Journal of the Operations Research
Society of Japan
2008, Vol. 51, No. 3, 241-264

SPARSE SECOND ORDER CONE PROGRAMMING FORMULATIONS FOR CONVEX OPTIMIZATION PROBLEMS

Kazuhiro Kobayashi
Tokyo Institute of Technology

Sunyoung Kim*
Ewha Women's University

Masakazu Kojima
Tokyo Institute of Technology

(Received February 21, 2008; Revised June 3, 2008)

Abstract Second order cone program (SOCP) formulations of convex optimization problems are studied. We show that various SOCP formulations can be obtained depending on how auxiliary variables are introduced. An efficient SOCP formulation that increases the computational efficiency is presented by investigating the relationship between the sparsity of an SOCP formulation and the sparsity of the Schur complement matrix. Numerical results of selected test problems using SeDuMi and LANCELOT are included to demonstrate the performance of the SOCP formulation.

Keywords: Optimization, convex optimization problem, second-order cone program, correlative sparsity, primal-dual interior-point method, the Schur complement matrix

1. Introduction

We consider second order cone program (SOCP) approaches for convex optimization problems. SOCPs have received plenty of attention in recent studies of optimization for their wide applicability and computational efficiency [1, 6, 7, 11, 13]. SOCP can be viewed as a special case of semidefinite programming (SDP) in the sense that second order cone inequalities can be represented as linear matrix inequalities. The computational efforts for solving SDPs are, however, known to be far greater than for SOCPs. It is thus recommended to use SOCP formulation for computational complexity concerns when an optimization problem can be formulated as both an SDP and an SOCP [1].

Formulating optimization problems as SOCPs provides computational advantages: it can be solved in polynomial-time, and the number of iterations required to find a solution is not much affected by a choice of initial points in practice. Nesterov and Nemirovski [12, 13] and Lobo et al. [11] showed that many kinds of problems could be formulated as SOCPs. They introduced second order cone representable functions or sets for convex optimization problems that can be formulated as SOCPs. For a convex optimization problem which can be formulated as an SOCP, there usually exists more than one formulation that are all equivalent. The computational complexity for solving the SOCPs thus varies depending on the formulation.

Sparsity has been utilized in various ways for solving large-sized problems and studied extensively. The correlative sparsity was introduced to handle the sparsity of polynomial optimization problems (POPs) in [18]. An $n \times n$ symmetric matrix $\boldsymbol{R}$, correlative sparsity pattern (csp) matrix, is constructed for the representation of the correlative sparsity of a POP with each element R_{ij} of the csp matrix $\boldsymbol{R}$ either 0 or $\star$ for a nonzero value.

*The research of S. Kim was supported by KOSEF R01-2005-000-10271-0 and KRF-2006-312-C00062

The importance of the correlative sparsity lies in the fact that applying sparse Cholesky factorization to the csp matrix $\boldsymbol{R}$ provides no fill-ins.

In the implementation of interior-point methods for SOCP, the sparsity of the Schur complement matrix was exploited by splitting the matrix into sparse and dense parts, factorizing the sparse part, and applying a low-rank update to the dense part [3, 15]. From this, we see that the computational efficiency of solving SOCPs will be improved if the sparse part of the Schur complement matrix contains less nonzero elements. In recent work [9], it is shown that if optimization problems have the correlative sparsity, then the same sparsity pattern exists in the Schur complement matrix in primal-dual interior point methods for LP, SOCP and SDP. However, SOCP formulation of a convex optimization problem is not unique. SOCP formulations that increase computational efficiency in solving the Schur complement equation have potential to perform better when solving large-sized problems.

The objective of this paper is to find SOCP formulations that increase the computational efficiency in solving the Schur complement equation. For this purpose, we define second order cone inequality (SOCI) representable sets and functions. We then investigate various SOCP formulations in terms of correlative sparsity, and show that different ways of formulating convex optimization problems with SOCI representable functions as SOCPs result in the same sparsity pattern in the sparse part of the Schur complement matrix for the original variables. The difference lies on the auxiliary variables introduced to formulate the convex optimization problem as an SOCP and how they create nonzero elements in the sparse part of the Schur complement matrix. We show that an efficient SOCP formulation can be obtained by minimizing the number of auxiliary variables. This efficient SOCP formulation has an SOCP inequality with the largest dimension among the various equivalent SOCP formulations. We recommend this formulation to increase the computational efficiency.

Another objective of this paper is to compare the results of solving optimization problems by using SeDuMi with those by using LANCELOT. SOCPs formulated from optimization problems can be solved by several available software packages based on primal-dual interior-point methods such as SeDuMi [14], MOSEK [19], SDPT3 [16]. Convex optimization problems with sparsity can also be solved with LANCELOT, which takes advantage of sparsity from the partial separability of optimization problems. The partial separability was introduced in connection with the efficient implementation of quasi-Newton methods for solving large unconstrained optimization [5]. The Hessian matrix of the problems with partial separability is sparse, and the LANCELOT optimization package [2] makes efficient use of this sparsity. We compare the numerical results from the LANCELOT with those by SeDuMi for SOCP formulations obtained by minimizing the number of auxiliary variables.

This paper is organized as follows: after introducing a brief description of an SOCI representable set or function, notation, and basic definitions, the restricted hyperbolic constraint are included in Section 2. Section 3 contains SOCP formulations of convex optimization problems. Section 4 includes the discussion on the correlative sparsity of various SOCP formulations and how efficient SOCP formulations can be obtained, based on SeDuMi's handling of the Schur complement matrix. Section 5 contains numerical experiments for unconstrained and constrained optimization problems. For constrained problems, the description of generating constrained test problems using existing functions from CUTEr [4] is included. Numerical results obtained using SeDuMi are compared with LANCELOT for various sparse optimization problems. Finally, Section 6 is devoted to concluding remarks.

2. Preliminaries

2.1. Notation and definition

Let $\mathcal{S}^n$ be the second order cone defined as

$$\mathcal{S}^n = \left\{ \boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n | \ x_1 \geq \left(\sum_{i=2}^{n} x_i^2 \right)^{1/2} \right\}.$$

For every $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{x} \succeq_{\mathbb{S}} \boldsymbol{0}$ denotes the second order cone (or quadratic cone) inequality, *i.e.*,

$$\boldsymbol{x} \succeq_{\mathbb{S}} \boldsymbol{0} \text{ if and only if } x_1 \geq \left(\sum_{i=2}^{n} x_i^2 \right)^{1/2}.$$

We use $\boldsymbol{x} = (x_1, \boldsymbol{x}_2) = (x_1, x_2, \ldots, x_n)$. We also let $\mathcal{S}^n_\star$ be the Cartesian product of several second order cones, *i.e.*, $\mathcal{S}^n_\star = \mathcal{S}^{k_1} \times \mathcal{S}^{k_2} \cdots \times \mathcal{S}^{k_m}$ where $\mathcal{S}^{k_i} \subset \mathbb{R}^{k_i}$ is a second order cone. If $\sum_{i=1}^{m} k_i = n$ and $\boldsymbol{x} \in \mathcal{S}^n_\star$, $\boldsymbol{x}$ can be expressed as $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m)$ where $\boldsymbol{x}_i \in \mathcal{S}^{k_i}$. Since $\mathcal{S}^{k_i}$ is a second order cone in $\mathbb{R}^{k_i}$, $\boldsymbol{x} \in \mathcal{S}^n_\star$ indicates that each subvector $\boldsymbol{x}_i$ $(i = 1, 2, \ldots, m)$ of $\boldsymbol{x}$ satisfies the inequality

$$x_{i1} \geq \left(\sum_{j=2}^{k_i} x_{ij}^2 \right)^{1/2},$$

where $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ik_i})$ $(1 \leq i \leq m)$. For every $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m) \in \mathbb{R}^n$, $\boldsymbol{x} \succeq_{\mathbb{S}} \boldsymbol{0}$ also denotes a product of multiple second order cone inequalities, *i.e.*, $\boldsymbol{x} \succeq_{\mathbb{S}} \boldsymbol{0}$ if and only if $x_{i1} \geq \left(\sum_{j=2}^{k_i} x_{ij}^2 \right)^{1/2}$ $(i = 1, 2, \ldots, m)$. Throughout the paper, for every convex subset K of $\mathbb{R}^n$, we let $\mathbb{F}(K)$ the set of real valued functions defined on an open neighborhood of K, $\mathbb{F}_+(K) = \{f \in \mathbb{F}(K) : f(\boldsymbol{x}) \geq 0 \text{ for every } \boldsymbol{x} \in K\}$, $\mathbb{F}_{++}(K) = \{f \in \mathbb{F}(K) : f(\boldsymbol{x}) > 0 \text{ for every } \boldsymbol{x} \in K\}$, $\mathbb{A}\mathrm{ff}(K) = \{f \in \mathbb{F}(K) : f \text{ is affine on } K\}$. Here we say that f is affine on K if $f((1-\lambda)\boldsymbol{x} + \lambda\boldsymbol{y}) = (1-\lambda)f(\boldsymbol{x}) + \lambda f(\boldsymbol{y})$ for every $\boldsymbol{x} \in K, \boldsymbol{y} \in K$ and $\lambda \in [0, 1]$.

We call a subset C of $\mathbb{R}^n$ *second order cone inequality (SOCI) representable* if there exists an affine map $\boldsymbol{F} : \mathbb{R}^{\widehat{m}+n} \to \mathcal{S}^q$ for which $C = \{\boldsymbol{z} \in \mathbb{R}^n : \boldsymbol{F}(\boldsymbol{y}, \boldsymbol{z}) \succeq_{\mathbb{S}} \boldsymbol{0} \text{ for some } \boldsymbol{y} \in \mathbb{R}^{\widehat{m}}\}$ holds , where we assume that $\hat{m} \geq 1$.

Every SOCI representable subset of $\mathbb{R}^n$ is convex. We assume that the entire n-dimensional space $\mathbb{R}^n$ is SOCI representable. In what follows, K denotes a fixed convex subset of $\mathbb{R}^n$ which is SOCI representable; hence $K = \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{F}_K(\boldsymbol{y}, \boldsymbol{x}) \succeq_{\mathbb{S}} \boldsymbol{0} \text{ for some } \boldsymbol{y} \in \mathbb{R}^{\widehat{m}}\}$, where $\boldsymbol{F}_K : \mathbb{R}^{\widehat{m}+n} \to \mathcal{S}^q$ is an affine map.

