R}^m$ so that their all components are randomly chosen from $[-1, 1]$. Then, we let $b := ((b^1)^\top, \ldots, (b^p)^\top)^\top \in \mathbb{R}^{n_1 + \cdots + n_p} = \mathbb{R}^n$ with $b^i := (\alpha_i, \tilde{b}_2^i, \ldots, \tilde{b}_{n_i}^i)^\top$ and $\alpha_i := (1 + r)\|(\tilde{b}_2^i, \ldots, \tilde{b}_p^i)\|$, where $r \in \mathbb{R}$ is randomly chosen from $(0, 1)$. Hereby SOCP (1.1) is guaranteed to be feasible at $x = 0$ since $b^i \in \mathcal{K}^{n_i}$ for each i. If the generated test problem is unbounded, then it is discarded and another problem is generated in the same way.
- In implementing Algorithm 1, we relax the termination criterion in Step 2 from $a_{t_{\rm in}^r}^\top x^r - b_{t_{\rm in}^r} \geq 0$ to $a_{t_{\rm in}^r}^\top x^r - b_{t_{\rm in}^r} \geq -10^{-8}$.
- The programs are coded by Matlab 7.4 and run on a machine with Pentium (R) CPUs (3.2 GHz $\times$2) and 2GB RAM.

4.1 Comparison to existing software: solving a single SOCP

Experiment 1

We first solve SOCP (1.1) with various choices of the Cartesian structure $\mathcal{K}$. For comparison purpose, we solve the generated SOCP not only by the DSPE method but also by SDPT3 solver [12], which is existing software based on the primal-dual interior-point method. We generate 10 test problems for each Cartesian structure $\mathcal{K}$.

Table 1 shows the obtained result, in which $\sharp$pivot denotes the number of pivot steps, and $e(x^*, y^*)$ is the nonnegative value defined by

$$e(x^*, y^*) := \mathrm{dist}\,(Ax^* + b, \mathcal{K}) + \mathrm{dist}\,(y^*, \mathcal{K}) + |(Ax^* + b)^\top y^*| + \|A^\top y^* - c\|,$$

where (x^*, y^*) is the solution obtained by the algorithms.[4] Notice that $e(x^*, y^*)$ evaluates the accuracy of the obtained solution. Indeed we can easily see that $e(x, y) \geq 0$ for any $(x, y) \in \mathbb{R}^m \times \mathbb{R}^n$, and $e(x, y) = 0$ if and only if x and y solve the primal SOCP (1.1) and the dual SOCP (2.13), respectively.

From the table, we can see that the accuracy of the obtained solutions by the DSPE method is sufficiently high compared with SDPT3. Moreover, the DSPE method tends to find the solution faster than SDPT3 as the dimension m of the decision variable x is small. However, as m becomes larger, the CPU time and the number of pivot steps of the DSPE method increase drastically, and for such problems, SDPT3 works much better than the DSPE method. On the other hand, the degree of increment for those values is milder even when the value of n (dimension of $\mathcal{K}$) increases.

4.2 Solving multiple SOCPs with similar structures

We next solve multiple test problems (10 problems) with similar structures successively for each trial. We also apply the DSPE method and SDPT3 solver for solving test problems. However, in applying the DSPE method (Algorithm 1), we determine the initial feasible extreme point in Step 0 by taking advantage of the basic set in the problem solved previously.

Experiment 2

We first consider the sequence of problems

$$\mathcal{L} = \{\mathrm{SOCP}(b^k)\}_{k=1}^{10},$$

where A and c are fixed and b is varied as $b = b^1, b^2, \ldots, b^{10}$. Here, b^1 is generated so that all components are randomly chosen from $[-1, 1]$, and b^k $(k = 2, \ldots, 10)$ are generated as

$$b^k := b^{k-1} + \frac{\delta \|b^{k-1}\|}{\sqrt{n}} u^n, \tag{4.1}$$

where $\delta > 0$ is a fixed small number and $u^n \in \mathbb{R}^n$ is the vector whose components are randomly chosen from $[-1, 1]$. We choose 3 different values and 8 different Cartesian structures for δ and $\mathcal{K}$, respectively. Moreover, for each $(\delta, \mathcal{K})$, we generate 10 families of problems $\mathcal{L} = \mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_{10}$. Therefore, we solve 2400 SOCPs in total.

