A second-order cone cutting surface method: complexity and application

Mohammad R. Oskoorouchi[*]
College of Business Administration,
California State University San Marcos,
San Marcos, CA 92096.
moskooro@csusm.edu

John E. Mitchell[†]
Department of Mathematical Sciences,
Rensselaer Polytechnic Institute,
Troy, NY 12180.
mitchj@rpi.edu

August 7, 2006

Abstract

We present an analytic center cutting surface algorithm that uses mixed linear and multiple second-order cone cuts. Theoretical issues and applications of this technique are discussed. From the theoretical viewpoint, we derive two complexity results. We show that an approximate analytic center can be recovered after simultaneously adding p second-order cone

[*]This work has been completed with the support of the partial research grant from the College of Business Administration, California State University San Marcos, and the University Professional Development Grant.

[†]This material is based upon work supported by the National Science Foundation under Grant No. 0317323. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

cuts in $O(p\log(p+1))$ Newton steps, and that the overall algorithm is polynomial. From the application viewpoint, we implement our algorithm on mixed linear-quadratic-semidefinite programming problems with bounded feasible region and report some computational results on randomly generated fully dense problems. We compare our cpu time with that of SDPLR, SDPT3, and SeDuMi and show that our algorithm outperforms these software packages on fully dense problems. We also show the performance of our algorithm on semidefinite relaxation of the maxcut and Lovasz theta problems.

Keywords: Second-order cone, semidefinite inequality, cutting plane techniques, semidefinite programming.

1 Introduction

The analytic center cutting plane method (ACCPM) is an efficient technique for nondifferentiable optimization problems. The method was first introduced by Sonnevend [28] in 1988. Theoretical issues of this method have been studied in the literature for several settings. The main difference in all methods is the geometry of cuts. In polyhedral cases, single linear, multiple linear, and quadratic cuts have been studied. The theoretical complexity of the method has been reported in several papers. See for instance, Ye [35], Atkinson and Vaidya [2], and Goffin, Luo and Ye [9] for single linear cuts, Ye [36] and Goffin and Vial [10] for multiple linear case, and Luo and Sun [18], Lüthi and Büeler [20], and Sharifi Mokhtarian and Goffin [26] for the case of quadratic cuts, and Luo and Sun [19] for self-concordant inequalities.

Recently, there has been a growing interest in ACCPM incorporated with nonpolyhedral models, aka, analytic center cutting surface method (ACCSM). Complexity results of ACCSM have been studied by several authors. Sun, Toh, and Zhao [31], Toh, Zhao, Sun [33], and Chua, Toh, and Zhao [8] examine the addition of linear cuts for semidefinite programs, Oskoorouchi and Goffin [22] derive the results for semidefinite cuts, and Oskoorouchi and Goffin [23] discuss second-order cone cuts. Krishnan and Mitchell [15] give a unifying framework for cutting surface approaches for semidefinite programming. Recently Basescu and Mitchell [3] presented a complexity framework that covers general conic programming.

ACCSM has been implemented in practice for various applications. Oskoorouchi and Goffin [24] implement an ACCSM with semidefinite cuts (SDC) to solve eigenvalue optimization problem, Krishnan and Mitchell [16] use a cut-and-price approach based on a polyhedral approximation to solve the maxcut problem to optimality, and Sivaramakrishnan et al. [27] apply a decomposition approach based on ACCSM for semidefinite relaxations of some combinatorial optimization problems. All of the above papers report interesting numerical results.

This paper contributes to the theory and application of ACCSM in the following ways: we first explore the theoretical issues of integrating mixed linear cuts (LC) and multiple second-order cone cuts (SOCC) with ACCSM and introduce a second-order cone cutting surface method (SOCCSM). We generalize the complexity results derived in [23] from single SOCC to multiple SOCC's. We derive two complexity results: the complexity of recentering after simultaneously adding $p(\geq 1)$ SOCCs and the complexity of the overall algorithm. In the implementation part, we replace a p-dimensional SDC by multiple SOCC's and illustrate that this replacement improves the computational results presented in [24]. We implement our algorithm on mixed linear-quadratic-semidefinite programming problems with bounded feasible region and test our algorithm with various randomly generated problems and compare our numerical results with that of SDPLR, due to Burer and Monteiro [6, 7] and Burer and Choi [5]; SDPT3-4.0-beta, due to Toh et al. [32] and Tutuncu et al. [34]; and SeDuMi, due to Sturm [29, 30]. By comparing our cpu time with the above-mentioned software packages we illustrate that SOCCSM is a very efficient technique for moderate to large size problems with fully dense coefficient matrices.

2 Preliminaries

Throughout this paper we extensively use some well-known characteristics of second-order cone programming. To keep the paper self-contained, we briefly review the most important properties of the second-order cone. More comprehensive analysis can be found in Alizadeh and Goldfarb [1].

First we introduce our notation: We use uppercase letters for matrices, lowercase letters for vectors and Greek letters for scalars. The space of $n-$dimensional symmetric matrices is denoted by $\mathcal{M}^n$, positive semidefinite matrices by $\mathcal{M}^n_+$, $n-$dimensional real vectors by $I\!R^n$, and nonnegative real vectors by $I\!R^n_+$. We also use $A \succeq 0$ to indicate that $A \in \mathcal{M}^n_+$. We use $\mathbf{1}$ for an all one vector, and $\mathbf{1}_i$ for a vector with 1 in the ith position and zero elsewhere.

For $x, s \in I\!\!R^n$, we use the following notation: "$x.s$" is a component-wise product of x_i and s_i, that is $(x.s)_i = x_i s_i$; $x.^{-1}$ is the component-wise inverse of x, and $\prod x = \prod_{i=1}^n x_i$. We use ";" for joining two vectors in a column, i.e., $(x;s)$ is a vector in $I\!\!R^{2n}$ made up of vectors x and s joined in a column.

For two matrices A and B, (A, B) makes a matrix by joining them in rows, and $A \oplus B$ makes a matrix by joining A and B in the diagonal

$$A \oplus B = \begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix};$$

$\|A\|_F$ is the Frobenius norm of A defined via $\|A\|_F = tr(A^T A)$, where "tr" adds the diagonal elements of a symmetric matrix; and finally $A \bullet B$ is the inner product of A and B defined via $A \bullet B = tr(A^T B)$.

The *second-order cone* is defined as follows:

$$\mathcal{K}_n = \{x \in I\!\!R^n : x = (\xi; \bar{x}), \|\bar{x}\| \le \xi\},$$

where $\|.\|$ is the standard Euclidean norm, ξ is a scalar, $\bar{x} \in I\!\!R^{n-1}$, and n is the dimension of $\mathcal{K}_n$. We use $x \succeq_{\mathcal{K}_n} y$ to indicate that $x - y \in \mathcal{K}_n$. When $n = 1$, $\mathcal{K}_n = I\!\!R_+$.

Associated with the second-order cone, one can define a special case of Euclidean Jordan Algebra. Let $x = (\xi; \bar{x}) \in I\!\!R^n$ and $s = (\sigma; \bar{s}) \in I\!\!R^n$. Define $x \circ s = (x^T s; \bar{u})$, where $\bar{u} \in I\!\!R^{n-1}$, with $\bar{u}_i = \xi s_i + \sigma x_i$. The unique identity vector of this algebra is represented by $e = (1; 0)$. Clearly, $x \circ e = e \circ x = x$. Conventionally, we represent $x \circ x$ by x^2. Every $x \in \mathcal{K}_n$ has a unique square root in $\mathcal{K}_n$.

Spectral decomposition and eigenvalues of x can be defined analogously to the cone of symmetric matrices. For $x = (\xi, \bar{x})$, one has

$$\lambda_1 = \xi + \|\bar{x}\| \text{ and } \lambda_2 = \xi - \|\bar{x}\|.$$

If λ_1 and λ_2 are both nonzero, then x is invertible, with x^{-1} satisfying $x \circ x^{-1} = e$. If λ_1 and λ_2 are both nonnegative, then $x \in \mathcal{K}_n$, and if they are both positive, then $x \in \mathcal{K}_n^\circ := \{x \in I\!\!R^n : x = (\xi; \bar{x}), \|\bar{x}\| < \xi\}$

Using the eigenvalues of x, the following algebraic matrix functions can be defined:

$$\mathbf{tr}(x) := \lambda_1 + \lambda_2 = 2\xi$$

$$\det(x) := \lambda_1 \lambda_2 = \xi^2 - \|\bar{x}\|^2$$

$$\|x\|_F := \sqrt{\lambda_1^2 + \lambda_2^2} = \sqrt{2}\|x\|$$

$$\|x\|_2 := \max\{|\lambda_1|, |\lambda_2|\} = |\xi| + \|\bar{x}\|$$

Now let $x = (\xi, \bar{x}) \in I\!R^n$ with $n \geq 2$ and define

$$Q_x := \begin{pmatrix} \|x\|^2 & 2\xi\bar{x}^T \\ 2\xi\bar{x} & \det(x)I + 2\bar{x}\bar{x}^T \end{pmatrix}.$$

Q_x is a quadratic operator that maps any vector $s \in I\!R^n$ to a vector composed of quadratic terms of x. For $n = 1$ with $x = \xi$, we can define the scalar $Q_x = \xi^2$.