For every subset K of $\mathbb{R}^n$ and every $f \in \mathbb{F}(K)$, let

$$\begin{aligned} \mathrm{epi}(f, K) &= \left\{(t, \boldsymbol{x}) \in \mathbb{R}^{1+n} : t - f(\boldsymbol{x}) \geq 0 \text{ and } \boldsymbol{x} \in K\right\} \\ &\quad \text{(the epigraph of } f \text{ restricted to } K), \\ \mathrm{hyp}(f, K) &= \left\{(t, \boldsymbol{x}) \in \mathbb{R}^{1+n} : t - f(\boldsymbol{x}) \leq 0 \text{ and } \boldsymbol{x} \in K\right\} \\ &\quad \text{(the hypograph of } f \text{ restricted to } K). \end{aligned}$$

Define

$$\begin{aligned} \mathbb{S}\mathrm{epi}(K) &= \{f \in \mathbb{F}(K) : \mathrm{epi}(f, K) \text{ is SOCI representable}\}, \\ \mathbb{S}\mathrm{hyp}(K) &= \{f \in \mathbb{F}(K) : \mathrm{hyp}(f, K) \text{ is SOCI representable}\}. \end{aligned}$$

In addition, for $f \in \mathbb{S}\mathrm{epi}(K)$ or $f \in \mathbb{S}\mathrm{hyp}(K)$, we call f an SOCI representable function.

When $K = \mathbb{R}^n$, we often omit K and use the symbols $\mathbb{F}$, $\mathbb{F}_+$, $\mathbb{F}_{++}$, $\mathbb{A}\mathrm{ff}$, $\mathrm{epi}(f), \mathrm{hyp}(f), \mathbb{S}\mathrm{epi}$ and $\mathbb{S}\mathrm{hyp}$ for $\mathbb{F}(K)$, $\mathbb{F}_+(K)$, $\mathbb{F}_{++}(K)$, $\mathbb{A}\mathrm{ff}(K)$, $\mathrm{epi}(f, K), \mathrm{hyp}(f, K), \mathbb{S}\mathrm{epi}(K)$ and $\mathbb{S}\mathrm{hyp}(K)$, respectively.

We note that the discussion of SOCI representable functions and sets in [11–13] deals with general concepts and representation. In this paper, we are more interested in computational efficiency of various SOCPs formulated from SOCI representable functions.

2.2. The restricted hyperbolic constraint

When formulating a problem as a second order cone, we frequently use *the restricted hyperbolic constraint*:

$$\boldsymbol{z}^T\boldsymbol{z} \leq uv,\ u \geq 0,\ v \geq 0 \Leftrightarrow \left(\ u+v,\ u-v,\ 2\boldsymbol{z}\ \right)^T \succeq_{\mathbb{S}} 0,$$

where $u \in \mathbb{R}$, $v \in \mathbb{R}$ and $\boldsymbol{z} \in \mathbb{R}^p$.

3. SOCP Formulation

We consider solving the following convex optimization problem by formulating it as an SOCP. Let $K = \mathbb{R}^n$.

$$\min \quad f_0(\boldsymbol{x}) \quad \text{subj. to} \quad f_j(\boldsymbol{x}) \leq 0\ (j = 1, 2, \ldots, \widetilde{m}). \tag{1}$$

If $f_j \in \mathbb{S}\mathrm{epi}$, then (1) can be formulated as an SOCP by introducing auxiliary variables.

As mentioned in Section 2, $f_j \in \mathbb{S}\mathrm{epi}$ implies that $\mathrm{epi}(f_j)$ can be represented as an SOCI; there exists an affine map $\boldsymbol{F}_j : \mathbb{R}^{\widehat{m}+1+n} \to \mathcal{S}^q$ for which

$$\mathrm{epi}(f_j, K) = \{(t, \boldsymbol{x}) \in \mathbb{R}^{1+n}\ : \boldsymbol{F}_j(\boldsymbol{z}, t, \boldsymbol{x}) \succeq_{\mathbb{S}} \boldsymbol{0} \text{ for some } \boldsymbol{z} \in \mathbb{R}^{\widehat{m}}\}$$

holds. Thus, an inequality $f_j(\boldsymbol{x}) \leq 0$ can be represented as

$$\boldsymbol{F}_j(\boldsymbol{z}, 0, \boldsymbol{x}) \succeq_{\mathbb{S}} \boldsymbol{0} \quad \text{for some } \boldsymbol{z} \in \mathbb{R}^{\widehat{m}}. \tag{2}$$

When a given optimization problem (1) described by SOCI representable functions is formulated as an SOCP, we need to introduce auxiliary variables, for instance $\boldsymbol{z}$ in (2), for affine maps. The way to represent an SOCI representable function as an SOCI is not unique as we see in the following.

Let us consider

$$\min f_0(\boldsymbol{x}) = \sum_{i \in \mathcal{I}} f_i(\boldsymbol{x})^{2^{P_i}} \tag{3}$$

where $P_i = \{0, 1, 2, 3, \ldots\}$ and $f_i \in \mathbb{A}\mathrm{ff}$ $(i \in \mathcal{I})$. Let

$$\mathcal{I}_0 \ = \ \{i : P_i = 0\},\ \mathcal{I}_1 = \{i : P_i = 1\},\ \mathcal{I}_2 = \{i : P_i \geq 2\},\ \mathcal{I} = \mathcal{I}_0 \cup \mathcal{I}_1 \cup \mathcal{I}_2.$$

Then, (3) can be written as

$$\min \sum_{i \in \mathcal{I}_0} f_i(\boldsymbol{x}) + \sum_{i \in \mathcal{I}_1} f_i(\boldsymbol{x})^2 + \sum_{i \in \mathcal{I}_2} t_i^2 \ \text{ subj. to } \ f_i(\boldsymbol{x})^{2^{(P_i-1)}} \leq t_i \ \ (i \in \mathcal{I}_2),$$

or equivalently,

$$\min \sum_{i\in\mathcal{I}_0} f_i(\boldsymbol{x}) + s \ \text{ subj. to } \ \sum_{i\in\mathcal{I}_1} f_i(\boldsymbol{x})^2 + \sum_{i\in\mathcal{I}_2} t_i^2 \le s,\ f_i(\boldsymbol{x})^{2^{(P_i-1)}} \le t_i \ \ (i\in\mathcal{I}_2).$$

If $P_i = 2$, then $f_i(\boldsymbol{x})^{2^{(P_i-1)}} \le t_i$ can be represented as $(t_i+1, t_i-1, 2f_i(\boldsymbol{x}))^T \succeq_{\mathbb{S}} \boldsymbol{0}$.

For $P_i \ge 3$, introducing auxiliary variables into the inequality $f_i(\boldsymbol{x})^{2^{(P_i-1)}} \le t_i$ until the power of $f_i(\boldsymbol{x})$ becomes 2, we replace it by the sequence of inequalities $t_i \ge u_i^2,\ u_i \ge v_i^2, \ldots, w_i \ge f_i(\boldsymbol{x})^2$. Then, the problem is equivalent to

$$\left.\begin{array}{ll} \min & \displaystyle\sum_{i\in\mathcal{I}_0} f_i(\boldsymbol{x}) + s \\ \text{subj. to} & \displaystyle\sum_{i\in\mathcal{I}_1} f_i(\boldsymbol{x})^2 + \sum_{i\in\mathcal{I}_2} t_i^2 \le s,\ t_i \ge u_i^2,\ u_i \ge v_i^2, \ldots, w_i \ge f_i(\boldsymbol{x})^2\ (i\in\mathcal{I}_2). \end{array}\right\} \tag{4}$$

Alternatively,

$$\left.\begin{array}{ll} \min & \displaystyle\sum_{i\in\mathcal{I}_0} f_i(\boldsymbol{x}) + \sum_{i\ \in\ \mathcal{I}_1} t_i + \sum_{i\ \in\ \mathcal{I}_2} t_i \\ \text{subj. to} & t_i \ge f_i(\boldsymbol{x})^2\ (i\in\mathcal{I}_1),\ t_i \ge u_i^2,\ u_i \ge v_i^2, \ldots, w_i \ge f_i(\boldsymbol{x})^2\ (i\in\mathcal{I}_2). \end{array}\right\} \tag{5}$$

Let α and β be the number of indices in $\mathcal{I}_1$ and $\mathcal{I}_2$, respectively, and $\mathcal{I}_1 = \{i_{1j} : j = 1, \ldots, \alpha\}$ and $\mathcal{I}_2 = \{i_{2k} : k = 1, \ldots, \beta\}$. We now consider the constraint

$$\sum_{i\in\mathcal{I}_1} f_i(\boldsymbol{x})^2 + \sum_{i\in\mathcal{I}_2} t_i^2 \le s \tag{6}$$

in the problem (4). The second order cone representation of (6) is

$$\left(s+1, s-1, 2f_{i_{11}}(\boldsymbol{x}), 2f_{i_{12}}(\boldsymbol{x}), \ldots, 2f_{i_{1\alpha}}(\boldsymbol{x}), 2t_{i_{21}}, 2t_{i_{22}}, \ldots, 2t_{i_{2\beta}}\right)^T \succeq_{\mathbb{S}} \boldsymbol{0},$$

and the size of the second order cone is $2+\alpha+\beta$. On the other hand, the sizes of all the second order cones induced from the constraints in (5) are 3. The SOCP formulation of (5) thus involves smaller size second order cones than that of (4). We note that another SOCP formulation can be obtained by replacing the variable s in the objective function of the problem (4) by $s' + s''$ and splitting the constraint $\sum_{i\in\mathcal{I}_1} f_i(\boldsymbol{x})^2 + \sum_{i\in\mathcal{I}_2} t_i^2 \le s$ into $\sum_{i\in\mathcal{I}_1} f_i(\boldsymbol{x})^2 \le s'$ and $\sum_{i\in\mathcal{I}_2} t_i^2 \le s''$. Introducing less auxiliary variables than (5) and more auxiliary variables than (4) leads to an SOCP formulation in which the size of the largest second order cone is smaller than that of (4) and larger than that of (5). Formulating the problem in which the size of the largest second order cone is the largest among the equivalent SOCP formulations provides good computational efficiency.