In applying the SDPE method to $\mathcal{L} = \{\mathrm{SOCP}(b^k)\}_{k=1}^{10}$, we use the two-phase method only for $\mathrm{SOCP}(b^1)$. For solving $\mathrm{SOCP}(b^k)$ $(k = 2, \ldots, 10)$, the initial basic set and feasible extreme point are set to be the optimal basic set and the dual optimum[5] of $\mathrm{SOCP}(b^{k-1})$, respectively. We notice that the feasible region of LSIP (2.15) does not change even if the value of b changes.

The obtained results are shown in Tables 2–4, each of which corresponds to the case where $\delta = 10^{-4}, 10^{-5}, 10^{-6}$, respectively. Moreover, in those tables, $\mathrm{CPU}_{\mathrm{first}}$ and $\mathrm{CPU}_{\mathrm{total}}$ denote the CPU time for solving the initial problem $\mathrm{SOCP}(b^1)$, and the total CPU time for solving 10 problems $\mathrm{SOCP}(b^1)$, $\mathrm{SOCP}(b^2), \ldots, \mathrm{SOCP}(b^{10})$ for each family of problems $\mathcal{L}$, respectively. Each value of the tables is an average for 10 families $\mathcal{L} = \mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_{10}$ for each $(\delta, \mathcal{K})$.

As is observed from the tables, the value of $\mathrm{CPU}_{\mathrm{first}}/\mathrm{CPU}_{\mathrm{total}}$ for the DSPE method is much larger than 0.1. This implies that the computational cost for solving $\mathrm{SOCP}(b^k)$ $(k \geq 2)$ is saved considerably due to the hot start technique. Especially, when $\delta = 10^{-5}$ and 10^{-6}, most of computational cost for solving the 10 SOCPs is caused by the two-phase method for $\mathrm{SOCP}(b^1)$. On the other hand, for SDPT3, the CPU time required for solving $\mathrm{SOCP}(b^k)$ does not depend on k so much, and therefore the value of $\mathrm{CPU}_{\mathrm{first}}/\mathrm{CPU}_{\mathrm{total}}$ is around 0.1.

[4] For the DSPE method, y^* is calculated by means of formula (2.17).

[5] In this case, the dual optimum of $\mathrm{SOCP}(b^k)$ means the optimum λ^* of the equivalent dual LSIP (2.15). This applies to $\mathrm{SOCP}(A^k)$ and $\mathrm{SOCP}(c^k)$ in Experiments 3 and 4.

Experiment 3

Next we consider the case where A and b are fixed and c is varied as $c = c^1, c^2, \ldots, c^{10}$. Similarly to Experiment 2, we generate the sequence of problems $\mathcal{L}$ as follows:

$$\begin{aligned} c^1 &:= u^m, \\ c^k &:= c^{k-1} + \frac{\delta \|c^{k-1}\|}{\sqrt{m}} u^m \ \ (k = 2, 3, \ldots, 10), \\ \mathcal{L} &= \{\mathrm{SOCP}(c^k)\}_{k=1}^{10}, \end{aligned}$$

where $\delta > 0$ is a fixed small number and $u^m \in \mathbb{R}^m$ is the vector whose components are randomly chosen from $[-1, 1]$. We choose 3 different values and 8 different Cartesian structures for δ and $\mathcal{K}$, respectively, and generate 10 families of problems $\mathcal{L} = \mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_{10}$ for each $(\delta, \mathcal{K})$.

Unlike Experiment 2, we may not be able to use the dual optimum of $\mathrm{SOCP}(c^{k-1})$ as the initial feasible extreme point for $\mathrm{SOCP}(c^k)$, since the dual feasible point[6] of $\mathrm{SOCP}(c^{k-1})$ may not be feasible for $\mathrm{SOCP}(c^k)$. We therefore generate the initial feasible extreme point of $\mathrm{SOCP}(c^k)$ as follows.