In this paper we are dealing with the vectors of the form $\mathbf{x} = (x_1; ...; x_k)$, where x_i is in the second-order cone $\mathcal{K}_{n_i}$. The primal algebra is therefore $\mathcal{K}^k = \mathcal{K}_{n_1} \times ... \times \mathcal{K}_{n_k}$. When there is no ambiguity, we drop the superscript k from $\mathcal{K}^k$. One can extend the above algebraic functions to these block forms.

The next lemma generalizes the inequality proved in [23] to the block format. The proof of this technical lemma is straightforward and is omitted.

Lemma 1 *Let* $\mathbf{x} = (x_1; ...; x_k) \in \mathcal{K}$, *where* $x_i \in \mathcal{K}_{n_i}$. *If* $\|\mathbf{x} - \mathbf{e}\|_2 < 1$, *then*

$$\log\det(\mathbf{x}) \geq \mathbf{tr}(\mathbf{x} - \mathbf{e}) - \frac{\|\mathbf{x} - \mathbf{e}\|_F^2}{2(1 - \|\mathbf{x} - \mathbf{e}\|_2)}. \tag{1}$$

Moreover, if $\|\mathbf{x}\|_F \leq 1$, *then*

$$\log\det(\mathbf{x} + \mathbf{e}) \geq \mathbf{tr}(\mathbf{x}) + \|\mathbf{x}\|_F + \log(1 - \|\mathbf{x}\|_F). \tag{2}$$

3 Second-order cone cutting surface method

In this section we present an analytic center cutting surface technique that uses mixed linear and multiple second-order cone cuts.

Let $A_i^T y \preceq_{\mathcal{K}_{s_i}} c_i$, for $i = 1, \ldots, n_q$ be n_q second-order cone inequalities and $A^T y \leq c$ be n_l linear inequalities. Define

$$\mathcal{D} = \{y \in I\!R^m : \mathbf{A}^T y \preceq_{\mathcal{K}} \mathbf{c}, \text{ and } A^T y \leq c\},$$

where $\mathbf{A} = (A_1, A_2, \ldots, A_{n_q})$, $\mathbf{c} = (c_1; c_2; \ldots; c_{n_q})$, $A \in I\!R^{m \times n_l}$ and $\mathcal{K} = \mathcal{K}_{s_1} \times \ldots \times \mathcal{K}_{s_{n_q}}$.

Suppose that $\mathcal{D}$ is a compact convex set that contains a full dimensional ball with ε radius. We are interested in finding a point in this ball. Let us call $\mathcal{D}$, the *dual set of localization.*

In the algorithm that we describe here, a query point is obtained by computing an approximate analytic center of the set of localization. For the moment, we assume that there exists an oracle that determines either the query point is in the ε-ball, or returns a cut that cuts off the current query point and contains the ε-ball. The cut is either a linear cut (LC) or a set of multiple second-order cone cuts (SOCC). We describe the details of this oracle in Section 5, where we discuss the implementation of the algorithm.

Let us first discuss a computational algorithm for the analytic center of $\mathcal{D}$. Let

$$\phi(\mathbf{s}, s) = \frac{1}{2} \log \det \mathbf{s} + \log \prod s,$$

where

$$\begin{aligned} \mathbf{s} &:= \mathbf{c} - \mathbf{A}^T y \succeq_{\mathcal{K}} 0 \\ s &:= c - A^T y \geq 0. \end{aligned} \tag{3}$$

It is easily verified that ϕ is a strictly concave function on $\mathcal{D}$. Therefore, the maximizer of this function over $\mathcal{D}$ exists and is unique. This maximizer is called the *analytic center* of $\mathcal{D}$. From the KKT optimality conditions y is the analytic center of $\mathcal{D}$ if and only if, there exists $\mathbf{x} = \mathbf{s}^{-1} \succeq_{\mathcal{K}} 0$ and $x = s.^{-1} \geq 0$ such that

$$\mathbf{A}\mathbf{x} + Ax = 0, \tag{4}$$

where $\mathbf{s}$ and s satisfy (3).

Corresponding to the optimality condition (4), one can derive the primal set of localization and its associated barrier function. Let

$$\mathcal{P} = \{\mathbf{x} \in \mathcal{K}^{n_q}, x \in I\!R_+^{n_l} : \mathbf{A}\mathbf{x} + Ax = 0\},$$

then

$$\psi(\mathbf{x}, x) = -\mathbf{c}^T \mathbf{x} + \frac{1}{2} \log \det \mathbf{x} - c^T x + \log \prod x$$

is strictly concave on $\mathcal{P}$. The Cartesian product of $\mathcal{P}$ and $\mathcal{D}$ gives the primal-dual set of localization. The corresponding barrier function is defined via

$$\Phi(\mathbf{x}, x, \mathbf{s}, s) = \psi(\mathbf{x}, x) + \phi(\mathbf{s}, s).$$

The unique maximizer of ψ over $\mathcal{P}$ and that of Φ over $\mathcal{P} \times \mathcal{D}$ coincide with the analytic center derived for $\mathcal{D}$. Therefore when there is no ambiguity, we refer to this point just as the analytic center.

Definition 2 *An approximate analytic center is a point that satisfies the dual feasibility (3), the primal feasibility (4) and*

$$\eta(\mathbf{x}, x, \mathbf{s}, s) \leq \eta < 1,$$

where $\eta^2(\mathbf{x}, x, \mathbf{s}, s) = \|Q_{\mathbf{x}^{1/2}}\mathbf{s} - \mathbf{e}\|_F^2 + \|xs - \mathbf{1}\|^2$.

An approximate analytic center can be computed using the primal, dual or primal-dual barrier functions. In this paper we use the primal directions to compute the analytic center. The reason is that in practice, adding the cuts returned by the oracle to the primal set of localization can be handled more efficiently than that of the dual or primal-dual sets. Notice that calculating the analytic center of one set yields the center of the other. Therefore, one can switch between the primal, dual and primal-dual sets as needed.

Let a strictly feasible point of $\mathcal{P}$ be given. Since ψ is strictly concave on $\mathcal{P}$, implementing Newton's method to maximize $\psi(\mathbf{x}, x)$ over $\mathcal{P}$ yields

$$d_{\mathbf{x}} = \mathbf{x} - Q_{\mathbf{x}}\mathbf{s} \quad (5)$$

$$d_x = x - X^2 s, \quad (6)$$

where X is a diagonal matrix made up of vector x, and $\mathbf{s}$ and s satisfy (3), with

$$y = G^{-1} g, \quad (7)$$

where

$$G = \mathbf{A} Q_{\mathbf{x}} \mathbf{A}^T + A X^2 A^T$$
$$g = \mathbf{A} Q_{\mathbf{x}} \mathbf{c} + A X^2 c.$$

Starting from a strictly feasible point, the above direction is implemented at each iteration until the desired accuracy is reached. One can prove that the rate of convergence becomes quadratic as the iteration gets closer to the analytic center.

We now present the framework of the *second-order cone cutting surface method (SOCCSM)*:

Algorithm 1 (SOCCSM) *Let $(\mathbf{x}^0, x^0)$, a strictly feasible point of $\mathcal{P}$ be given*

Step 1. *Compute $(\bar{\mathbf{x}}, \bar{x})$, an approximate analytic center of $\mathcal{P}$ using the directions $d_{\mathbf{x}}$ and d_x given in (5) and (6). Compute $\bar{y}$, an approximate center of $\mathcal{D}$ from (7).*

Step 2. *Call the oracle. If $\bar{y}$ is in the ε-ball, stop.*

Step 3. *If the oracle returns a single linear cut $b^T y \leq d$, update $\mathcal{P}$ via*

$$\mathcal{P}^+ = \{\mathbf{x} \succeq_{\mathcal{K}^{n_q}} 0, x \geq 0, \zeta \geq 0 : \mathbf{A}\mathbf{x} + Ax + b\zeta = 0\}.$$

Otherwise go to Step 4.

Step 4. *If the oracle returns multiple second-order cone cuts $\mathbf{B}^T y \preceq_{\mathcal{K}} \mathbf{d}$, update $\mathcal{P}$ via*

$$\mathcal{P}^+ = \{\mathbf{x} \succeq_{\mathcal{K}^{n_q}} 0, \mathbf{z} \succeq_{\mathcal{K}^p} 0, x \geq 0 : \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} + Ax = 0\}.$$

Step 5. *Find a strictly feasible point of $\mathcal{P}^+$ and return to Step 1.*

In the remainder of this section, we elaborate Steps 3-5 in greater detail. Step 2 will be discussed in Section 5.