If a problem has a constraint $\sum_{i\in\mathcal{I}} f_i(\boldsymbol{x})^{2^{P_i}} \le g(\boldsymbol{x})$, the SOCP formulation described above can be also applied, where $g \in \mathbb{A}\mathrm{ff}$.

Similarly, problems that involve $\left(\sum_{i\in\mathcal{I}} f_i(\boldsymbol{x})^{2^{P_i}}\right)^{1/2}$, with $P_i \ge 1$ (hence $\mathcal{I}_0 = \emptyset$) in their objective functions to be minimized or their inequality constraints can be formulated as SOCPs using

$$s \ge \left(\sum_{i\in\mathcal{I}_1} f_i(\boldsymbol{x})^2 + \sum_{i\in\mathcal{I}_2} t_i^2\right)^{1/2},\ t_i \ge u_i^2,\ u_i \ge v_i^2, \ldots, w_i \ge f_i(\boldsymbol{x})^2\ (i\in\mathcal{I}_2).$$

We also mention that SOCP formulations of problems involving $\left(\sum_{i\in\mathcal{I}} f_i(\boldsymbol{x})^{2^{P_i}}\right)/g(\boldsymbol{x})$ in their objective functions to be minimized, where $g \in \mathbb{A}\mathrm{ff}_{++}(K)$ for some SOCI representable convex subset K of $\mathbb{R}^n$, can be derived. More precisely, we consider the constraints

$$\left(\sum_{i\in\mathcal{I}} f_i(\boldsymbol{x})^{2^{P_i}}\right)/g(\boldsymbol{x}) \leq t \text{ and } \boldsymbol{x} \in K$$

where t is an auxiliary variable representing the objective value, and write

$$\left(\sum_{i\in\mathcal{I}} f_i(x)^{2^{P_i}}\right) \leq tg(\boldsymbol{x}) \text{ and } \boldsymbol{x} \in K$$

as second order cones by using the restricted hyperbolic constraint given in Section 2.2.

For theoretical complexity aspect of an SOCP, Tsuchiya [17] showed that the long-step algorithm for SOCP using NT direction has $O(k \log \epsilon^{-1})$ iteration-complexity to reduce the duality gap by a factor of ϵ, where k is the number of second order cones. We observe this with three SOCP formulations of varying number of second order cones in the following illustrative example.

An illustrative example: the Chained singular function

As an illustrative example, we show three different SOCP formulations of the Chained singular function. We consider minimizing the Chained singular function

$$\min \sum_{i\in J} \left((x_i + 10x_{i+1})^2 + 5(x_{i+2} - x_{i+3})^2 + (x_{i+1} - 2x_{i+2})^4 + 10(x_i - 10x_{i+3})^4\right), \qquad (7)$$

where $J = \{1, 3, 5, \ldots, n-3\}$ and n is a multiple of 4. This can be rewritten as

$$\begin{array}{ll} \min & \displaystyle\sum_{i\in J}(s_i + t_i + p_i + q_i) \\ \text{subj. to} & s_i \geq (x_i + 10x_{i+1})^2,\ t_i \geq 5\,(x_{i+2} - x_{i+3})^2,\ r_i \geq (x_{i+1} - 2x_{i+2})^2, \\ & p_i \geq r_i^2,\ u_i \geq \sqrt{10}\,(x_i - 10x_{i+3})^2,\ q_i \geq u_i^2\ (i \in J). \end{array}$$

We can formulate this problem as an SOCP:

$$\left.\begin{array}{ll} \min & \displaystyle\sum_{i\in J}(s_i + t_i + p_i + q_i) \\ \text{subj. to} & \begin{pmatrix} s_i + 1 \\ s_i - 1 \\ 2(x_i + 10x_{i+1}) \end{pmatrix} \succeq_{\mathbb{S}} 0,\ \begin{pmatrix} t_i + 1 \\ t_i - 1 \\ 2\sqrt{5}(x_{i+2} - x_{i+3}) \end{pmatrix} \succeq_{\mathbb{S}} 0, \\ & \begin{pmatrix} r_i + 1 \\ r_i - 1 \\ 2(x_{i+1} - 2x_{i+2}) \end{pmatrix} \succeq_{\mathbb{S}} 0,\ \begin{pmatrix} p_i + 1 \\ p_i - 1 \\ 2r_i \end{pmatrix} \succeq_{\mathbb{S}} 0, \\ & \begin{pmatrix} u_i + 1 \\ u_i - 1 \\ 2\sqrt[4]{10}(x_i - 10x_{i+3}) \end{pmatrix} \succeq_{\mathbb{S}} 0,\ \begin{pmatrix} q_i + 1 \\ q_i - 1 \\ 2u_i \end{pmatrix} \succeq_{\mathbb{S}} 0\ (i \in J). \end{array}\right\} \qquad (8)$$

The sizes of all the second order cones are 3.

Minimizing the Chained singular function (7) can also be rewritten as

$$\begin{array}{ll} \min & s + \sum_{i\in J}(p_i + q_i) \\ \text{subj. to} & s \geq \sum_{i\in J}\left((x_i + 10x_{i+1})^2 + 5(x_{i+2} - x_{i+3})^2\right), \\ & r_i \geq (x_{i+1} - 2x_{i+2})^2,\ p_i \geq r_i^2, \\ & u_i \geq \sqrt{10}(x_i - 10x_{i+3})^2,\ q_i \geq u_i^2\ (i \in J). \end{array}$$

We can formulate this problem as an SOCP:

$$\left.\begin{array}{ll} \min & s + \sum_{i\in J}(p_i + q_i) \\ \text{subj. to} & \text{a single SOCP inequality to be derived from} \\ & s \geq \sum_{i\in J}\left((x_i + 10x_{i+1})^2 + 5(x_{i+2} - x_{i+3})^2\right), \\ & \begin{pmatrix} r_i + 1 \\ r_i - 1 \\ 2(x_{i+1} - 2x_{i+2}) \end{pmatrix} \succeq_{\mathbb{S}} 0,\ \begin{pmatrix} p_i + 1 \\ p_i - 1 \\ 2r_i \end{pmatrix} \succeq_{\mathbb{S}} 0, \\ & \begin{pmatrix} u_i + 1 \\ u_i - 1 \\ 2\sqrt[4]{10}(x_i - 10x_{i+3}) \end{pmatrix} \succeq_{\mathbb{S}} 0,\ \begin{pmatrix} q_i + 1 \\ q_i - 1 \\ 2u_i \end{pmatrix} \succeq_{\mathbb{S}} 0\ (i \in J). \end{array}\right\} \tag{9}$$

The single SOCP inequality is of the following form

$$\begin{pmatrix} s + 1 \\ s - 1 \\ 2(x_1 + 10x_{1+1}) \\ 2(x_3 + 10x_{3+1}) \\ \vdots \\ 2(x_{n-3} + 10x_{n-3+1}) \\ 2\sqrt{5}(x_{1+2} - x_{1+3}) \\ 2\sqrt{5}(x_{3+2} - x_{3+3}) \\ \vdots \\ 2\sqrt{5}(x_{n-3+2} - x_{n-3+3}) \end{pmatrix} \succeq_{\mathbb{S}} 0.$$

The dimension of this single SOCP inequality is n. Now we have two different SOCP formulations (8) and (9).

A different SOCP formulation can be derived: If we write minimizing the Chained singular function

$$\begin{array}{ll} \min & s \\ \text{subj. to} & s \geq \sum_{i\in J}\left((x_i + 10x_{i+1})^2 + 5(x_{i+2} - x_{i+3})^2 + r_i^2 + u_i^2\right), \\ & r_i \geq (x_{i+1} - 2x_{i+2})^2,\ u_i \geq \sqrt{10}(x_i - 10x_{i+3})^2\ (i \in J), \end{array}$$

an SOCP can be formulated as

$$\left.\begin{array}{ll} \min & s \\ \text{subj. to} & \text{a single SOCP inequality to be derived from} \\ & s \geq \sum_{i\in J}\left((x_i + 10x_{i+1})^2 + 5(x_{i+2} - x_{i+3})^2 + r_i^2 + u_i^2\right), \\ & \begin{pmatrix} r_i + 1 \\ r_i - 1 \\ 2(x_{i+1} - 2x_{i+2}) \end{pmatrix} \succeq_{\mathbb{S}} 0,\ \begin{pmatrix} u_i + 1 \\ u_i - 1 \\ 2\sqrt[4]{10}(x_i - 10x_{i+3}) \end{pmatrix} \succeq_{\mathbb{S}} 0\ (i \in J). \end{array}\right\} \tag{10}$$

The single SOCP inequality is represented as

$$\begin{pmatrix} s+1 \\ s-1 \\ 2(x_1+10x_{1+1}) \\ 2(x_3+10x_{3+1}) \\ \vdots \\ 2(x_{n-3}+10x_{n-3+1}) \\ 2\sqrt{5}(x_{1+2}-x_{1+3}) \\ 2\sqrt{5}(x_{3+2}-x_{3+3}) \\ \vdots \\ 2\sqrt{5}(x_{n-3+2}-x_{n-3+3}) \\ 2r_1 \\ 2r_3 \\ \vdots \\ 2r_{n-3} \\ 2u_1 \\ 2u_3 \\ \vdots \\ 2u_{n-3} \end{pmatrix} \succeq_{\mathbb{S}} 0.$$

Note that the size of the first second order cone is $2(n-1)$ and the size of all the other second order cones is 3.

4. Sparsity

When different SOCP formulations shown in Section 3 are solved by a software based on primal-dual interior-point methods, their computational efficiency varies. The most time consuming part is solving the Schur complement equation. Two important factors that affect the efficiency of solving the Schur complement equation are the sparsity and the size of the Schur complement matrix. We examine how SeDuMi [14] handles the Schur complement equations from various SOCP formulations, resulting in different computational time.