> Let $\mathcal{B}^*_{k-1}$ be the optimal basic set for $\mathrm{SOCP}(c^{k-1})$. We first calculate $\tilde{\lambda} \in \mathbb{R}^{(T)}$ such that $\mathrm{supp}\,\tilde{\lambda} = \mathcal{B}^*_{k-1}$ and the equality constraint of the dual LSIP (2.15) with $c = c^k$ holds. (To this end, we have only to solve an m-dimensional linear equation.) If the nonnegativity condition $\tilde{\lambda} \in \mathbb{R}^{(T)}_+$ is satisfied, then we set $\tilde{\lambda}$ to be the initial feasible extreme point for $\mathrm{SOCP}(c^k)$. (In this case, $\mathcal{B}^*_{k-1}$ becomes the initial basic set.) If $\tilde{\lambda} \notin \mathbb{R}^{(T)}_+$, then we apply the two-phase method.

The obtained results are shown in Tables 5–7, each of which corresponds to the case where $\delta = 10^{-4}, 10^{-5}, 10^{-6}$, respectively. In those tables, $\mathrm{CPU}_{\mathrm{first}}$ and $\mathrm{CPU}_{\mathrm{total}}$ are defined in the same way as Experiment 2, and "$\sharp$ hot start" denotes the number of times that we succeeded in the hot start, i.e., we could inherit the basic set from the previous problem.[7] Those values are also the average for 10 families of problems $\mathcal{L} = \mathcal{L}_1, \ldots, \mathcal{L}_{10}$ for each $(\delta, \mathcal{K})$. As the tables show, the value of $\mathrm{CPU}_{\mathrm{total}}$ tends to smaller than SDPT3 when $\delta = 10^{-6}$, since the basic sets were inherited more often. Particularly, when $\mathcal{K}$ consists of multiple SOCs, this tendency is more remarkable. However, when $\delta = 10^{-5}$ or 10^{-4}, the basic sets often failed to be inherited, and hence the total CPU time tends to larger than the case of $\delta = 10^{-6}$.

Experiment 4

In the final experiment, vectors b and c are fixed, and matrix A is varied as $A = A^1, A^2, \ldots, A^{10}$. Similarly to Experiments 2 and 3, we generate the sequence of problems $\mathcal{L}$ as follows:

$$\begin{aligned} A^1 &:= \text{an } m \times m \text{ matrix whose components are randomly chosen from } [-1, 1], \\ A^k &:= A^{k-1} D \ (k = 2, 3, \ldots, 10), \\ \mathcal{L} &= \{\mathrm{SOCP}(A^k)\}_{k=1}^{10}, \end{aligned}$$

where D is an $m \times m$ diagonal matrix whose diagonal entries are randomly chosen from $[1 - \delta, 1 + \delta]$ with a given positive number $\delta > 0$. We choose 3 different values and 8 different Cartesian structures for δ and $\mathcal{K}$, respectively, and generate 10 families of problems $\mathcal{L} = \mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_{10}$ for each $(\delta, \mathcal{K})$. In this experiment, we generate the initial feasible extreme point as follows, where a^k_t denotes the value of a_t with $A := A^k$. (a_t is defined by (2.10).)

[6] Here, the dual feasible point of $\mathrm{SOCP}(c^k)$ means the feasible point λ for the equivalent dual LSIP (2.15). This applies to $\mathrm{SOCP}(A^k)$ in Experiment 5.

[7] Notice that the value of $\sharp$ hot start is at most 9 since we have to use the two-phase method for solving $\mathrm{SOCP}(c^1)$.