After adding a cut to the set of localization, whether an LC or a set of SOCCs, the analytic center of the updated set of localization should be recovered. As mentioned before, Newton's method is employed to obtain an approximate center from a strictly feasible point. However, after adding a cut, the only available information is the previous approximate center, which may not be strictly feasible. Therefore, we need an efficient procedure to obtain an initial point for Newton's algorithm. The procedure that we describe here not only gives a strictly feasible point in $\mathcal{P}^+$, but also gives a *warm start* for the Newton directions. In Section 4.1, we show that starting from such a point requires $O(p \log(p + 1))$ Newton steps to recover the analytic center after adding p SOCCs. This procedure was initially proposed by Mitchell and Todd [21] in the linear case when a single cut is added.

For the sake of simplicity, we combine the two types of cuts and treat a linear cut as a second-order cone cut of size 1. All algebraic functions defined for the second-order cone can be simplified to be used for the linear inequalities. For example for the linear cut $s := c - a^T y \geq 0$, we define $\det(s) = s^2$. With this definition the potential function term from the SOCC and LC become identical. Therefore "$\phi(\mathbf{s}) = \frac{1}{2} \log \det \mathbf{s}$" works for both cases regardless of the size of n. Notice that with the above definition $\mathbf{tr}(s) = 2s$, and Q_s is simply the scalar s^2. All other definitions as well as Lemma 1 hold.

A strictly feasible point of $\mathcal{P}^+$ updated in Step 3 or 4, can be obtained from the following optimization problem:

$$\begin{array}{ll} \max & \frac{1}{2} \log \det \mathbf{z} \\ s.t. & \\ & \mathbf{A} d_{\mathbf{x}} + \mathbf{B}\mathbf{z} = 0 \\ & \|Q_{\mathbf{w}^{-1/2}} d_{\mathbf{x}}\|_F \leq 1, \end{array}$$

where standard choices for $\mathbf{w}$ include $\mathbf{w} = \mathbf{x}$, $\mathbf{w} = \mathbf{s}^{-1}$, and $\mathbf{w} = \mathbf{x}^{1/2}\mathbf{s}^{-1/2}$. Using the KKT optimality conditions, the updating direction reads

$$d_{\mathbf{x}} = -Q_{\mathbf{w}} \mathbf{A}^T H^{-1} \mathbf{B}\mathbf{z}$$

where

$$H = A Q_{\mathbf{w}} A^T \tag{8}$$

and

$$\mathbf{z}^{-1} = p \mathbf{B}^T H^{-1} \mathbf{B}\mathbf{z}. \tag{9}$$

The updating directions depend on $\mathbf{z}$. Let

$$\varphi(\mathbf{z}) = -\frac{p}{2} \mathbf{z}^T V \mathbf{z} + \frac{1}{2} \log \det \mathbf{z}, \tag{10}$$

where $V = \mathbf{B}^T H^{-1} \mathbf{B}$. Observe that (9) is indeed the optimality condition of

$$\max\{\varphi(\mathbf{z}) : \mathbf{z} \in \mathcal{K}^p\},$$

and since φ is a strictly concave function, Newton's method is most suitable for this problem. Therefore using the quadratic approximation of $\varphi(\mathbf{z} + d_{\mathbf{z}})$, one can derive

$$d_{\mathbf{z}} = (pV + Q_{\mathbf{z}}^{-1})^{-1}(\mathbf{z}^{-1} - pV\mathbf{z})$$

or

$$d_{\mathbf{z}} = Q_{\mathbf{z}^{1/2}}(pQ_{\mathbf{z}^{1/2}}VQ_{\mathbf{z}^{1/2}} + I)^{-1}(\mathbf{e} - pQ_{\mathbf{z}^{1/2}}V\mathbf{z})$$

Observe that the updating directions after adding a single linear cut can be simplified via

$$\begin{aligned} d_{\mathbf{x}} &= -\zeta Q_{\mathbf{w}}\mathbf{A}^T H^{-1} b \\ \zeta &= (b^T H^{-1} b)^{-1/2}. \end{aligned}$$

In the next section we discuss the convergence analysis and complexity of our algorithm.

4 Convergence analysis and complexity

In this section we present two complexity results. First we establish a bound on the number of Newton steps to recover centrality after adding multiple SOCCs, and then we discuss the convergence and complexity of the overall algorithm.

4.1 Complexity of recovering the center

Let $\mathcal{P}$ and $\mathcal{D}$ be the current primal and dual localization sets respectively, and $\bar{\mathbf{x}}$ and $\bar{y}$ be their approximate analytic centers. Let $\mathcal{P}^+$ be the updated primal set as in Step 4 of Algorithm 1. In order to derive the theoretical complexity, we need to make an assumption on the cuts in the dual space.

Assumption 1 *The updated dual set of localization is*

$$\mathcal{D}^+ = \{y \in \mathcal{D} : \mathbf{B}^T y \preceq_{\mathcal{K}^p} \mathbf{B}^T \bar{y}\}.$$

That is, the cuts pass through the center.

Note that while Assumption 1 appears to be necessary in the complexity analysis, it does not interfere with our algorithm in practice. This is because in practice we use the primal space to recover the centrality. As we observed in Section 3, in the primal space the location of the cuts does not matter; and the updating direction can always be efficiently obtained using the primal setting.

We need a dual direction. Similar to the primal case, one can obtain a dual updating direction by solving the optimization problem

$$\begin{array}{ll} \max & \frac{1}{2}\log\det(-\mathbf{B}^T d_y) \\ s.t. & \\ & \|Q_{\mathbf{w}^{1/2}}\mathbf{A}^T d_y\|_F \leq 1 \end{array}$$

where $d_y = y - \bar{y}$, and the same choices are available for $\mathbf{w}$ as before. From the KKT optimality conditions the optimal d_y reads

$$d_y = -\frac{1}{p}H^{-1}\mathbf{B}\mathbf{t}^{-1}$$

and

$$\mathbf{t} = \frac{1}{p}\mathbf{B}^T H^{-1}\mathbf{B}\mathbf{t}^{-1}, \tag{11}$$

with

$$d_{\mathbf{s}} = -\mathbf{A}^T d_y$$

Note that in view of (11)

$$\mathbf{z} \circ \mathbf{t} = \frac{1}{p}\mathbf{e}. \tag{12}$$

We now fix $\mathbf{w} = \mathbf{s}^{-1}$, so $H = \mathbf{A}Q_{\mathbf{s}^{-1}}\mathbf{A}^T = \mathbf{A}Q_s^{-1}\mathbf{A}^T$. We have the following lemma. A similar lemma for the case of single SOCC is presented in [23]. The extension to multiple SOCCs can be done following the same line of proof, so we omit the proof.

Lemma 3 *Let* $\mathbf{x}^+ = (\mathbf{x} + \alpha d_{\mathbf{x}}; \alpha\mathbf{z})$ *and* $\mathbf{s}^+ = (\mathbf{s} + \alpha d_{\mathbf{s}}; \alpha\mathbf{t})$, *for* $\alpha < 1 - \eta$. *Then*

$$\phi(\mathbf{s}^+) \geq \phi(\mathbf{s}) + \frac{1}{2}\left(\alpha + \log(1 - \frac{\alpha}{1-\eta})\right) + \frac{1}{2}\log\det\alpha\mathbf{t} \tag{13}$$

$$\psi(\mathbf{x}^+) \geq \psi(\mathbf{x}) + \frac{1}{2}\left(\alpha + \log(1 - \frac{\alpha}{1-\eta})\right) + \frac{1}{2}\log\det\alpha\mathbf{z} \tag{14}$$

$$\Phi(\mathbf{x}^+, \mathbf{s}^+) \geq \Phi(\mathbf{x}, \mathbf{s}) + (\alpha + \log(1 - \frac{\alpha}{1-\eta})) + 2p\log\alpha - p\log p. \tag{15}$$

The next theorem establishes a bound on the gap between the next analytic center and the updated point $(\mathbf{x}^+, \mathbf{s}^+)$.