Consider the primal-dual standard form SOCP:

$$\text{min} \quad \sum_{i=1}^{\ell} \boldsymbol{c}_i^T \boldsymbol{x}_i \quad \text{subj. to} \quad \sum_{i=1}^{\ell} \boldsymbol{A}_i \boldsymbol{x}_i = \boldsymbol{b}, \ \boldsymbol{x}_i \succeq_{\mathbb{S}} \boldsymbol{0} \ (i = 1, 2, \ldots, \ell), \tag{11}$$

$$\text{max} \quad \boldsymbol{b}^T \boldsymbol{y} \quad \text{subj. to} \quad \boldsymbol{s}_i = \boldsymbol{c}_i - \boldsymbol{A}_i^T \boldsymbol{y} \succeq_{\mathbb{S}} \boldsymbol{0} \ \ (i = 1, 2, \ldots, \ell), \tag{12}$$

where $\boldsymbol{c}_i,\ \boldsymbol{x}_i,\ \boldsymbol{s}_i \in \mathbb{R}^{k_i}$, $\boldsymbol{A}_i \in \mathbb{R}^{m \times k_i}$ $(i = 1, \ldots, \ell)$, and $\boldsymbol{b},\ \boldsymbol{y} \in \mathbb{R}^m$.

In primal-dual interior-point methods for solving SOCP, the Cholesky factorization is commonly used for the solution of the Schur complement equation. The sparsity of the Schur complement matrix can be explained in connection with the sparsity of SOCP by considering the correlative sparsity pattern (csp) matrix, which was originally proposed for a POP [18], for the dual standard form SOCP (12). The csp matrix of SOCP is defined by $m \times m$ symmetric matrix $\boldsymbol{R}$, called the correlative sparsity pattern (csp) matrix whose element R_{jk} is either 0 or $*$ for a nonzero value. The symbol $*$ was assigned to all diagonal elements of $\boldsymbol{R}$ and also to each off-diagonal element $R_{jk} = R_{kj}$ $(1 \leq j < k \leq m)$ if and only if the variables y_j and y_k appear simultaneously in a second order cone inequality constraint

$\boldsymbol{s}_i = \boldsymbol{c}_i - \boldsymbol{A}_i^T \boldsymbol{y} \succeq_{\mathbb{S}} \boldsymbol{0}$. We note that the sparsity pattern of the Schur complement matrix (the coefficient matrix of the Schur complement equation) coincides with the csp matrix $\boldsymbol{R}$ [9].

In the implementation of the primal-dual interior-point method, the Schur complement matrix is splitted into sparse and dense parts. Then, the sparse part is factorized and a low-rank update is applied to the dense part. Therefore, even with the dense Schur complement matrix, the sparsity structure of the sparse part of the Schur complement matrix can still be exploited after the splitting. Thus, the computational efforts for solving the Schur complement equation depend on the size of the Schur complement matrix, the sparsity of the sparse part, and the rank of the dense part.

Notice that SOCPs formulated with small second order cones have more variables than those with large second order cones because more auxiliary variables are introduced. As a result, the size of the Schur complement matrix is larger.

The sparsity of the Schur complement matrix is determined by the sparsity of the data matrices $\boldsymbol{A}_i$ $(i = 1, 2, \ldots, \ell)$ in (12). More specifically, the nonzero pattern of the sparse part of the Schur complement matrix coincides with the nonzero pattern of the matrix $\sum_{i=1}^{\ell} \boldsymbol{A}_i \boldsymbol{A}_i^T$. From this, we can observe that different SOCP formulations from various ways of introducing auxiliary variables do not change the sparsity pattern of the sparse part of the Schur complement matrix essentially, and only the number of variables are different. This will be shown with the illustrative example in this section.

The csp matrix becomes increasingly dense as a second order cone of SOCP formulation includes more variables, which was called a dense constraint in [9], for example (10). We need to know how SeDuMi handles the Schur complement equation to compare the computational efficiency of various SOCP formulations. SeDuMi implements the product-form Cholesky factorization based on the rank-1 Fletcher-Powell method [3] for solving the Schur complement equation. The product form approach can be described as follows. Let $\mathcal{I} \subset \{1, \ldots, \ell\}$ be the index set of second order cones involving many variables. We also let $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_\ell)$ be an interior-feasible solution of the primal standard form SOCP (11) and $(\boldsymbol{y}, \boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_\ell)$ an interior-feasible solution of the dual standard form SOCP (12). We notice that SeDuMi uses the Nesterov-Todd direction as the search direction, and denote

$$
\begin{aligned}
\gamma(\boldsymbol{x}_i) &= \sqrt{x_{i1}^2 - \|\boldsymbol{x}_{i2}\|^2}, \\
u_{i1} &= \{(\gamma(\boldsymbol{s}_i) x_{i1} + \gamma(\boldsymbol{x}_i) s_{i1})/\gamma(\boldsymbol{x}_i)\} \sqrt{\boldsymbol{x}_i^T \boldsymbol{s}_i + \gamma(\boldsymbol{x}_i)\gamma(\boldsymbol{s}_i)}, \\
\boldsymbol{u}_{i2} &= \{(-\gamma(\boldsymbol{s}_i) \boldsymbol{x}_{i2} + \gamma(\boldsymbol{x}_i) \boldsymbol{s}_{i2})/\gamma(\boldsymbol{x}_i)\} \sqrt{\boldsymbol{x}_i^T \boldsymbol{s}_i + \gamma(\boldsymbol{x}_i)\gamma(\boldsymbol{s}_i)}.
\end{aligned}
$$

In addition, we denote by A_{i1} the first column of $\boldsymbol{A}_i$. Then, the Schur complement matrix is represented as

$$
\sum_{i=1}^{\ell} \boldsymbol{A}_i \boldsymbol{F}_i \boldsymbol{A}_i^T = \sum_{i=1}^{\ell} \frac{\gamma^2(\boldsymbol{u}_i)}{2} \boldsymbol{A}_i \boldsymbol{A}_i^T + \sum_{i=1}^{\ell} (\boldsymbol{v}_i \boldsymbol{v}_i^T - \boldsymbol{w}_i \boldsymbol{w}_i^T),
$$

where $\boldsymbol{v}_i = \boldsymbol{A}_i \boldsymbol{u}_i$, $\boldsymbol{w}_i = \gamma(\boldsymbol{u}_i) A_{i1}$, and $\boldsymbol{F}_i = (\gamma^2(\boldsymbol{u}_i)/2)\boldsymbol{I}_i + \boldsymbol{u}_i \boldsymbol{u}_i^T - \gamma^2(\boldsymbol{u}_i) I_{i1} (I_{i1})^T$. Here $\boldsymbol{I}_i$ is the $k_i \times k_i$ identity matrix whose first column is denoted I_{i1}. For $i \in \mathcal{I}$, $\boldsymbol{A}_i \boldsymbol{F}_i \boldsymbol{A}_i^T$ is splitted into the sparse and dense parts. More precisely, $\sum_{i=1}^{\ell} \boldsymbol{A}_i \widetilde{\boldsymbol{F}}_i \boldsymbol{A}_i^T + \sum_{i \in \mathcal{I}} (\boldsymbol{v}_i \boldsymbol{v}_i^T - \boldsymbol{w}_i \boldsymbol{w}_i^T)$ where $\widetilde{\boldsymbol{F}}_i = \boldsymbol{F}_i$ for $i \notin \mathcal{I}$ and $\widetilde{\boldsymbol{F}}_i = (\gamma^2(\boldsymbol{u}_i)/2)\boldsymbol{I}$ for $i \in \mathcal{I}$. Then, the sparse part $\sum_{i=1}^{\ell} \boldsymbol{A}_i \widetilde{\boldsymbol{F}}_i \boldsymbol{A}_i^T$ is factorized, and the rank-1 Fletcher-Powell update for the dense part

$\sum_{i\in\mathcal{I}}(\boldsymbol{v}_i\boldsymbol{v}_i^T - \boldsymbol{w}_i\boldsymbol{w}_i^T)$ is applied to the factorization. Notice that the rank of the dense part is $2\#\mathcal{I}$. As a result, increasing the number of second order cones of large dimensions in an SOCP formulation requires more applications of the rank-1 Fletcher-Powell update to the factorization. SeDuMi uses a threshold value to decide how large size of a second order cone is regarded as large.

Illustrative example

Three different SOCP formulations of the Chained singular function in Section 3 result in different computational performance. We investigate their difference with the size of the Schur complement matrix, the csp matrix of $\boldsymbol{A}$, and the sparsity of the sparse part of Schur complement matrix.

Consider the Chained singular function with $n = 4$. For SOCP formulation (8), we define the vector $\boldsymbol{y}$ in (12) by $\boldsymbol{y} = (y_1, y_2, \ldots, y_{10}) = (x_1, x_2, x_3, x_4, s_1, t_1, r_1, p_1, u_1, q_1)$. The 10×10 csp matrix of the SOCP formulation (8) is shown in Figure 1. For the SOCP formulation

$$\begin{pmatrix}
* & * & & * & * & & & * & & \\
* & * & * & & * & & * & & & \\
 & * & * & * & & * & * & & & \\
* & & * & * & & * & & & * & \\
* & * & & & * & & & & & \\
 & & * & * & & * & & & & \\
 & * & * & & & & * & * & & \\
 & & & & & & * & * & & \\
* & & & * & & & & & * & * \\
 & & & & & & & & * & *
\end{pmatrix}$$

Figure 1: The csp matrix of (8)

(9), the vector $\boldsymbol{y}$ in (12) is defined by $\boldsymbol{y} = (y_1, y_2, \ldots, y_9) = (x_1, x_2, x_3, x_4, s_1, r_1, p_1, u_1, q_1)$, and the 9×9 csp matrix of (9) is shown in Figure 2. For the SOCP formulation (10), the vector $\boldsymbol{y}$ in (12) is defined by $\boldsymbol{y} = (y_1, y_2, \ldots, y_7) = (x_1, x_2, x_3, x_4, s_1, r_1, u_1)$. In this SOCP formulation, all elements in $\boldsymbol{y}$ are included in the first cone, and thus the 7×7 csp matrix becomes completely dense. Its csp matrix is shown in Figure 3.

$$\begin{pmatrix}
* & * & * & * & * & & & & \\
* & * & * & * & * & & & * & \\
* & * & * & * & * & * & & & \\
* & * & * & * & * & * & & & \\
* & * & * & * & * & & & * & \\
 & & * & * & & * & * & & \\
 & & & & & * & * & & \\
 & * & & & * & & & * & * \\
 & & & & & & & * & *
\end{pmatrix}$$

Figure 2: The csp matrix of (9)