Let $\mathcal{B}^*_{k-1}$ be the optimal basic set for SOCP(A^{k-1}). If a^k_t ($t \in \mathcal{B}^*_{k-1}$) are linearly dependent, then we solve SOCP(A^k) by the two-phase method. If a^k_t ($t \in \mathcal{B}^*_{k-1}$) are linearly independent, then we calculate $\tilde{\lambda} \in \mathbb{R}^{(T)}$ such that $\operatorname{supp} \tilde{\lambda} = \mathcal{B}^*_{k-1}$ and the equality constraint of the dual LSIP (2.15) with $A = A^k$ holds. If $\tilde{\lambda} \in \mathbb{R}^{(T)}_+$, then we set $\tilde{\lambda}$ to be the initial feasible extreme point for SOCP(A^k). If $\lambda \notin \mathbb{R}^{(T)}_+$, then we apply the two-phase method.

The obtained results are shown in Tables 8–10, each of which corresponds to the case where $\delta = 10^{-4}, 10^{-5}, 10^{-6}$, respectively. In the tables, the meanings of $\mathrm{CPU}_{\mathrm{total}}$, $\mathrm{CPU}_{\mathrm{first}}$ and $\sharp$hot start are the same as Experiment 3. As the tables show, the basic sets tend to be inherited more often when $\delta = 10^{-6}$. Especially, when $\mathcal{K}$ consists of multiple SOCs, this tendency is more remarkable. We expect that this is because the degree of linear independence of the vectors a^k_t ($t \in \mathcal{B}^*_{k-1}$) is stronger.[8] However, in case of $\delta = 10^{-5}$ or 10^{-4}, such a hot start technique tends to be unsuccessful.

5 Final remarks

In this paper, we reformulated an SOCP as an LSIP and studied their strong duality. Moreover, we proposed the simplex type algorithm for SOCP, by applying the DSPE method for the reformulated LSIP. We also solved a number of test problems, and observed that the DSPE method is advantageous when we solve multiple SOCPs with similar structures. We expect that the simplex type approach studied in this paper will be useful, when it is applied to the algorithms that are required to solve multiple SOCPs successively as subproblems.

References

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

[2] Chvátal, V.: *Linear programming*, WH Freeman, 1983.

[3] Goberna, M. A. and Jornet, V.: Geometric Fundamentals of the Simplex Method in Semi-Infinite Programming, *OR Spektrum*, Vol. 10 (1988), 145–152.

[4] Goberna, M. A. and López, M. A.: *Linear Semi-Infinite Optimization*, John Wiley and Sons Ltd., 1998.

[5] Lobo, M. S., Vandenberghe, L., Boyd, S. and Lebret, H.: Applications of second-order cone programming, *Linear Algebra and Its Applications*, Vol. 284 (1998), 193–228.

[6] Monteiro, R. and Tsuchiya, T.: Polynomial convergence of primal-dual algorithms for the second-orderone program based on the MZ-family of directions, *Mathematical Programming*, Vol. 88 (2000), 61–83.

[7] Muramatsu, M.: A pivoting procedure for a class of second-order cone programming, *Optimization Methods and Software*, Vol. 21 (2005), 295–315.

[8] Muramatsu, M.: Towards a pivoting procedure for a class of second-cone programming problems having multiple cone constraints, *Pacific Journal of Optimization*, Vol. 3 (2007), 87–97.

[8] This means that the full-rank property is maintained even if the square matrix with column vectors a^k_t ($t \in \mathcal{B}^*_{k-1}$) is perturbed.

[9] Pataki, G.: Cone-LP's and semidefinite programs: geometry and a simplex-type method, in *Integer Programming and Combinatorial Optimization*, Vol. 1084 of *Lecture notes in computer science*, Springer Berlin, Heidelberg, 1996, 162–174.

[10] Sasakawa, T. and Tsuchiya, T.: Optimal Magnetic Shield Design with Second-Order Cone Programming, *SIAM Journal on Scientific Computing*, Vol. 24 (2003), 1930–1950.

[11] Sturm, J. F.: *Using SeDuMi 1.02, a matlab toolbox for optimization over symmetric cones* (*Updated for Version 1.05*), `http://sedumi.ie.lehigh.edu/`, 2001.

[12] Toh, K. C., Tütüncü, R. H. and Todd, M. J.: *SDPT3 version 4.0 – a matlab software for semidefinite-quadratic-linear programming*, `http://www.math.nus.edu.sg/~mattohkc/sdpt3.html`, 2006.