Theorem 4 *Let $\mathcal{P}^+ \times \mathcal{D}^+$ be the updated primal-dual set of localization and $(\mathbf{x}^+, \mathbf{s}^+)$ be a strictly feasible point defined in Lemma 3. Let $(\mathbf{x}^a, \mathbf{s}^a)$ be the analytic center of $\mathcal{P}^+ \times \mathcal{D}^+$. Then*

$$\Phi(\mathbf{x}^a, \mathbf{s}^a) - \Phi(\mathbf{x}^+, \mathbf{s}^+) \leq p \log p - \vartheta(p, \alpha, \eta)$$

where

$$\vartheta(p, \alpha, \eta) = p + 2p \log \alpha + \alpha + \log(1 - \frac{\alpha}{1-\eta}) - \frac{\eta^2}{4(1-\eta)}$$

Proof. Since

$$\Phi(\mathbf{x}, \mathbf{s}) = -\mathbf{x}^T \mathbf{s} + \frac{1}{2} \log \det(Q_{\mathbf{x}^{1/2}} \mathbf{s})$$

and $(\mathbf{x}, \mathbf{s})$ is an approximate analytic center, in view of (1)

$$\begin{aligned} \Phi(\mathbf{x}, \mathbf{s}) &\geq -\mathbf{x}^T \mathbf{s} + \frac{1}{2} \mathbf{tr}(Q_{\mathbf{x}^{1/2}} \mathbf{s} - \mathbf{e}) - \frac{\eta^2}{4(1-\eta)} \\ &= -n^k - \frac{\eta^2}{4(1-\eta)}. \end{aligned} \tag{16}$$

Now in view of Lemma 3

$$\begin{aligned} &\Phi(\mathbf{x}^+, \mathbf{s}^+) \geq \\ &\quad -n^k - \frac{\eta^2}{4(1-\eta)} + (\alpha + \log(1 - \frac{\alpha}{1-\eta})) + 2p \log \alpha - p \log p. \end{aligned}$$

The theorem now follows from the above inequality and noting that $\Phi(\mathbf{x}^a, \mathbf{s}^a) = -n^k - p$. ■

Theorem 4 proves that after adding $p(\geq 1)$ SOCCs to the set of localization simultaneously, the gap between the primal-dual barrier function at $(\mathbf{x}^+, \mathbf{s}^+)$ and at the new analytic center is bounded by $O(p \log(p+1))$. On the other hand Newton's method is known to increase the primal-dual potential function at least by a constant amount at each iteration. Thus at most $O(p \log(p+1))$ Newton steps are needed to recover centrality after adding p SOCCs.

4.2 Convergence

In this section we derive a bound on the total number of cuts needed to obtain a point in the ε-ball. We establish upper and lower bounds on the dual barrier function after k iterations, and then show that these bounds must cross. The algorithm must terminate before the bounds cross. Let us first establish a bound on the dual barrier function at the analytic center of $\mathcal{D}^+$.

Let ψ^*, ϕ^*, Φ^* be the optimal values of primal, dual and primal-dual barrier functions respectively. That is $\psi^* = \psi(\mathbf{x}^a)$, $\phi^* = \phi(\mathbf{s}^a)$, and $\Phi^* = \Phi(\mathbf{x}^a, \mathbf{s}^a)$, and let ψ^+, ϕ^+, Φ^+ be the updated barrier functions for $\mathcal{P}^+$, $\mathcal{D}^+$ and $\mathcal{P}^+ \times \mathcal{D}^+$ respectively. From (14)

$$(\psi^+)^* \geq \psi(\mathbf{x}) + \frac{1}{2}\log\det\mathbf{z} + \vartheta_1(p,\alpha,\eta),$$

where

$$\vartheta_1(p,\alpha,\eta) = \frac{1}{2}\left(\alpha + \log(1 - \frac{\alpha}{1-\eta})\right) + p\log\alpha.$$

Since $(\psi^+)^* + (\phi^+)^* = -n - p$, one has

$$(\phi^+)^* \leq -n - p - \psi(\mathbf{x}) - \frac{1}{2}\log\det\mathbf{z} - \vartheta_1(p,\alpha,\eta).$$

On the other hand in view of (16)

$$\psi(\mathbf{x}) \geq \psi^* - \frac{\eta^2}{4(1-\eta)}.$$

Therefore

$$(\phi^+)^* \leq \phi^* - \frac{1}{2}\log\det\mathbf{z} + \frac{\eta^2}{4(1-\eta)} - p - \vartheta_1(p,\alpha,\eta). \tag{17}$$

We now obtain a lower bound on $\frac{1}{2}\log\det\mathbf{z}$. Notice that from (12) $\mathbf{z}^T(\mathbf{B}^T H^{-1}\mathbf{B})\mathbf{z} = \frac{1}{p}\mathbf{z}^T\mathbf{z}^{-1} = 1$. Consequently, since z maximizes $\varphi(z)$, for any $\mathbf{z}^0 \in \mathcal{K}^p$ such that

$$(\mathbf{z}^0)^T(\mathbf{B}^T H^{-1}\mathbf{B})\mathbf{z}^0 = 1, \tag{18}$$

one has

$$\log\det\mathbf{z} \geq \log\det\mathbf{z}^0. \tag{19}$$

Now define

$$\mathbf{z}^0 = \frac{\bar{\mathbf{z}}}{\sqrt{\bar{\mathbf{z}}^T\mathbf{B}^T H^{-1}\mathbf{B}\bar{\mathbf{z}}}} \tag{20}$$

where $\bar{\mathbf{z}} = (\bar{z}_1;\ldots;\bar{z}_p)$ is defined such that $\bar{z}_i = \gamma_i^{-1}e_i$, where

$$\gamma_i = \sqrt{(b_1^i)^T H^{-1} b_1^i}, \tag{21}$$

and b_1^i is the first column of matrix B_i. With this definition observe that

$$\begin{aligned}
\bar{\mathbf{z}}^T\mathbf{B}^T H^{-1}\mathbf{B}\bar{\mathbf{z}} &= \sum_{i=1}^p\sum_{j=1}^p \bar{z}_i^T B_i^T H^{-1} B_j \bar{z}_j \\
&= \sum_{i=1}^p\sum_{j=1}^p \gamma_i\gamma_j e_i^T B_i^T H^{-1} B_j e_j \\
&\leq p^2
\end{aligned}$$

Now on one hand, $\mathbf{z}^0$ satisfies (18), and on the other hand

$$\log\det\mathbf{z}^0 \geq \log\left(\frac{1}{p}\right)^{2p} + \log\det\bar{\mathbf{z}}$$

Therefore (19) reads

$$\log\det\mathbf{z} \geq -2p\log p + \log\det\bar{\mathbf{z}}$$

and since $\log\det\bar{\mathbf{z}} = \sum\log\det\bar{z}_i = \sum\log\gamma_i^{-2}$, then

$$\log\det\mathbf{z} \geq -2p\log p - 2\sum\log\gamma_i. \tag{22}$$

Inequalities (17) and (22) together yield the following inequality:

$$(\phi^+)^* \leq \phi^* + p\log p + \sum\log\gamma_i + \frac{\eta^2}{4(1-\eta)} - p - \vartheta_1(p,\alpha,\eta).$$

With the arbitrary values $\eta = 0.15$ and $\alpha = 0.60$, we proved the following lemma:

Lemma 5 *If the oracle returns p blocks of SOCC, where $p \geq 1$ and the dual set of localization $\mathcal{D}$ is updated by adding these cuts simultaneously, then the optimal value of the updated dual barrier function has the following upper bound:*

$$(\phi^+)^* \leq \phi^* + \sum_{i=1}^{p}\log\gamma_i + p\log p.$$

Next, we present a lemma to establish an upper bound on the optimal value of the dual barrier function at the k-th iteration. In order to keep this bound simple, we make a scaling assumption.

Assumption 2 *The initial dual set of localization $\mathcal{D}^0$ is the unit ball.*

It is important to note that Assumption 2 is simply a scaling assumption and it is made to keep constants away from the bound.

Lemma 6 *Let n_k be the total number of cuts up to the iteration k. Let γ_i, for $i = 1,\ldots,n_k$ be defined as in (21). Then*

$$(\phi^k)^* \leq \sum_{i=1}^{n_k}\log\gamma_i + n_k\log p_{max},$$

where $p_{max} := \max\{p_i, i = 1,\ldots,n_k\}$.

Proof. Let k be the current iteration. From Lemma 5

$$(\phi^k)^* \leq (\phi^{k-1})^* + \sum_{i=1}^{n_k} \log \gamma_i + p_k \log p_k,$$

where p_k is the number of SOCCs added in the k-th iteration. Since $p_{max} \geq p_k$ for all k, applying this inequality recursively, one has

$$(\phi^k)^* \leq (\phi^0)^* + \sum_{i=1}^{n_k} \log \gamma_i + n_k \log p_{max}.$$

The lemma follows from Assumption 2. ■

We now define a condition number on a second-order cone cut.

Definition 7 *Let $B^T y \preceq_{\mathcal{K}} d$ be a second-order cone cut and $u \in \mathbb{R}^m$. Define*

$$\mu = \max_{\|u\| \leq 1} \det(B^T u),$$

This condition number was first defined for semidefinite cuts in [22], and then modified for the second-order cone cuts in [23]. We make the following assumption on the multiple second-order cone cuts added at each iteration:

Assumption 3 *Let $(A_i^j)^T y \preceq c_i^j$, for $i = 1, \ldots, p_j$ be multiple SOCCs added at iteration $j = 1, \ldots, k$ and let*

$$\mu_i^j = \max_{\|u\| \leq 1} \det((A_i^j)^T u) \qquad (23)$$

be the condition numbers. Then

$$\mu^j := \min_{i=1,\ldots,p_j} \mu_i^j > 0, \quad \textit{for all} \quad j = 1, \ldots, k.$$

We now find a lower bound on the optimal value of the dual barrier function at the kth iteration.