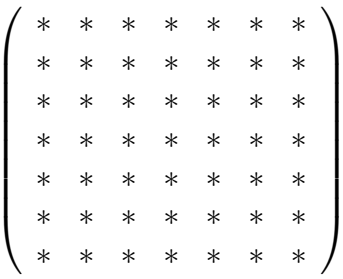

Figure 3: The csp matrix of (10)

As we see from Figure 1, 2 and 3, the sparsity decreases from the csp matrix of (8), (9) to (10). More precisely, as the size of the largest second order cone in the SOCP formulation increases, the csp matrix gradually loses sparsity. Note that in (9), the first second order cone inequality contains x_1, x_2, x_3, x_4, and s and this inequality makes the 5×5 submatrix of the csp matrix to be completely dense. Moreover, the first second order cone inequality in (10) contains $x_1, x_2, x_3, x_4, s, r_1$, and u_1 which makes 7×7 submatrix of the csp matrix completely dense. In SeDuMi, the nonzero pattern of the sparse part of the Schur complement matrix of (8) is equivalent to the csp matrix of (8) shown in Fig 1. For (9) and (10), when a second order cone involving most variables is considered large in SeDuMi, that is, $\#\mathcal{I} = 1$, the sparsity pattern of the sparse part of the Schur complement matrix is shown in Figure 4 and 5, respectively. If we compare Figure 1, 4 and 5, we see that the size of the Schur complement matrix decreases from 10, 9 to 7, and the number of nonzero elements are reduced from 38, 29 to 23, and the sparsity pattern remains almost the same. As a result, the computational time for factorizing the sparse part is expected to decrease slightly or stay almost equal.

After factorizing the sparse part, the rank-1 Fletcher-Powell update is applied to the factorization to account for the dense part. As n increases, the size of the largest cone of SOCP formulations (9) and (10) grows, and that second order cone is regarded as large by SeDuMi. For (8), SeDuMi did not find any large second order cones, namely, $\#\mathcal{I} = 0$. For (9) and (10) with $n = 500, 1000$ and 2000, our test showed that SeDuMi determined the number of large second order cone $\#\mathcal{I} = 1$. This indicates that the rank of the dense part is 2. The computational time for applying the rank-1 update for the dense part is small and (10) has the smallest size of the Schur complement matrix among the three formulations. Consequently, solving (10) consumes the least amount of cpu time, as shown in the following numerical experiments.

The three SOCP formulations of the Chained singular function were tested with SeDuMi on a Macintosh Dual 2.5GHz PowerPC G5 with 2GB DDR SDRAM. In Tables 1-3, n means the number of variables, sizeA the size of SOCP problem in the SeDuMi input format, #nzA the number of nonzeros in the coefficient matrix A of SOCP problem to be solved by SeDuMi, #it the number of iterations, and rel.err the relative error of SOCP values.

Table 1, 2 and 3 show the numerical results of the SOCP formulation (8), (9) and (10). The asterisk mark in Table 1, 2, and 3 means that numerical difficulty was encountered while SeDuMi was solving the problem. Numerical problems in SeDuMi encountered while solving (10) may have caused an increase in the number of iterations and large rel.err. Currently, it is not well-understood why SeDuMi, an implementation of the primal-dual

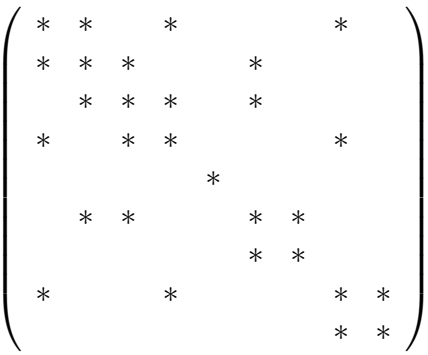

Figure 4: Sparse pattern of the sparse part for SOCP (9)

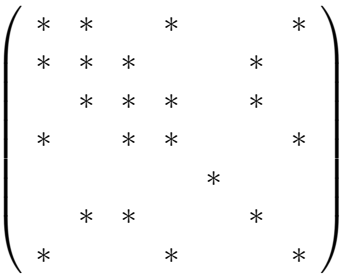

Figure 5: Sparsity pattern of the sparse part for SOCP (10)

Table 1: Numerical results of SOCP (8)

Chained Singular, SOCP (8)						
n	sizeA	#nzA	rel.err	cpu	#it	cpu/#it
500	[4482, 1994]	5478	5.7e-14	5.48	25	0.22
1000	[8982, 3994]	10978	0.0e-0	13.48	26	0.52
2000	[17982, 7994]	21978	0.0e-0	∗37.48	25	1.50

Table 2: Numerical results of SOCP (9)

Chained Singular, SOCP (9)						
n	sizeA	#nzA	rel.err	cpu	#it	cpu/#it
500	[3488, 1497]	4484	1.2e-7	3.47	18	0.19
1000	[6988, 2997]	8984	2.4e-7	7.14	18	0.40
2000	[13988, 5997]	17984	4.7e-7	19.06	18	1.06

Table 3: Numerical results of SOCP (10)

Chained Singular, SOCP (10)						
n	sizeA	#nzA	rel.err	cpu	#it	cpu/#it
500	[2492, 999]	3488	1.2e-6	∗3.26	23	0.14
1000	[4992, 1999]	6988	9.6e-5	∗5.89	24	0.25
2000	[9992, 3999]	13988	1.6e-4	∗14.15	28	0.51

interior-point method, gives numerical problems for some problems. We can only guess that computational aspects of the primal-dual interior-point methods employed by SeDuMi may contribute to numerical stability of SeDuMi. In this regard, we can not say that the result here indicates that (10) always cause numerical problems in SeDuMi. Further research on the numerical stability of SeDuMi is necessary.

In Table 3, the formulation (10) consumes the least amount of cpu time and cpu time per iteration, despite numerical difficulty which usually increases cpu time. Note that the size of A in (10) is smaller than (8) and (9), indicated by sizeA, and the numbers of nonzero elements shown in the column of #nzA in Table 3 are smaller than those in Tables 1 and 2. This resulted in the least amount of cpu time per iteration of (10).

As we already pointed out, the long-step algorithm for SOCP using NT direction has $O(k \log \epsilon^{-1})$ iteration-complexity to reduce the duality gap by a factor of ϵ, where k is the number of second order cones. The SOCP formulation in which the size of the largest second order cone is the largest among the various equivalent SOCP formulations is the formulation with the smallest number of second order cones among the various equivalent SOCP formulations. Thus, for computational efficiency, it is recommended to use this SOCP formulation. Besides the number of second order cones, the sparsity of the problem also affects the computational efficiency of the algorithm. As we have seen from these examples, we can observe that different SOCP formulations from various ways of introducing auxiliary variables do not change the sparsity pattern of the sparse part of the Schur complement matrix essentially. Moreover, our numerical results show that (10) provides the best computational efficiency. Thus it is recommended to use the SOCP formulation in which the size of the largest second order cone is the largest among the various equivalent SOCP formulations.

5. Numerical Results

We test SOCP formulations of convex optimization problems using SeDuMi and compare numerical results with LANCELOT. For unconstrained problems, we use the arwhead function, the engval1 function, nondquar function, the vardim function in addition to the Chained singular function in Section 4. Constrained test problems are generated by modifying some of the unconstrained problems and adding constraints because suitable constrained test problems of the type presented in this paper for the numerical experiments are unavailable.

Numerical results for unconstrained problems are presented in Section 5.1. Generating constrained test problems is described in Section 5.2 with numerical results. All the numerical tests were performed using the Matlab toolbox SeDuMi [14] for SOCP formulation, and using LANCELOT on a Macintosh Dual 2.5GHz PowerPC G5 with 2GB DDR SDRAM. We use the notation described in Table 4 for the description of numerical results.

5.1. Unconstrained problems

We selected minimization problems of the following functions for the test of unconstrained problems from CUTEr [4].

The arwhead function

$$\sum_{i=1}^{n-1}(-4x_i + 3.0) + \sum_{i=1}^{n-1}(x_i^2 + x_n^2)^2. \tag{13}$$

Table 4: Notation

n	the number of variables
sizeA	the size of SOCP problem in the SeDuMi input format
#nz	the number of nonzeros in the coefficient matrix A of SOCP problem for SeDuMi
cpu	cpu time consumed by SeDuMi and LANCELOT in seconds
#iter.	the number of iterations
gradNorm	the infinity norm of the gradient vector of the objective function at the obtained solution $\boldsymbol{x}$
infeasErr	$\max_i\{\max(g_i(\boldsymbol{x}), 0)\}$ where $g_i(\boldsymbol{x}) \leq 0$ $(i = 1, \ldots, m)$ are constraint and $\boldsymbol{x}$ is the obtained solution.
functVal	the objective function value at the end of iteration
s/f	S if num.err = 0 and pinf =dinf =0 from SeDuMi output. F if num.err = 1, or pinf=1, or dinf =1, or infeasErr>1.0e-2.
init.pt	initial guess for LANCELOT
inform	LANCELOT warning message; 0: successful, 1: maxit reached, 3: the step taken during the current iteration is so small that no difference will be observed in the function values, 5: insufficient space, 8: Check constraints and try starting with different initial values, 10: Some of internal functions are not large enough.

The engval1 function

$$\sum_{i=1}^{n-1}(-4x_i + 3.0) + \sum_{i=1}^{n-1}(x_i^2 + x_{i+1}^2)^2. \tag{14}$$

The nondquar function

$$\sum_{i=1}^{n-2}(x_i + x_{i+1} + x_n)^4 + (x_1 - x_2)^2 + (x_{n-1} + x_n)^2. \tag{15}$$

The vardim function

$$\sum_{i=1}^{n}(x_i - 1)^2 + (\sum_{i=1}^{n} ix_i - \frac{n(n+1)}{2})^2 + (\sum_{i=1}^{n} ix_i - \frac{n(n+1)}{2})^4. \tag{16}$$

Note that these functions belong to the class of $\mathbb{S}$epi and can thus be formulated as SOCPs. The numerical results of the problems are shown in Table 5, 6, 7, and 8, respectively. The SOCP formulation of the problems in which the size of the largest second order cone is the largest among various SOCP formulations is tested for all numerical experiment for computational efficiency as discussed in Section 4. In all Tables, we see that the matrix A when applying SeDuMi for solving SOCP is very sparse from the size of A and the number of nonzero elements of A shown in the columns of sizeA and #nzA, respectively.