[13] Tsuchiya, T.: A convergence analysis of the scaling-invariant primal-dual path-following algorithms for second-order cone programming, *Optimization Methods and Software*, Vol. 11 (1999), 141–182.

[14] Vandenberghe, L. and Boyd, S.: Semidefinite programming, *SIAM Review*, Vol. 38 (1996), 49–95.

[15] Yamashita, M., Fujisawa, K., Fukuda, M., Nakata, K. and Nakata, M.: A high-performance software package for semidefinite programs: SDPA 7, Research report B-463, Dept. of Mathematical and Computing Science, Tokyo Institute of Technology, Tokyo, Japan, 2010.

Table 1: Result of Experiment 1

m	n	$\mathcal{K}$	CPU time (s)		♯pivot	$e(x^*, y^*)$	
			SDPT3	DSPE	DSPE	SDPT3	DSPE
5	10	$\mathcal{K}^{10}$	0.136	0.030	105.5	1.326e-08	3.681e-09
5	50	$(\mathcal{K}^{10})^5$	0.791	0.042	73.6	1.146e-08	3.803e-09
5	100	$(\mathcal{K}^{10})^{10}$	1.646	0.046	62.8	1.292e-08	3.543e-09
5	100	$\mathcal{K}^{100}$	0.136	0.030	108.0	1.260e-08	3.748e-09
5	500	$(\mathcal{K}^{100})^5$	0.800	0.059	102.5	1.292e-08	4.120e-09
5	1000	$(\mathcal{K}^{100})^{10}$	1.788	0.091	109.6	1.124e-08	3.752e-09
5	200	$\mathcal{K}^{200}$	0.132	0.033	108.8	2.081e-08	3.887e-09
5	1000	$(\mathcal{K}^{200})^5$	0.951	0.069	106.2	1.309e-08	4.260e-09
5	2000	$(\mathcal{K}^{200})^{10}$	1.896	0.103	112.9	1.123e-08	4.113e-09
10	100	$\mathcal{K}^{100}$	0.143	0.172	525.6	9.254e-09	4.166e-09
10	500	$(\mathcal{K}^{100})^5$	0.909	0.351	496.7	1.556e-08	4.337e-09
10	1000	$(\mathcal{K}^{100})^{10}$	1.962	0.453	487.3	1.312e-08	4.221e-09
10	200	$\mathcal{K}^{200}$	0.152	0.184	547.1	1.138e-08	4.450e-09
10	1000	$(\mathcal{K}^{200})^5$	0.904	0.384	524.1	1.871e-08	4.227e-09
10	2000	$(\mathcal{K}^{200})^{10}$	1.943	0.490	486.2	1.577e-08	4.420e-09
20	100	$\mathcal{K}^{100}$	0.150	1.665	3717.8	1.541e-08	4.629e-09
20	500	$(\mathcal{K}^{100})^5$	1.048	3.237	3398.0	2.331e-08	4.323e-09
20	1000	$(\mathcal{K}^{100})^{10}$	2.271	3.919	3314.0	1.546e-08	4.542e-09
20	200	$\mathcal{K}^{200}$	0.158	1.713	3686.1	8.613e-09	4.732e-09
20	1000	$(\mathcal{K}^{200})^5$	1.130	3.772	3690.5	2.197e-08	4.424e-09
20	2000	$(\mathcal{K}^{200})^{10}$	2.208	4.541	3529.4	2.207e-08	4.588e-09
50	100	$\mathcal{K}^{100}$	0.266	174.705	107254.8	4.069e-08	4.759e-09
50	500	$(\mathcal{K}^{100})^5$	1.279	134.420	69106.3	3.694e-08	4.713e-09
50	1000	$(\mathcal{K}^{100})^{10}$	2.808	146.180	67093.8	3.696e-08	4.775e-09
50	200	$\mathcal{K}^{200}$	0.269	167.010	100891.9	2.216e-08	4.785e-09
50	1000	$(\mathcal{K}^{200})^5$	1.556	179.963	88713.5	3.796e-08	4.824e-09
50	2000	$(\mathcal{K}^{200})^{10}$	3.629	220.170	85746.6	2.469e-08	4.678e-09