Lemma 8

$$(\phi^k)^* \geq n_k \log(\varepsilon \sqrt{\mu_{min}}),$$

where $\mu_{min} = \min_{j=1,\ldots,k} \mu^j$ and ε is the radius of the ε-ball.

Proof. Let y^c be the center of the ε-ball. For each $i = 1, \ldots, p_j$ and each $j = 1, \ldots, k$, let u_i^j be the vector that achieves the maximum μ_i^j in (23). Then since the dual set of localization $\mathcal{D}$ contains the ε-ball, one has $c_i^j - (A_i^j)^T y \succeq \varepsilon (A_i^j)^T u_i^j$, and so

$$\begin{aligned}
\det(\mathbf{c} - \mathbf{A}^T y^c) &= \prod_{j=1}^{k} \prod_{i=1}^{p_j} \det(c_i^j - (A_i^j)^T y) \\
&\geq \prod_{j=1}^{k} \prod_{i=1}^{p_j} \det(\varepsilon (A_i^j)^T u_i^j) \\
&\geq \varepsilon^{2n_k} \prod_{j=1}^{k} (\mu^j)^{p_j} \\
&\geq \varepsilon^{2n_k} \mu_{min}^{n_k}.
\end{aligned}$$

The proof follows. ■

Combining Lemmas 6 and 8 gives the following inequality:

$$\frac{1}{n_k} \sum_{i=1}^{n_k} \log \gamma_i^2 \geq \log \left(\frac{\varepsilon \sqrt{\mu_{min}}}{p_{max}} \right)^2. \tag{24}$$

On the other hand, since $\prod \gamma_i^2 \leq \left(\frac{\sum \gamma_i^2}{n_k} \right)^{n_k}$ then

$$\frac{1}{n_k} \sum_{i=1}^{n_k} \log \gamma_i^2 \leq \log \frac{\sum_{i=1}^{n_k} \gamma_i^2}{n_k}. \tag{25}$$

Inequalities (24) and (25) yield

$$n_k \left(\frac{\varepsilon \sqrt{\mu_{min}}}{p_{max}} \right)^2 \leq \sum_{i=1}^{n_k} \gamma_i^2. \tag{26}$$

It remains to bound the right hand side of (26). Let us first make another scaling assumption.

Assumption 4 *Let $A_i^T y \preceq_{\mathcal{K}} A_i^T \bar{y}$, for $i = 1, \ldots, n^q$, and $a_i^T y \leq a_i^T \bar{y}$, for $i = 1, \ldots, n^{lc}$, be second-order cone cuts and linear cuts added to the set of localization. One can assume that*

$$\max_i \{ \|A_i\|_F, \|a_i\| \} \leq 1.$$

Assumption 4 is another scaling assumption and does not reduce generality. Notice that if $\|A_i\|_F \leq 1$, then the Euclidean norm of all columns of A_i is less than or equal 1.

Lemma 9 *Let*

$$\mathcal{H}_k = I + \frac{1}{16}\sum_{i=1}^{n_k}(b_1^i)(b_1^i)^T,$$

where b_1^i's are the first columns of matrices B_i of the second-order cone cuts, Then

$$(b_1^i)^T \mathcal{H}_k^{-1} b_1^i \geq \gamma_i^2.$$

Proof. See Lemma 15 of [23]. ■

The next lemma establishes an upper bound on the right hand side of Inequality (26). The structure of this proof has some similarities to that of Lemma 9 in [36] for the case of linear programming. Assumption 2 and our Lemma 9 enable us to prove a stronger result with an upper bound that is only linear in m. We include the proof in detail in order to make the strengthening clear.

Lemma 10

$$\sum_{i=1}^{n_k}\gamma_i^2 \leq 2m(p_{max}+16)\log(1+\frac{n_k}{4m})$$

Proof. Let

$$\mathcal{H}_k = \mathcal{H}_{k-1} + \frac{1}{16}\sum_{i=1}^{p_k} b_1^i(b_1^i)^T,$$

with $\mathcal{H}_0 = I$. Let $p_k \geq 2$ (the case of $p_k = 1$ yields a tighter bound, see [23]). One has

$$\det(\mathcal{H}_k) = (1+\frac{\bar{\gamma}^2}{16})\det\left(\mathcal{H}_{k-1} + \frac{1}{16}\sum_{i=2}^{p_k} b_1^i(b_1^i)^T\right) \tag{27}$$

where $\bar{\gamma}^2 = (b_1^1)^T\left(\mathcal{H}_{k-1} + \frac{1}{16}\sum_{i=2}^{p_k} b_1^i(b_1^i)^T\right)^{-1} b_1^1$. Now let

$$\mathcal{J} = I + \frac{1}{16}\sum_{i=2}^{p_k}\mathcal{H}_{k-1}^{-1/2} b_1^i(b_1^i)^T\mathcal{H}_{k-1}^{-1/2}.$$

We prove that

$$\mathcal{J}^{-1} \succeq \frac{16}{p_{max}+16} I. \tag{28}$$

It suffices to show that $x^T\mathcal{J}x \leq \frac{p_{max}+16}{16}$, for all $x \in I\!R^m$ with $\|x\| = 1$. This can be seen from the following chain of inequalities and Assumption 4.

$$x^T\mathcal{J}x \quad = \quad \|x\| + \frac{1}{16}\sum_{i=2}^{p_k}(x^T\mathcal{H}_{k-1}^{-1/2}b_1^i)^2$$

$$\begin{aligned} &\leq 1 + \frac{1}{16}\sum_{i=2}^{p_k}(b_1^i)^T\mathcal{H}_{k-1}^{-1}b_1^i \\ &\leq 1 + \frac{1}{16}\sum_{i=2}^{p_k}\|b_1^i\|^2 \end{aligned}$$

Therefore

$$\begin{aligned} \bar{\gamma}^2 &= (b_1^1)^T\mathcal{H}_{k-1}^{-1/2}\mathcal{J}^{-1}\mathcal{H}_{k-1}^{-1/2}b_1^1 \\ &\geq \frac{16}{p_{max}+16}(b_1^1)^T\mathcal{H}_{k-1}^{-1}b_1^1 \\ &\geq \frac{16\gamma_1^2}{p_{max}+16} \end{aligned}$$

Therefore (27) reads

$$\det(\mathcal{H}_k) \geq (1+\frac{\gamma_1^2}{p_{max}+16})\det\left(\mathcal{H}_{k-1}+\frac{1}{16}\sum_{i=2}^{p_k}b_1^i(b_1^i)^T\right)$$

Repeating this inequality for $i = 2, \dots, p_k$, and taking "log" from both sides one has

$$\log\det\mathcal{H}_k \geq \sum_{i=1}^{p_k}\log\left(1+\frac{\gamma_i^2}{p_{max}+16}\right) + \log\det\mathcal{H}_{k-1}.$$

On the other hand since $\gamma_i \leq 1$ and $p_{max} \geq 2$,one has

$$\log\left(1+\frac{\gamma_i^2}{p_{max}+16}\right) \geq \frac{\gamma_i^2}{2(p_{max}+16)}.$$

Consequently

$$\log\det\mathcal{H}_k \geq \frac{1}{2}\sum_{i=1}^{p_k}\frac{\gamma_i^2}{p_{max}+16} + \log\det\mathcal{H}_{k-1}.$$

Notice that, a tighter inequality can be derived when $p_k = 1$.

By repeating the same procedure for all second-order cone cuts, one has

$$\log\det\mathcal{H}_k \geq \frac{1}{2(p_{max}+16)}\sum_{i=1}^{n_k}\gamma_i^2 + \log\det\mathcal{H}_0$$

Now since $\log\det\mathcal{H}_k \leq m\log(\frac{\mathbf{tr}\mathcal{H}_k}{m})$ and

$$\mathbf{tr}\mathcal{H}_k \leq m + \frac{n_k}{16},$$

therefore

$$\frac{1}{2(p_{max}+16)}\sum_{i=1}^{n_k}\gamma_i^2 \leq m\log\left(1+\frac{n_k}{16m}\right).$$

The lemma follows immediately. ■

Combining Lemma 10 and inequality (26), yields our main result.

Theorem 11 *The second-order cone cutting surface method (Algorithm 1) finds a point in the ε-ball when the total number of linear and second-order cone cuts reaches the bound*

$$O\left(\frac{mp_{max}^3}{\varepsilon^2 \mu_{min}}\right)$$

Theorem 11 shows that Algorithm 1 is polynomial with respect to m and p.