The default values of the parameters in LANCELOT were used except the maximum number of iterations, which was increased to $200,000$. Input for LANCELOT for all the problems was prepared in SIF format. The column of "inform" shows information that

Table 5: Numerical results of minimizing the arwhead function using SeDuMi for SOCP and LANCELOT

SeDuMi for SOCP formulated with largest second order cone						
n	s/f	sizeA	#nzA	gradNorm	functVal	cpu
1000	S	$[4997, 2000]$	4997	1.41e-4	9.75e-7	1.26
5000	S	$[24997, 10000]$	24997	2.13e-4	9.24e-6	13.42
LANCELOT						
n	inform	#iter.	init.pt	gradNorm	functVal	cpu
1000	0	5	$x_i = 1.0\ \forall i$	2.01e-6	1.69e-10	0.28
1000	0	14	$x_i = 10.0\ \forall i$	2.55e-13	0.0e-0	0.67
5000	0	5	$x_i = 1.0\ \forall i$	1.98e-7	1.11e-12	0.81
5000	0	14	$x_i = 10.0\ \forall i$	1.85e-13	0.0e-0	1.41

LANCELOT returns after it finishes solving the problems, and the meaning of each value is included in Table 4.

In Table 5, we see that solving with LANCELOT provides more accurate solutions faster than using SeDuMi for SOCP formulations. For the engval1 function, LANCELOT obtained optimal objective function values faster for $n = 1000$, however, in the case of $n = 10000$ and $n = 15000$ with init.pt 10.0, it consumed more cpu time than SeDuMi as indicated in Table 6.

Table 6: Numerical results of minimizing the engval1 function using SeDuMi for SOCP and LANCELOT

SeDuMi for SOCP formulated with largest second order cone						
n	s/f	sizeA	#nzA	gradNorm	functVal	cpu
1000	S	$[4997, 2000]$	4997	2.35e-4	1.10819e+4	2.02
10000	S	$[49997, 20000]$	49997	2.27e-4	1.10993e+4	25.10
15000	S	$[74997, 30000]$	74997	1.95e-4	1.66499e+4	28.20
LANCELOT						
n	Inform	#iter.	init.pt	gradNorm	functVal	cpu
1000	0	7	$x_i = 2.0\ \forall i$	2.47e-6	1.10819e+4	0.26
1000	0	13	$x_i = 10.0\ \forall i$	7.50e-12	1.10819e+4	0.63
10000	0	7	$x_i = 2.0\ \forall i$	2.47e-6	1.10993e+4	1.34
10000	3	13747	$x_i = 10.0\ \forall i$	1.66e-7	1.10993e+4	274.47
15000	0	7	$x_i = 2.0\ \forall i$	2.47e-6	1.66499e+4	1.94
15000	3	10152	$x_i = 10.0\ \forall i$	1.23e-7	1.66499e+4	324.79

The numerical results for the nondquar function are presented in Table 7. It took longer to solve with SeDuMi than LANCELOT. We observe that the performance of LANCELOT is better in terms of cpu time for both case of initial points $\boldsymbol{x} = (1, -1, 1, -1, \ldots)$ and $x_i = 10.0$ for all i.

In Table 8, the numerical results for the vardim function are shown. When LANCELOT ended in inform 3 for solving the vardim function for $n = 10000, 15000$, the values of the column of gradNorm are large compared to terminating with inform 0. We observe that the function values are small for these cases. LANCELOT spent more cpu time with

Table 7: Numerical results of minimizing the nondquar function using SeDuMi for SOCP and LANCELOT

SeDuMi for SOCP formulated with largest second order cone						
n	s/f	sizeA	#nzA	gradNorm	functVal	cpu
1000	S	[4994, 1999]	5994	4.81e-5	1.68e-5	2.35
10000	S	[49994, 19999]	59994	9.03e-3	3.16e-3	31.43
50000	S	[249994, 99999]	299994	1.03e+1	1.34e-1	227.22
LANCELOT						
n	inform	#iter.	init.pt	gradNorm	functVal	cpu
1000	0	17	$\boldsymbol{x}$ =(1,-1,1,-1,...)	8.47e-6	1.39e-9	0.18
1000	0	28	$x_i = 10.0\ \forall i$	0.0e-0	9.31e-10	0.49
10000	0	19	$\boldsymbol{x}$ =(1,-1,1,-1,...)	6.61e-6	5.23e-10	1.17
10000	0	29	$x_i = 10.0\ \forall i$	1.15e-15	7.40e-8	1.13
50000	0	20	$\boldsymbol{x}$ =(1,-1,1,-1,...)	5.97e-6	4.16e-10	5.61
50000	0	31	$x_i = 10.0\ \forall i$	6.87e-15	6.07e-10	7.66

Table 8: Numerical results of minimizing the vardim function using SeDuMi and LANCELOT

SeDuMi for SOCP formulated with largest second order cone						
n	s/f	sizeA	#nzA	gradNorm	functVal	cpu
1000	S	[1008, 1002]	3005	1.55e-2	2.79e-10	0.29
10000	S	[10008, 10002]	30005	8.60e-3	9.22e-7	3.3
15000	S	[15008, 15002]	45005	3.16e-2	1.55e-6	6.47
LANCELOT						
n	inform	#iter.	init.pt	gradNorm	functVal	cpu
1000	0	36	$x_i = 1 - i/n\ \forall i$	1.89e-7	8.95e-21	1.0
1000	0	56	$x_i = 10.0\ \forall i$	5.74e-17	7.72e-14	1.78
10000	3	48	$x_i = 1 - i/n\ \forall i$	2.65e-3	1.77e-14	102.07
10000	3	61	$x_i = 10.0\ \forall i$	1.07e-3	3.78e-9	159.12
15000	3	50	$x_i = 1 - i/n\ \forall i$	1.63e-2	2.97e-13	237.67
15000	3	75	$x_i = 10.0\ \forall i$	1.18e-2	4.03e-9	412.28

init.pt $x_i = 10$ for all i. SeDuMi could find optimal objective function values faster for $n = 10000, 15000$. We see that the cpu time consumed by SeDuMi is smaller than that of LANCELOT, and LANCELOT needs a good initial point to have fast convergence.

5.2. Constrained problems

Constrained test problems for SOCP formulations presented in this paper are scarce in the literature. We generated test problems using some problems in CUTEr [4]. We first describe how the test problems were generated, and then present the numerical results of the problems. Main purpose of generating constrained test problems was to create problems with rational polynomials, square root of polynomials and nonlinear inequality constraints. The arwhead function, the engval1 function, and the nondquar function were modified for the objective function of the test problems, and the Chained singular function was used to create constraints.

We create 6 test problems, which are called **P1, P2, P3, P4, P5**, and **P6**. In the description of test problems, we use $J = \{1, 3, 5, \ldots, n-3\}$, $g_i(\boldsymbol{x}) = (x_i + 10x_{i+1})^2 + 5(x_{i+2} - x_{i+3})^2 + (x_{i+1} - 2x_{i+2})^4 + 10(x_i - 10x_{i+3})^4$ $(i \in J)$, where n is a multiple of 4. The constraints in the form of $g_i(\boldsymbol{x} - 2\boldsymbol{e}) \leq \rho$, where $\boldsymbol{e} = (1, 1, \ldots, 1)^T \in \mathbb{R}^n$, are included in the test problems. Here ρ denotes a parameter, which will be fixed to 1000 for numerical tests. We use a large number 1000 for ρ because the value of $g_i(\boldsymbol{x})$ tends to be large for even small values of x_i's. Using the arwhead function (13), test problems **P1** and **P2** are generated as follows.

$$\left.\begin{array}{lll} \mathbf{P1}: & \min & \displaystyle\sum_{i=1}^{n-1}(-4x_i + 3.0) + \sum_{i=1}^{n-1}(x_i^2 + x_n^2)^2 \\ & \text{subj. to} & g_j(\boldsymbol{x} - 2\boldsymbol{e}) \leq \rho \quad (j \in J). \end{array}\right\}$$

$$\left.\begin{array}{lll} \mathbf{P2}: & \min & \displaystyle\left(\sum_{k=1}^{n-1}(-4x_k + 3.0)^2\right)^{1/2} + \sum_{i=1}^{n-1}\frac{(x_i^2 + x_n^2)^2}{1 + x_i + x_n} \\ & \text{subj. to} & g_j(\boldsymbol{x} - 2\boldsymbol{e}) \leq \rho \quad (j \in J), \ \boldsymbol{x} \in K, \end{array}\right\}$$

where $K = \{\boldsymbol{x} \in \mathbb{R}^n : x_i \geq 0 \ (i = 1, 2, \ldots, n)\}$.

Constrained test problems **P3** and **P4** are derived using the engval1 function.

$$\left.\begin{array}{lll} \mathbf{P3}: & \min & \displaystyle\sum_{i=1}^{n-1}(-4x_i + 3.0) + \sum_{i=1}^{n-1}(x_i^2 + x_{i+1}^2)^2 \\ & \text{subj. to} & g_j(\boldsymbol{x} - 2\boldsymbol{e}) \leq \rho \ (j \in J). \end{array}\right\}$$

$$\left.\begin{array}{lll} \mathbf{P4}: & \min & \displaystyle\sum_{i=1}^{n-1}(-4x_i + 3.0) + \sum_{i=1}^{n-1}(x_i^2 + x_{i+1}^2)^2 \\ & & \displaystyle - \sum_{i=1}^{n-1}(1 + x_i + x_{i+1})^{1/4}\left(1 + \frac{x_i}{2} + \frac{x_{i+1}}{i+1}\right)^{1/2} \\ & \text{subj. to} & g_j(\boldsymbol{x} - 2\boldsymbol{e}) \leq \rho \quad (j \in J), \ \boldsymbol{x} \in K, \end{array}\right\}$$

where $K = \{\boldsymbol{x} \in \mathbb{R}^n : x_i \geq 0 \ (i = 1, 2, \ldots, n)\}$.