Table 2: Result of Experiment 2 ($\delta = 10^{-6}$)

m	$\mathcal{K}$	$\text{CPU}_{\text{total}}$		$\text{CPU}_{\text{first}}$		$\text{CPU}_{\text{first}}/\text{CPU}_{\text{total}}$	
		SDPT3	DSPE	SDPT3	DSPE	SDPT3	DSPE
10	$\mathcal{K}^{20}$	1.444	0.185	0.215	0.157	0.149	0.849
10	$(\mathcal{K}^{20})^{10}$	14.254	0.223	1.511	0.210	0.106	0.942
10	$\mathcal{K}^{100}$	1.480	0.217	0.146	0.173	0.099	0.797
10	$(\mathcal{K}^{100})^{10}$	17.869	0.432	1.773	0.410	0.099	0.949
20	$\mathcal{K}^{40}$	1.852	1.849	0.188	1.804	0.102	0.976
20	$(\mathcal{K}^{40})^{10}$	19.906	2.480	1.983	2.453	0.100	0.989
20	$\mathcal{K}^{100}$	1.951	1.937	0.198	1.906	0.101	0.984
20	$(\mathcal{K}^{100})^{10}$	22.181	3.659	2.223	3.625	0.100	0.991

Table 3: Result of Experiment 2 ($\delta = 10^{-5}$)

m	$\mathcal{K}$	$\text{CPU}_{\text{total}}$		$\text{CPU}_{\text{first}}$		$\text{CPU}_{\text{first}}/\text{CPU}_{\text{total}}$	
		SDPT3	DSPE	SDPT3	DSPE	SDPT3	DSPE
10	$\mathcal{K}^{20}$	1.812	0.233	0.181	0.196	0.100	0.841
10	$(\mathcal{K}^{20})^{10}$	18.178	0.364	1.806	0.308	0.099	0.846
10	$\mathcal{K}^{100}$	1.887	0.293	0.190	0.212	0.101	0.724
10	$(\mathcal{K}^{100})^{10}$	19.948	0.592	1.996	0.471	0.100	0.796
20	$\mathcal{K}^{40}$	2.321	2.043	0.229	2.007	0.099	0.982
20	$(\mathcal{K}^{40})^{10}$	20.771	3.017	2.069	2.677	0.100	0.887
20	$\mathcal{K}^{100}$	2.094	2.241	0.214	1.946	0.102	0.868
20	$(\mathcal{K}^{100})^{10}$	23.145	4.253	2.303	3.534	0.100	0.831

Table 4: Result of Experiment 2 ($\delta = 10^{-4}$)

m	$\mathcal{K}$	$\text{CPU}_{\text{total}}$		$\text{CPU}_{\text{first}}$		$\text{CPU}_{\text{first}}/\text{CPU}_{\text{total}}$	
		SDPT3	DSPE	SDPT3	DSPE	SDPT3	DSPE
10	$\mathcal{K}^{20}$	2.039	0.361	0.206	0.198	0.101	0.548
10	$(\mathcal{K}^{20})^{10}$	17.299	0.876	1.729	0.343	0.100	0.392
10	$\mathcal{K}^{100}$	1.909	0.753	0.188	0.215	0.098	0.286
10	$(\mathcal{K}^{100})^{10}$	19.788	1.524	1.978	0.455	0.100	0.299
20	$\mathcal{K}^{40}$	2.233	3.477	0.233	1.984	0.104	0.571
20	$(\mathcal{K}^{40})^{10}$	19.841	6.630	1.972	2.534	0.099	0.382
20	$\mathcal{K}^{100}$	1.996	6.281	0.199	1.967	0.100	0.313
20	$(\mathcal{K}^{100})^{10}$	21.706	11.092	2.181	3.730	0.100	0.336