5 Application: large scale conic optimization

In this section we discuss an application of SOCCSM on large scale conic optimization with bounded feasible region. Consider the following pair of primal and dual linear-quadratic-semidefinite programming problems:

$$\begin{array}{ll} \min & C \bullet X + \mathbf{c}^T\mathbf{x} + c^T x \\ s.t. & \\ & \mathcal{A}X + \mathbf{A}\mathbf{x} + Ax = b \\ & I \bullet X \qquad\qquad = 1 \\ & X \succeq 0, \mathbf{x} \succeq_{\mathcal{K}} 0, x \geq 0 \end{array} \tag{29}$$

and

$$\begin{array}{llll} \max & b^T y + z & & \\ s.t. & & & \\ & \mathcal{A}^T y + zI & \preceq & C \\ & \mathbf{A}^T y & \preceq_{\mathcal{K}} & \mathbf{c} \\ & A^T y & \leq & c \end{array} \tag{30}$$

where $C \in \mathcal{M}^{n_s}$, $\mathbf{c} \in I\!R^{n_q}$, $c \in I\!R^{n_l}$, $b \in I\!R^m$, $\mathbf{A} \in I\!R^{m \times n_q}$, $A \in I\!R^{m \times n_l}$, and the linear operator $\mathcal{A} : \mathcal{M}^{n_s} \to I\!R^m$, defined by $(\mathcal{A}X)_i = A_i \bullet X$ and its adjoint operator $\mathcal{A}^T : I\!R^m \to \mathcal{M}^{n_s}$, defined by $\mathcal{A}^T y = \sum_{i=1}^m y_i A_i$ are all the given problem parameters; and $X \in \mathcal{M}^{n_s}, \mathbf{x} \in I\!R^{n_q}$, and $x \in I\!R^{n_l}$ are the primal problem variables, and $y \in I\!R^m$ and $z \in I\!R$ are the variables of the dual problem. We assume that the matrices A_i, $i = 1, \ldots, m$ are linearly independent. Notice that the constraint $I \bullet X = 1$ is implied from our assumption that the feasible region is bounded (See Helmberg [12]).

In this section we implement our algorithm on some randomly generated problems with dense matrices and on some maxcut and Lovasz theta problems. Moreover, we compare our results with that of some well-known software, namely SDPLR[1], due to Burer and Monteiro [6, 7] and Burer and Choi [5]; SDPT3-4.0-beta[2],due to Toh et al. [32] and Tutuncu et al. [34]; and SeDuMi[3], due to Sturm [29, 30].

First observe that the equivalent formulation of Problem (30) as an eigenvalue optimization problem reads

$$\begin{array}{ll} \max & b^T y + \lambda_{min}(C - \mathcal{A}^T y) \\ s.t. & \\ & \mathbf{A}^T y \preceq_{\mathcal{K}} \mathbf{c} \\ & A^T y \leq c. \end{array} \tag{31}$$

Let us first assume that the feasible region of Problem (31) is bounded. Later on in this section, we describe a procedure that extends the algorithm to unbounded situations. Given our assumption on the feasible region, without loss of generality, one can assume that the second-order cone inequality $\mathbf{A}^T y \preceq_{\mathcal{K}} \mathbf{c}$ contains $(A^0)^T y \preceq_{\mathcal{K}} e$, where $A^0 = \begin{pmatrix} \mathbf{0} & -I_m \end{pmatrix}$. That is the feasible region of Problem (31) falls in the unit ball $\|y\| \leq 1$.

Let us first briefly study the objective function of Problem (31) and review some important properties of the minimum eigenvalue function. More comprehensive analysis of the minimum eigenvalue function and eigenvalue optimization can be found in Overton [25], Lewis and Overton [17] and Helmberg and Rendl [13]. Let

$$f(y) := b^T y + \lambda_{min}(C - \mathcal{A}^T y).$$

It is well-known that the minimum eigenvalue of a symmetric matrix can be cast as a semidefinite programming problem. Therefore $f(y)$ can be written as:

$$f(y) = b^T y + \min \left\{ (C - \mathcal{A}^T y) \bullet V : \mathbf{tr} V = 1, V \in \mathcal{M}_+^{n_s} \right\},$$

Although $C - \mathcal{A}^T y$ is a differentiable matrix function with respect to y, the eigenvalues of this matrix are not differentiable at points where they have multiplicity greater than one. In

[1]available at http://dollar.biz.uiowa.edu/ burer/software/SDPLR/papers.html

[2]available at http://www.math.nus.edu.sg/ mattohkc/sdpt3.html

[3]available at Advanced Optimization Library http://sedumi.mcmaster.ca

maximizing the minimum eigenvalue of an affine combination of symmetric matrices, it is often the case that the maximum occurs where $f(y)$ is nondifferentiable. In such cases, one can work with the subdifferential set rather than the gradient. The subdifferential of function $f(y)$ using the Clarke generalized gradient of $\lambda_{min}(C - \mathcal{A}^T y)$ and a chain rule can be derived as

$$\partial f(y) := b - \{v \in I\!\!R^m : v_i = (Q^T A_i Q) \bullet V, trV = 1, V \in \mathcal{M}_+^{\hat{p}}\},$$

where $Q \in I\!\!R^{n \times \hat{p}}$ is a matrix whose orthonormal columns are the eigenvectors corresponding to the minimum eigenvalue with multiplicity $\hat{p}$. Observe that if the maximum eigenvalue is unique ($\hat{p} = 1$), then the subdifferential set will reduce to a unique vector, which is the gradient of $f(y)$. In other words, function $f(y)$ is differentiable, if the multiplicity of λ_{min} is one.

Now let $\mathcal{F}^0$ be the feasible region of Problem (31). That is

$$\mathcal{F}^0 = \{y \in I\!\!R^m : \mathbf{A}^T y \preceq_{\mathcal{K}} \mathbf{c} \text{ and } A^T y \le c\}.$$

The following lemma provides upper and lower bounds for the optimal objective value of Problem (31):

Lemma 12 *Let f^* be the optimal objective value of the eigenvalue optimization problem (31). Let $y^0 \in \mathcal{F}^0$ be feasible. Then $|f^*| \le \delta$, where*

$$\delta = max\left\{|f(y^0)|, \frac{1}{n_s}\left(|\mathbf{tr}C| + \|\mathcal{A}I - n_s b\|\right)\right\}.$$

Proof. First observe the lower bound $f^* \ge f(y^0)$ for any feasible $y^0 \in \mathcal{F}^0$. On the other hand the objective function of the primal problem (29) provides an upper bound on f^*. We use a restricted version of (29). First consider a relaxation of (31) where all constraints are removed except for the ball constraint. That is

$$\begin{array}{lll} \max & b^T y + z & \\ s.t. & & \\ & \mathcal{A}^T y + zI & \preceq \quad C \\ & (A^0)^T y & \preceq_{\mathcal{K}} \quad e. \end{array}$$

The primal problem (29) is, therefore, restricted to

$$\begin{array}{ll} \min & C \bullet X + e^T x_q \\ s.t. & \\ & \mathcal{A}X + A^0 x_q = b \\ & I \bullet X \qquad\quad = 1 \\ & X \succeq 0, x_q \succeq_{\mathcal{K}_{m+1}} 0, \end{array} \tag{32}$$

where $x_q = (\xi_q; \bar{x}_q)$, with $\bar{x}_q \in I\!\!R^m$.

Clearly the objective function of problem (32) at a feasible point provides an upper bound for Problem (31). To obtain a feasible point of Problem (32), let $X^0 = \frac{1}{n_s} I \succ 0$. From the definition of A^0, $\bar{x}_q^0 = \frac{1}{n_s}\mathcal{A}I - b$ satisfies the linear constraint $\mathcal{A}X^0 + A^0 x_q^0 = b$, and any $\xi_q^0 \geq \|\bar{x}_q^0\|$ satisfies $x_q^0 \succeq_{\mathcal{K}} 0$. Therefore

$$f(y^0) \leq f^* \leq \frac{\mathbf{tr}C}{n_s} + \|\frac{1}{n_s}\mathcal{A}I - b\|.$$

The lemma now follows from the above inequality. ■

Now let

$$\mathcal{D}^0 = \{(y; z) \in I\!\!R^{m+1} : y \in \mathcal{F}^0, |b^T y + z| \leq \delta\},$$

be our initial localization set. That is $\mathcal{D}^0$ contains the feasible region of Problem (31) and the optimal objective value f^*, and of course, is bounded and convex.