Using the nondquar function (15), we obtain test problems **P5** and **P6**.

$$\left.\begin{array}{lll} \mathbf{P5}: & \min & \displaystyle\sum_{i=1}^{n-2}(x_i+x_{i+1}+x_n)^4+(x_1-x_2)^2+(x_{n-1}-x_n)^2 \\ & \text{subj. to} & g_j(\boldsymbol{x}-2\boldsymbol{e})\le\rho \ \ (j\in J). \end{array}\right\}$$

$$\left.\begin{array}{lll} \mathbf{P6}: & \min & \displaystyle\sum_{i=1}^{n-2}\frac{(x_i+x_{i+1}+x_n)^4}{1+x_i+x_{i+1}+x_n}+\frac{(x_1-x_2)^2}{1+x_1+x_2}+\frac{(x_{n-1}-x_n)^2}{1+x_{n-1}+x_n} \\ & \text{subj. to} & g_j(\boldsymbol{x}-2\boldsymbol{e})\le\rho \ \ (j\in J),\ \boldsymbol{x}\in K, \end{array}\right\}$$

where $K=\{\boldsymbol{x}\in\mathbb{R}^n : x_i\ge 0\ (i=1,\ldots,n)\ \}$.

Notice that $g_j\in\mathbb{S}\text{epi}$ in **P1, P2, P3, P4, P5** and **P6**, and the constraints $g_j(\boldsymbol{x}-2\boldsymbol{e})\le\rho\,(j\in J)$ can be formulated as second order cone inequalities. We see the objective functions of **P1, P3** and **P5**, the constraint functions $g_j(\boldsymbol{x})$ are in the class of $\mathbb{S}\text{epi}$. Also, the objective functions of **P2, P4, P6** are in the class of $\mathbb{S}\text{epi}(K)$. As a result, the problem **P1, P2, P3, P4, P5** and **P6** can be formulated as SOCPs.

We show SOCP formulations of **P2** and **P4**. Since the other problems have resembling terms in the objective function and the same constraints as **P2** and **P4**, their SOCP formulations can be derived similarly.

The problems **P2** and **P4** have a common constraint $g_j(\boldsymbol{x}-2\boldsymbol{e})\le\rho\ \ (j\in J)$. We show how the second order cone inequalities of the constraint can be described. Consider

$$\left.\begin{array}{l} \rho\ge\left((x_j-2)+10\,(x_{j+1}-2)\right)^2+5\left((x_{j+2}-2)-(x_{j+3}-2)\right)^2+q_j^2+r_j^2, \\ q_j\ge\left((x_{j+1}-2)-2\,(x_{j+2}-2)\right)^2,\ r_j\ge\sqrt{10}\left((x_j-2)-10\,(x_{j+3}-2)\right)^2\ \ (j\in J). \end{array}\right\} \quad (17)$$

Then, the SOCP formulation of (17) can be written as

$$\left.\begin{array}{l} \begin{pmatrix} \rho+1 \\ \rho-1 \\ 2\left\{(x_j-2)+10\,(x_{j+1}-2)\right\} \\ 2\sqrt{5}\left\{(x_{j+2}-2)-(x_{j+3}-2)\right\} \\ 2q_j \\ 2r_j \end{pmatrix}\succeq_{\mathbb{S}}\mathbf{0}, \\ \begin{pmatrix} q_j+1 \\ q_j-1 \\ 2\left\{(x_{j+1}-2)-2\,(x_{j+2}-2)\right\} \end{pmatrix}\succeq_{\mathbb{S}}\mathbf{0}, \\ \begin{pmatrix} r_j+1 \\ r_j-1 \\ 2\sqrt[4]{10}\left\{(x_j-2)-10\,(x_{j+3}-2)\right\} \end{pmatrix}\succeq_{\mathbb{S}}\mathbf{0}\ \ (j\in J). \end{array}\right\} \quad (18)$$

The problem **P2** is equivalent to the problem

$$\left.\begin{array}{ll} \min & s+\displaystyle\sum_{i=1}^{n-1}t_i \\ \text{subj. to} & s\ge\left(\displaystyle\sum_{k=1}^{n-1}(-4x_k+3.0)^2\right)^{1/2},\ t_i\ge\dfrac{u_i^2}{1+x_i+x_n}, \\ & u_i\ge x_i^2+x_n^2\ (i=1,2,\ldots,n-1), \\ & \text{Second order cone inequalities in (18)}, \\ & x_i\ge 0\ \ (i=1,2,\ldots,n). \end{array}\right\} \quad (19)$$

The SOCP formulation of **P2** with the largest size of second order cone is

$$\left.\begin{array}{ll} \min & s+\sum_{i=1}^{n-1} t_i \\ \text{subj. to} & \begin{pmatrix} s \\ -4x_1+3.0 \\ -4x_2+3.0 \\ \vdots \\ -4x_{n-1}+3.0 \end{pmatrix} \succeq_{\mathbb{S}} \mathbf{0}, \begin{pmatrix} t_i+1+x_i+x_n \\ t_i-1-x_i-x_n \\ 2u_i \end{pmatrix} \succeq_{\mathbb{S}} \mathbf{0}, \\ & \begin{pmatrix} u_i+1 \\ u_i-1 \\ 2x_i \\ 2x_n \end{pmatrix} \succeq_{\mathbb{S}} \mathbf{0} \ (i=1,2,\ldots,n-1), \\ & \text{Second order cone inequalities in (18),} \\ & x_i \geq 0 \ \ (i=1,2,\ldots,n). \end{array}\right\}$$

The problem **P4** is equivalent to the problem

$$\left.\begin{array}{ll} \min & \sum_{i=1}^{n-1}(-4x_i+3.0)+t+\sum_{i=1}^{n-1}(-u_i) \\ \text{subj. to} & t \geq \sum_{i=1}^{n-1} v_i^2, \quad v_i \geq x_i^2+x_{i+1}^2, \\ & u_i^2 \leq w_i\left(1+\frac{x_i}{2}+\frac{x_{i+1}}{i+1}\right), \ w_i^2 \leq 1+x_i+x_{i+1} \ (i=1,2,\ldots,n), \\ & \text{Second order cone inequalities in (18),} \\ & x_i \geq 0 \ \ (i=1,2,\ldots,n-1). \end{array}\right\}$$

The SOCP formulation of **P4** with the largest size of second order cone is

$$\left.\begin{array}{ll} \min & \sum_{i=1}^{n-1}(-4x_i+3.0)+t+\sum_{i=1}^{n-1}(-u_i) \\ \text{subj. to} & \begin{pmatrix} t+1 \\ t-1 \\ 2v_1 \\ 2v_2 \\ \vdots \\ 2v_{n-1} \end{pmatrix} \succeq_{\mathbb{S}} \mathbf{0}, \begin{pmatrix} v_i+1 \\ v_i-1 \\ 2x_i \\ 2x_{i+1} \end{pmatrix} \succeq_{\mathbb{S}} \mathbf{0}, \\ & \begin{pmatrix} w_i+1+\frac{x_i}{2}+\frac{x_{i+1}}{i+1} \\ w_i-1-\frac{x_i}{2}-\frac{x_{i+1}}{i+1} \\ 2u_i \end{pmatrix} \succeq_{\mathbb{S}} \mathbf{0}, \\ & \begin{pmatrix} 1+x_i+x_{i+1}+1 \\ 1+x_i+x_{i+1}-1 \\ 2w_i \end{pmatrix} \succeq_{\mathbb{S}} \mathbf{0}, \ (i=1,2,\ldots,n) \\ & \text{Second order cone inequalities in (18),} \\ & x_i \geq 0 \ \ (i=1,2,\ldots,n-1). \end{array}\right\}$$

Table 9: Numerical results from solving **P1** using SeDuMi and LANCELOT

SeDuMi						
n	s/f	sizeA	#nzA	infeasErr	functVal	cpu
1000	S	[10986, 2999]	12982	5.53e-4	1.0293e+4	7.15
5000	S	[54986, 14999]	64982	8.29e-4	5.1297e+4	53.12
LANCELOT						
n	inform	#iter.	init.pt	infeasErr	functVal	cpu
1000	3	2165	$x_i = 1.0\ \forall i$	8.25e-8	1.0293e+4	22.28
1000	3	33455	$x_i = 10.0\ \forall i$	5.26e-7	1.0293e+4	190.39
5000	3	2548	$x_i = 1.0\ \forall i$	9.31e-9	5.1297e+4	1114.93
5000	3	27201	$x_i = 10.0\ \forall i$	2.28e-6	5.1299e+4	1255.33

Table 10: Numerical results from solving **P2** using SeDuMi and LANCELOT

SeDuMi						
n	s/f	sizeA	#nzA	infeasErr	functVal	cpu
1000	S	[13982, 3998]	19974	2.94e-4	3.0860e+3	53
5000	S	[69982, 19998]	99974	1.17e-4	1.5117e+4	55.42
LANCELOT						
n	inform	#iter.	init.pt	infeasErr	functVal	cpu
1000	3	11805	$x_i = 1.0\ \forall i$	1.62e-6	3.0860e+3	55.99
1000	3	83580	$x_i = 10.0\ \forall i$	3.28e-7	3.1100e+3	402.53
5000	1	200000	$x_i = 1.0\ \forall i$	-	1.5117e+4	7534.52
5000	3	168780	$x_i = 10.0\ \forall i$	1.80e-6	1.5137e+4	6855.94

The numerical results for the problems **P1** and **P2** are shown in Tables 9 and 10, respectively. In Table 9, we notice that LANCELOT took more cpu time to compute optimal objective function values than SeDuMi. The objective function values obtained by LANCELOT with init.pt 10.0 are larger than the ones obtained with init.pt 1.0 for $n = 5000$.

From Table 10, we see that SeDuMi provided the optimal objective function values faster than LANCELOT. If init.pt 10.0 is used for LANCELOT, it took longer time to attain slightly larger values for the objective function values than with init.pt 1.0. For $n = 5000$ and init.pt = 1, LANCELOT reached the maximum number of iterations, but the objective function value was as good as the one obtained by SeDuMi.

In Table 11, solving the SOCP formulation of **P3** with SeDuMi and solving **P3** with LANCELOT for $n = 5000, 10000$, and 15000 resulted in the same objective function values. We note that the cpu time consumed by SeDuMi was less than LANCELOT using init.pt=10.0.