Table 5: Result of Experiment 3 ($\delta = 10^{-6}$)

m	$\mathcal{K}$	$\text{CPU}_{\text{total}}$		$\sharp$ hot start	$\text{CPU}_{\text{first}}$		$\text{CPU}_{\text{first}}/\text{CPU}_{\text{total}}$	
		SDPT3	DSPE	DSPE	SDPT3	DSPE	SDPT3	DSPE
10	$\mathcal{K}^{20}$	1.458	0.177	9.0	0.151	0.162	0.104	0.915
10	$(\mathcal{K}^{20})^{10}$	13.469	0.204	9.0	1.352	0.185	0.100	0.907
10	$\mathcal{K}^{100}$	1.451	0.233	8.7	0.144	0.172	0.099	0.738
10	$(\mathcal{K}^{100})^{10}$	16.483	0.443	8.7	1.636	0.325	0.099	0.734
20	$\mathcal{K}^{40}$	1.984	8.370	6.3	0.201	2.298	0.101	0.275
20	$(\mathcal{K}^{40})^{10}$	19.654	4.458	8.0	1.877	2.078	0.096	0.466
20	$\mathcal{K}^{100}$	1.902	7.638	6.8	0.190	2.393	0.100	0.313
20	$(\mathcal{K}^{100})^{10}$	21.832	7.655	7.7	2.182	3.241	0.100	0.423

Table 6: Result of Experiment 3 ($\delta = 10^{-5}$)

m	$\mathcal{K}$	$\text{CPU}_{\text{total}}$		$\sharp$ hot start	$\text{CPU}_{\text{first}}$		$\text{CPU}_{\text{first}}/\text{CPU}_{\text{total}}$	
		SDPT3	DSPE	DSPE	SDPT3	DSPE	SDPT3	DSPE
10	$\mathcal{K}^{20}$	2.074	1.792	2.9	0.208	0.260	0.100	0.145
10	$(\mathcal{K}^{20})^{10}$	17.571	0.816	6.8	1.748	0.261	0.099	0.320
10	$\mathcal{K}^{100}$	1.933	1.254	5.1	0.200	0.255	0.103	0.203
10	$(\mathcal{K}^{100})^{10}$	20.593	1.277	7.0	2.057	0.410	0.100	0.321
20	$\mathcal{K}^{40}$	1.958	24.889	0.0	0.196	2.476	0.100	0.099
20	$(\mathcal{K}^{40})^{10}$	21.626	18.019	2.4	2.164	2.298	0.100	0.128
20	$\mathcal{K}^{100}$	2.054	25.611	0.0	0.212	2.623	0.103	0.102
20	$(\mathcal{K}^{100})^{10}$	22.934	31.623	0.8	2.279	3.407	0.099	0.108

Table 7: Result of Experiment 3 ($\delta = 10^{-4}$)

m	$\mathcal{K}$	$\text{CPU}_{\text{total}}$		$\sharp$hot start	$\text{CPU}_{\text{first}}$		$\text{CPU}_{\text{first}}/\text{CPU}_{\text{total}}$	
		SDPT3	DSPE	DSPE	SDPT3	DSPE	SDPT3	DSPE
10	$\mathcal{K}^{20}$	1.992	2.370	0.8	0.198	0.265	0.099	0.112
10	$(\mathcal{K}^{20})^{10}$	17.326	2.541	1.0	1.722	0.276	0.099	0.109
10	$\mathcal{K}^{100}$	2.023	2.625	0.0	0.204	0.259	0.101	0.099
10	$(\mathcal{K}^{100})^{10}$	20.931	4.135	0.2	2.083	0.428	0.100	0.104
20	$\mathcal{K}^{40}$	2.288	26.767	0.0	0.218	2.598	0.095	0.097
20	$(\mathcal{K}^{40})^{10}$	22.214	24.535	0.0	2.238	2.465	0.101	0.100
20	$\mathcal{K}^{100}$	2.071	26.002	0.0	0.201	2.569	0.097	0.099
20	$(\mathcal{K}^{100})^{10}$	23.366	34.497	0.0	2.326	3.407	0.100	0.099