Let $(y^0; z^0) \in \mathcal{D}^0$ be an initial query point. If f is differentiable at y^0, then $\hat{p} = 1$ and the matrix Q reduces to a column vector q, and the set $\mathcal{D}^0$ can be replaced by

$$\mathcal{D} := \{(y; z) \in \mathcal{D}^0 : \hat{b}^T y + z \leq \hat{d}, \;\; b^T y + z \geq \max(-\delta, f(y^0))\},$$

where $\hat{b} \in I\!\!R^m$, with $\hat{b}_i = q^T A_i q$, for $i = 1, ..., m$, and $d = q^T C q$. Otherwise, if f is not differentiable at y^0, then $\hat{p} > 1$, and the set $\mathcal{D}^0$ can be replaced by

$$\mathcal{D} := \{(y; z) \in \mathcal{D}^0 : \hat{\mathcal{B}}^T y + zI \preceq \hat{D}, \;\; b^T y + z \geq \max(-\delta, f(y^0))\},$$

where $\hat{\mathcal{B}}^T y + zI \preceq \hat{D}$ is a $\hat{p}$-dimensional semidefinite inequality that contains the optimal solution of Problem (31) and $\hat{\mathcal{B}}^T y = \sum_{i=1}^m y_i \hat{B}_i$, $\hat{B}_i = Q^T A_i Q$, for $i = 1, ..., m$.

We relax this $\hat{p}$-dimensional semidefinite inequality by second-order cone inequalities. First observe that if $A \in \mathcal{M}^2_+$, then positive semidefiniteness of A can be represented as a second-order cone inequality [14]. That is

$$\begin{pmatrix} \alpha & \gamma \\ \gamma & \beta \end{pmatrix} \succeq 0, \quad \text{if and only if} \quad \alpha + \beta \geq \left\| \begin{pmatrix} \alpha - \beta \\ 2\gamma \end{pmatrix} \right\|,$$

and the norm inequality is equivalent to a second-order cone inequality:

$$\begin{pmatrix} \alpha + \beta \\ \alpha - \beta \\ 2\gamma \end{pmatrix} \in \mathcal{S}_3. \tag{33}$$

On the other hand, we know that every 2×2 principle submatrix of a positive semidefinite matrix must be positive semidefinite. Therefore $A \in \mathcal{M}^n_+$ can be relaxed into $\frac{n(n-1)}{2}$ second-order cone inequalities.

Now consider the semidefinite inequality $\sum_{k=1}^m y_k \hat{B}^k + zI \preceq \hat{D}$, where $\hat{B}_k, \hat{D} \in \mathcal{M}^{\hat{p}}$. Consider the 2×2 principle submatrix in locations i and j, for $i < j$. One has

$$\begin{pmatrix} \hat{D}_{ii} & \hat{D}_{ij} \\ \hat{D}_{ij} & \hat{D}_{jj} \end{pmatrix} - \sum_{k=1}^m y_k \begin{pmatrix} \hat{B}^k_{ii} & \hat{B}^k_{ij} \\ \hat{B}^k_{ij} & \hat{B}^k_{jj} \end{pmatrix} - z \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \succeq 0$$

or

$$\begin{pmatrix} \hat{D}_{ii} - \sum y_k \hat{B}^k_{ii} - z & \hat{D}_{ij} - \sum y_k \hat{B}^k_{ij} \\ \hat{D}_{ij} - \sum y_k \hat{B}^k_{ij} & \hat{D}_{jj} - \sum y_k \hat{B}^k_{jj} - z \end{pmatrix} \succeq 0.$$

In view of (33), the above inequality is equivalent to

$$\begin{pmatrix} \hat{D}_{ii} + \hat{D}_{jj} - \sum y_k (\hat{B}^k_{ii} + \hat{B}^k_{jj}) - 2z \\ \hat{D}_{ii} - \hat{D}_{jj} - \sum y_k (\hat{B}^k_{ii} - \hat{B}^k_{jj}) \\ 2\hat{D}_{ij} - 2 \sum y_k \hat{B}^k_{ij} \end{pmatrix} \in \mathcal{S}_3. \tag{34}$$

Now let $d^{ij} = (\hat{D}_{ii} + \hat{D}_{jj}; \hat{D}_{ii} - \hat{D}_{jj}; 2\hat{D}_{ij})$ and $B^{ij} \in \mathbb{R}^{m \times 3}$ be a matrix whose kth row is defined via $(\hat{B}^k_{ii} + \hat{B}^k_{jj}, \hat{B}^k_{ii} - \hat{B}^k_{jj}, 2\hat{B}^k_{ij})$. Then (34) reads

$$d^{ij} - (B^{ij})^T y - 2ez \in \mathcal{S}_3,$$

for all $i < j$. Therefore the semidefinite inequality $\hat{\mathcal{B}}^T y + zI \preceq \hat{D}$ can be relaxed into $\frac{\hat{p}(\hat{p}-1)}{2}$ second-order cone inequalities, and the set of localization be enlarged via

$$\mathcal{D} = \{(y, z) \in \mathcal{D}^0 : \mathbf{B}^T y + 2z\mathbf{e} \preceq_{\mathcal{K}} \mathbf{d}, \ \textit{and} \ b^T y + z \geq \min(-\delta, f(y^0))\}$$

where $\mathbf{B} = (B_1 \;\; B_2 \;\; ... \;\; B_p)$, $\mathbf{d} = (d_1; d_2; ...; d_p)$, and $\mathcal{K} = \mathcal{K}_3 \times \mathcal{K}_3 \times ... \times \mathcal{K}_3$ composed of p blocks, where $p = \frac{\hat{p}(\hat{p}-1)}{2}$ and B_k's and d_k's are as defined above.

Notice that one can generate far more SOCCs from a single SDC than just those coming from pairs of eigenvectors from a particular eigenbasis. This can be done by multiplying an appropriate $\hat{p} \times 2$ matrix U by the semidefinite cut to give $U^T(D - \mathcal{B}^T y - zI)U$ and requiring this 2×2 matrix to be positive semidefinite. Different U's would give different combinations of eigenvectors. Unfortunately, it is not clear which are the useful U's. The procedure described above can be regarded as putting one 1 in each column of U.

We implement Algorithm 1 with the set of localization $\mathcal{D}$. Depending on which one of the above cases occurs, a single linear cut or a set of multiple second-order cone cuts will be added to $\mathcal{D}$ and the lower bound is updated. Therefore at the kth iteration the set of localization has the following structure:

$$\mathcal{D}^k = \{(y; z) \in I\!R^{m+1} : (\hat{\mathbf{A}}^k)^T y + 2z\mathbf{e} \preceq_{\mathcal{K}} \hat{\mathbf{c}}^k, (\hat{A}^k)^T y + \mathbf{1}z \leq \hat{c}^k, b^T y + z \geq \theta^k\},$$

where $\hat{\mathbf{A}}^k$ contains $n_q^k = 2 + \sum_{i=1}^k p_i$ blocks of SOCCs, $\hat{A}^k \in I\!R^{m \times n_l^k}$ contains n_l^k linear cuts, and $\theta^k = \max(\theta^{k-1}, f(y^{k-1}))$, is the best lower bound.

The set $\mathcal{D}^k$ is a compact convex set that is described by linear and second-order cone inequalities. The lower bound cut $b^T y + z \geq \theta^k$ is a linear cut and could be incorporated into the linear inequalities. However, since in our algorithm we implement a weighted analytic center with the weight on the lower bound cut, we prefer to study this cut separately. In the presence of many subgradient cuts, the use of the weighted analytic center helps to centralize the analytic center.

6 Computational experience

In this section we illustrate some preliminary computational results of Algorithm 1 when implemented on pair of (29) and (30). The data for the test problems are randomly generated from normal distributions with different means and standard deviations. We first derive a stopping criterion to terminate the algorithm.

6.1 Stopping criterion

To terminate the algorithm we compute the gap between the lower and upper bounds on the optimal objective value at each iteration. Recall that the lower bound θ^k is regularly updated at each iteration after adding cuts. We now describe a procedure to update the upper bound.

Given the definition of $\mathcal{D}^k$, at the kth iteration of the algorithm the following relaxation of the dual problem (30) is formed:

$$\begin{array}{lrcl} \max & b^T y + z & & \\ s.t. & & & \\ & (\hat{\mathbf{A}}^k)^T y + 2\mathbf{e}z & \preceq_{\mathcal{K}} & \hat{\mathbf{c}}^k \\ & (\hat{A}^k)^T y + \mathbf{1}z & \leq & \hat{c}^k \\ & -b^T y - z & \leq & -\theta^k. \end{array}$$

An upper bound for this problem is obtained by evaluating the objective function of the restricted primal problem at a feasible point. The restricted primal problem corresponding to the relaxed dual reads

$$\begin{array}{ll} \min & (\hat{\mathbf{c}}^k)^T \mathbf{x} + (\hat{c}^k)^T x - \theta^k \xi \\ s.t. & \\ & \hat{\mathbf{A}}^k \mathbf{x} + \hat{A} x - b\xi = b \\ & 2\mathbf{e}\mathbf{x} + \mathbf{1}x - \xi = 1 \\ & \mathbf{x} \succeq_{\mathcal{K}} 0, x \geq 0, \xi \geq 0. \end{array} \tag{35}$$

On the other hand from Section 3, Equation (4), one can imply that $\mathbf{x}^k \succ_{\mathcal{K}} 0, x^k > 0, \xi^k > 0$ at the kth iteration of SOCCSM satisfies:

$$\begin{array}{rcl} \hat{\mathbf{A}}^k \mathbf{x}^k + \hat{A}^k x^k - b\xi^k & = & 0 \\ 2\mathbf{e}\mathbf{x}^k + \mathbf{1}x^k - \xi^k & = & 0. \end{array}$$

Therefore $(\mathbf{x}^k/\xi^k, x^k/\xi^k, 0)$ is feasible for the restricted primal problem (35), and therefore

$$\delta^k := \frac{1}{\xi^k}\left((\hat{\mathbf{c}}^k)^T \mathbf{x}^k + (\hat{c}^k)^T x^k\right)$$

is an upper bound on the optimal objective value f^*.