Table 11: Numerical results from solving **P3** with SeDuMi and LANCELOT

SeDuMi						
n	s/f	sizeA	#nzA	infeasErr	functVal	cpu
5000	S	[54986, 14999]	64982	1.06e-4	2.5023e+4	19.3
10000	S	[109986, 29999]	129982	1.17e-3	5.0056e+4	70.4
15000	S	[164986, 44999]	194982	9.20e-4	7.5089e+4	84.9
LANCELOT						
n	inform	#iter.	init.pt	infeasErr	functVal	cpu
5000	3	220	$x_i = 1.0\ \forall i$	9.76e-9	2.5024e+4	22.20
5000	3	9794	$x_i = 10.0\ \forall i$	6.09e-9	2.5024e+4	321.87
10000	3	160	$x_i = 1.0\ \forall i$	1.84e-9	5.0056e+4	26.66
10000	3	10044	$x_i = 10.0\ \forall i$	5.48e-11	5.0056e+4	630.58
15000	3	2503	$x_i = 1.0\ \forall i$	1.04e-6	7.5089e+4	24.00
15000	3	9779	$x_i = 10.0\ \forall i$	1.19e-8	7.5089e+4	943.55

In Table 12, solving the SOCP formulation of **P4** with SeDuMi is shown to be a better approach than LANCELOT for accuracy and getting smaller objective function values. For $n = 15000$, LANCELOT resulted in insufficient memory.

From Table 13 for **P5** and Table 14 for **P6**, we notice that large infeasible errors were obtained by LANCELOT as shown in the column of infeasErr in the tables. We also tested for small-sized problem such as $n = 8$, the same objective function values were obtained from SeDuMi and LANCELOT for both **P5** and **P6**. However, with increasing n, LANCELOT failed to attain optimal objective function values while SeDuMi provided optimal objective function values with small infeasible errors.

In unconstrained test problems, LANCELOT performed faster if initial points were given close to the optimal solution. Otherwise, optimal solutions were attained faster by solving the SOCP formulation with SeDuMi. Numerical experiments for constrained problems show that using SeDuMi for the SOCP formulation of the problems provides optimal objective function values in less cpu time except for **P3**. LANCELOT failed to obtain an optimal objective function value in some cases. We mention LANCELOT is a local optimizer while SeDuMi based on primal-dual interior-point methods is a global optimizer, thus, does not depend on initial points for convergence. Good initial points are necessary to have conver-

Table 12: Numerical results from solving **P4** with SeDuMi and LANCELOT

SeDuMi						
n	s/f	sizeA	#nzA	infeasErr	functVal	cpu
5000	S	[84980, 24997]	124970	9.25e-4	3.3338e+3	72.45
10000	S	[169980, 49997]	249970	8.63e-4	6.6790e+3	199.84
15000	S	[254980, 74997]	374970	2.29e-4	1.0024e+4	345.19
LANCELOT						
n	inform	#iter.	Init.pt	infeasErr	functVal	cpu
5000	3	244	$x_i = 1.0\ \forall i$	5.32e-2	2.5041e+4	21.38
5000	3	59749	$x_i = 10.0\ \forall i$	2.32e-2	8.9121e+4	3718.86
10000	3	245	$x_i = 1.0\ \forall i$	3.79e-2	5.0091e+4	26.67
10000	3	533	$x_i = 10.0\ \forall i$	3.38e-2	1.7768e+5	2698.54
15000	5	-		-	-	-

Table 13: Numerical results from solving **P5** with SeDuMi and LANCELOT

SeDuMi						
n	s/f	sizeA	#nzA	infeasErr	functVal	cpu
5000	S	[49985, 14998]	69979	2.69e-4	8.1213e+4	54.34
10000	S	[99985, 29998]	139979	2.34e-4	1.6179e+5	124.48
15000	F	[149985, 44998]	209979	2.07e-4	2.4220e+5	$*$192.52
LANCELOT						
n	inform	#iter.	Init.pt	infeasErr	functVal	cpu
5000	3	2120	$x_i = 1.0\ \forall i$	5.10e-1	1.9877e+5	2.88
5000	3	2447	$x_i = 10.0\ \forall i$	7.84e-0	9.5439e+5	37.21
10000	3	2869	$x_i = 1.0\ \forall i$	2.96e+1	1.6179e+5	4774.05
10000	3	3537	$x_i = 10.0\ \forall i$	1.18e+3	1.6822e+7	8630.30
15000	3	2974	$x_i = 1.0\ \forall i$	9.39e+1	7.5836e+6	8029.10
15000	3	2900	$x_i = 10.0\ \forall i$	2.21e+1	2.3696e+6	327.51

Table 14: Numerical results from solving **P6** with SeDuMi and LANCELOT

SeDuMi						
n	s/f	sizeA	#nzA	infeasErr	functVal	cpu
5000	S	[59983,11997]	109973	3.17e-4	2.6941e+4	40.84
10000	S	[119983,39997]	219973	1.95e-4	5.3729e+4	94.97
15000	S	[179983,59997]	329973	1.96e-4	8.0476e+4	196.08
LANCELOT						
n	Inform	#iter.	init.pt	infeasErr	functVal	cpu
5000	10	731	$x_i = 1.0\ \forall i$	3.33e-1	1.6260e+5	446.28
5000	3	6339	$x_i = 10.0\ \forall i$	4.33e+0	2.4236e+6	191.52
10000	3	5256	$x_i = 1.0\ \forall i$	1.05e+0	2.9769e+5	14584.56
10000	10	4540	$x_i = 10.0\ \forall i$	1.95e+1	5.2815e+6	243.40
15000	3	3639	$x_i = 1.0\ \forall i$	1.50e-0	7.1146e+5	16404.76
15000	10	3735	$x_i = 10.0\ \forall i$	1.47e+2	8.0147e+6	303.33

gence to the optimal solution with LANCELOT. The approach using SOCP is effective when good initial points are not available. We have observed in the numerical experiments that SOCP is effective in solving constrained problems.

6. Concluding Discussions

We have shown that convex optimization problems of SOCI representable functions can be formulated as SOCPs in various ways. The computational efficiency of solving the Schur complement equation in the primal-dual interior-point methods depends on the SOCP formulation. Introducing a smaller-number of auxiliary variables when formulating a convex optimization problem as an SOCP provides a smaller-sized Schur complement matrix. In terms of sparsity, the sparsity pattern of the sparse part of the Schur complement matrix remains almost the same for various SOCP formulations. Therefore, if the rank of the dense part of the Schur complement matrix, which is determined by SeDuMi, is small, then, the computational efficiency increases by introducing a minimum number of the auxiliary variables.

Numerical experiments shown in Section 5 demonstrate that SOCP formulations can be solved efficiently by SeDuMi compared with LANCELOT when good initial points are not available. Solving the SOCP formulation by SeDuMi is shown to be more effective to obtain better optimal values than LANCELOT for the constrained test problems.

References

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

[2] A.R. Conn, N.I.M. Gould, and Ph.L. Toint: LANCELOT, *A Fortran Package for Large-Scale Nonlinear Optimization (Release A)* (Springer, Heidelberg, 1992).

[3] D. Goldfarb and K. Scheinberg: Product-form Cholesky factorization in interior-point methods for second order cone programming. *Mathematical Programming*, **103** (2005), 153–179.

[4] N.I.M. Gould, D. Orban, and Ph.L. Toint: Cuter, a constrained and unconstrained testing environment, revisited. *Transactions on Mathematical Software*, **29** (2003), 373–394.

[5] A. Griewank and Ph.L. Toint: On the unconstrained optimization of partially separable functions. In M.J.D. Powell (eds.): *Nonlinear Optimization 1981* (Academic Press, New York, 1982), 301–312.

[6] S. Kim and M. Kojima: Second order cone programming relaxations of quadratic optimization problems. *Optimization Methods and Software*, **15** (2001), 201–224.

[7] S. Kim and M. Kojima: Exact solutions of some nonconvex quadratic optimization problems via SDP and SOCP relaxations. *Computational Optimization and Applications*, **26** (2003), 143–154.

[8] S. Kim, M. Kojima, and H. Waki: Generalized Lagrangian duals and sums of squares relaxations of sparse polynomial optimization problems. *SIAM Journal on Optimization*, **15** (2005), 697–719.

[9] K. Kobayashi, S. Kim, and M. Kojima: Correlative sparsity in primal-dual interior point methods for LP, SDP, and SOCP. *Applied Mathematics and Optimization*, **58** (2008), 69–88.

[10] M. Kojima, S. Kim, and H. Waki: Sparsity in sums of squares of polynomials. *Mathematical Programming*, **103** (2005), 45–62.

[11] M. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret: Applications of second-order cone programming. *Linear Algebra and its Applications*, **284** (1998), 193–228, Special Issue on Linear Algebra in Control, Signals and Image Processing.

[12] A. Nemirovski: What can be expressed via conic quadratic and semidefinite programming?. August 1999, Talk presented at RUTCOR weekly seminar.

[13] Yu.E. Nesterov and A. Nemirovski: *Interior Point Polynomial Methods in Convex Programming: Theory and Applications* (SIAM, Philadelphia, 1994).

[14] F.J. Sturm: Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, **11 & 12** (1999), 625–653.

[15] F.J. Sturm: Implementation of interior point methods for mixed semidefinite and second order cone optimization. *Optimization Methods and Software*, **17** (2002), 1105–1154.

[16] K. Toh, M.J. Todd, and R.H. Tütüntü: SDPT3 — a MATLAB software package for semidefinite programming. Dept. of Mathematics, National University of Singapore (1998), Singapore.

[17] T. Tsuchiya: A convergence analysis of the scaling-invariant primal-dual path-following algorithms for second-order cone programming. *Optimization Methods and Software*, **11 & 12** (1999), 141–182.

[18] H. Waki, S. Kim, M. Kojima, and M. Muramatsu: Sums of squares and semidefinite programming relaxations for polynomial optimization problems with structured sparsity. *SIAM Journal on Optimization*, **17** (2006), 218–242.

[19] E. Andersen: MOSEK. http://www.mosek.com/.

Kazuhiro Kobayashi
Department of Mathematical and Computing Sciences,
Tokyo Institute of Technology,
2-12-1 Oh-Okayama, Meguro-ku, Tokyo 152-8552 Japan.
E-mail: `kazuhir2@gmail.com`