Table 8: Result of Experiment 4 ($\delta = 10^{-6}$)

m	$\mathcal{K}$	$\text{CPU}_{\text{total}}$		$\sharp$hot start	$\text{CPU}_{\text{first}}$		$\text{CPU}_{\text{first}}/\text{CPU}_{\text{total}}$	
		SDPT3	DSPE	DSPE	SDPT3	DSPE	SDPT3	DSPE
10	$\mathcal{K}^{20}$	1.489	0.794	6.4	0.228	0.228	0.153	0.287
10	$(\mathcal{K}^{20})^{10}$	12.617	0.419	8.0	1.261	0.208	0.100	0.496
10	$\mathcal{K}^{100}$	1.455	0.975	5.7	0.144	0.224	0.099	0.230
10	$(\mathcal{K}^{100})^{10}$	16.813	0.992	7.4	1.673	0.367	0.100	0.370
20	$\mathcal{K}^{40}$	1.840	23.896	1.2	0.179	2.770	0.097	0.116
20	$(\mathcal{K}^{40})^{10}$	19.038	11.094	5.3	1.886	2.215	0.099	0.200
20	$\mathcal{K}^{100}$	1.837	21.896	2.2	0.178	2.798	0.097	0.128
20	$(\mathcal{K}^{100})^{10}$	20.864	26.309	2.7	2.077	3.606	0.100	0.137

Table 9: Result of Experiment 4 ($\delta = 10^{-5}$)

m	$\mathcal{K}$	$\text{CPU}_{\text{total}}$		$\sharp$hot start	$\text{CPU}_{\text{first}}$		$\text{CPU}_{\text{first}}/\text{CPU}_{\text{total}}$	
		SDPT3	DSPE	DSPE	SDPT3	DSPE	SDPT3	DSPE
10	$\mathcal{K}^{20}$	1.670	2.640	0.8	0.168	0.288	0.101	0.109
10	$(\mathcal{K}^{20})^{10}$	18.325	2.183	1.8	1.818	0.257	0.099	0.118
10	$\mathcal{K}^{100}$	1.850	2.921	0.2	0.187	0.292	0.101	0.100
10	$(\mathcal{K}^{100})^{10}$	20.530	3.906	1.1	2.054	0.442	0.100	0.113
20	$\mathcal{K}^{40}$	2.114	29.582	0.0	0.215	2.931	0.102	0.099
20	$(\mathcal{K}^{40})^{10}$	20.769	25.558	0.0	2.098	2.658	0.101	0.104
20	$\mathcal{K}^{100}$	1.967	29.799	0.0	0.196	2.962	0.100	0.099
20	$(\mathcal{K}^{100})^{10}$	22.509	36.049	0.0	2.253	3.596	0.100	0.100

Table 10: Result of Experiment 4 ($\delta = 10^{-4}$)

m	$\mathcal{K}$	$\text{CPU}_{\text{total}}$		$\sharp$hot start	$\text{CPU}_{\text{first}}$		$\text{CPU}_{\text{first}}/\text{CPU}_{\text{total}}$	
		SDPT3	DSPE	DSPE	SDPT3	DSPE	SDPT3	DSPE
10	$\mathcal{K}^{20}$	1.999	2.988	0.0	0.197	0.310	0.099	0.104
10	$(\mathcal{K}^{20})^{10}$	18.269	2.909	0.0	1.924	0.294	0.105	0.101
10	$\mathcal{K}^{100}$	1.898	3.024	0.0	0.181	0.300	0.095	0.099
10	$(\mathcal{K}^{100})^{10}$	18.654	4.517	0.0	1.863	0.458	0.100	0.101
20	$\mathcal{K}^{40}$	2.116	30.578	0.0	0.211	3.105	0.100	0.102
20	$(\mathcal{K}^{40})^{10}$	20.820	25.623	0.0	2.085	2.558	0.100	0.100
20	$\mathcal{K}^{100}$	1.985	30.376	0.0	0.196	3.037	0.099	0.100
20	$(\mathcal{K}^{100})^{10}$	21.972	35.406	0.0	2.203	3.640	0.100	0.103