In our computational results we measure the gap between the upper and lower bounds

$$\frac{\delta^k - \theta^k}{1 + \delta^k}$$

at each iteration. The algorithm stops when this gap falls below an ε.

6.2 Random problems

We now present some numerical results[4] of implementing SOCCSM on problems (29) and (30) with fully dense matrices. We compare our results with that of SDPLR, SDPT3, and SeDuMi.

In this section we initialize the problem with the quadratic constraints $\|y\| \leq 1$ and $|z| \leq \delta$ to ensure the compactness of the feasible region, and $c \geq 0$ in linear constraints $A^T y \leq c$ to guarantee a nonempty feasible region. To illustrate the performance of our algorithm, we use the Matlab function "RANDN" to generate random data with full density from a normal distribution. In Section 6.3 we implement the algorithm on sparse problems with unbounded feasible region.

Tables 1 and 2 illustrate computational results and cpu times for problems with different n_s, m, and n_l, the dimension of the original semidefinite block, the number of constraints in (29), and the size of the original linear block respectively, that are presented in the first column. The columns under lc and socc show the number of linear cuts and second-order cone blocks added before the stopping criterion is satisfied. A second-order cone block contains multiple SOCCs obtained from relaxing a semidefinite cut. The maximum number of cuts in an SOCC block is indicated by p. The dimension of cut is indicated by "*dim*", which represents the total number of SOCC's and LC's added. The column under "gap" shows the relative error between the upper and lower bounds. Finally, the last four columns compare the cpu time of SOCCSM, SDPLR, SDPT3, and SeDuMi in seconds.

Notice that we use different duality gaps depending on the size of the problem. The cpu time of all software have been calculated according to the corresponding gap. While this serves our purpose in comparing cpu times, it saves us time in running the test problems. We also impose a time limit of 10,000 seconds on all codes. Therefore a cpu time of 9999 indicates that the

[4]The results are obtained on a laptop computer with Intel(R) Pentium(R) M processor 2.00GHz and 2.00 GB of RAM.

software exceeds this time limit.

Table 1: Computational results on random problems

(n_s, m, n_l)	lc	socc	p	*dim*	gap	SOCCSM	SDPLR	SDPT3	SeDuMi
300,50,100	14	46	3	109	9.2e-4	16	29	69	352
300,50,300	5	24	2	53	8.6e-4	8	35	105	450
300,100,300	9	46	4	178	9.2e-4	59	71	110	620
300,100,600	4	31	4	121	7.8e-4	56	78	225	843
300,200,800	6	49	7	295	8.5e-4	246	127	438	1261
300,200,1100	4	44	4	169	8.6e-4	191	146	503	1834
300,300,800	11	79	8	518	9.2e-4	739	–	656	–
300,300,1000	5	62	7	335	9.2e-4	561	–	747	–
500,50,200	4	33	4	128	7.6e-4	77	144	247	1245
500,50,600	1	21	2	43	9.1e-4	21	155	468	1623
500,100,500	3	42	4	158	8.4e-4	169	265	1031	2084
500,100,1000	9	21	3	69	7.3e-4	67	358	1227	2504
500,200,500	11	81	5	384	9.1e-4	643	–	1555	–
500,200,2000	9	30	3	94	8.6e-4	398	–	1791	–
500,300,500	18	78	8	434	4.9e-3	895	–	2013	–
500,300,1000	1	44	7	221	4.7e-3	646	–	2558	–
800,10,400	0	9	2	18	4.8e-4	11	128	409	2281
800,10,800	8	1	2	10	4.2e-4	7	159	553	2798
800,50,500	1	22	3	58	9.1e-4	77	433	1453	3449
800,50,1000	18	1	2	20	7.3e-4	42	693	1624	3922
800,100,800	2	32	4	114	9.4e-4	313	–	2811	–
800,100,1200	2	27	3	73	9.5e-4	199	–	3587	–
800,200,1000	1	28	4	107	3.4e-3	467	–	5814	–
800,200,1500	1	24	5	98	4.5e-3	424	–	6332	–

The numerical results presented in these two tables show that SOCCSM can efficiently handle large and dense problems. Comparing the cpu times show that it performs significantly better than SDPLR, SDPT3, and SeDuMi when applied to dense problems with large n_s and small to moderate m. In almost all of the test problems presented here, SOCCSM reaches the desired accuracy much faster than the other three software packages[5].

[5]It should be mentioned that SDPLR has been designed for Low-Rank problems and SDPT3 has been designed for large and sparse problems and they both performs extremely well when applied to appropriate problems.

Unfortunately we could not test all of our problems with SDPLR and SeDuMi due to the input structure of these two software packages. Both SDPLR and SeDuMi store problem data in one (sparse) coefficient matrix. Obviously when the problem is dense this matrix exceeds the memory capacity. SDPT3's input structure is slightly better as it uses cells to store data. We indicate these failures by dashes.

There are situations where SDPLR and SDPT3 outperform SOCCSM even on dense problems (e.g., $n_s = 300$ and $m = 200$ for SDPLR and $n_s = 300$, $m = 300$, and $n_l = 800$ for SDPT3). This, more or less, happens whenever m is relatively large. The reason is that m directly defines the dimension of the set of localization. Obviously the larger the set of localization, the more cuts are required to satisfy the desired accuracy, and therefore the slower the convergence. This is a weakness of ACCPM and our algorithm is no exception.

Table 2: Computational results on random problems

(n_s, m, n_l)	lc	socc	p	*dim*	gap	SOCCSM	SDPLR	SDPT3	SeDuMi
1000,10,500	2	7	2	16	7.6e-4	14	227	986	3995
1000,10,2000	8	1	2	10	5.5e-4	18	241	1798	5012
1000,50,400	11	17	2	45	8.4e-4	110	–	2097	–
1000,50,900	1	19	2	39	8.8e-4	69	–	2553	–
1000,100,500	3	46	3	137	3.9e-3	223	–	–	–
1000,100,1000	0	20	3	49	4.3e-3	134	–	–	–
2000,10,100	1	12	2	25	7.3e-4	112	–	4929	–
2000,10,500	1	8	2	17	7.1e-4	65	–	5996	–
2000,10,1000	0	7	2	14	4.8e-4	59	–	6617	–
2000,20,100	1	13	3	34	2.6e-3	177	–	8136	–
2000,20,500	0	8	2	16	3.2e-3	83	–	8664	–
2000,20,800	1	7	2	15	3.7e-3	70	–	9050	–
2500,10,100	2	6	2	14	4.5e-3	104	–	9999	–
2500,10,700	4	1	2	6	3.9e-3	32	–	9999	–
2500,20,50	4	18	3	41	4.3e-3	368	–	9999	–
2500,20,500	6	2	2	10	4.8e-3	87	–	9999	–
3000,10,100	1	9	2	19	4.3e-3	177	–	–	–
3000,10,500	0	6	2	12	3.2e-3	106	–	–	–

An interesting observation is that, while the cpu time of SDPLR, SDPT3, and SeDuMi consistently increases as the problem dimension and the number of constraints rise, SOCCSM shows a

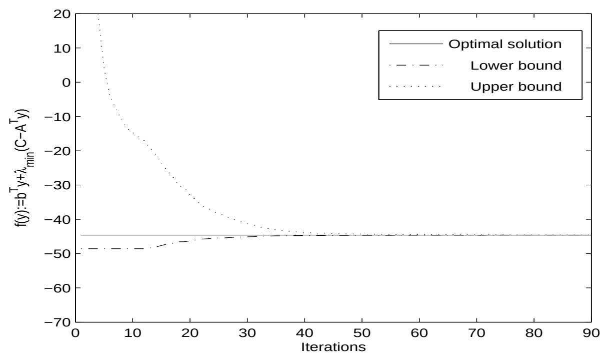

Figure 1: Convergence of a random problem with $n_s = 300$, $m = 300$, and $n_l = 800$

better performance when dealing with problems with more constraints. This is indeed an inherit advantage of SOCCSM. For instance, a random problem with $n_s = 800$ and $m = 10$ is solved in 11 seconds when $n_l = 400$, and in 7 seconds when we add additional 400 linear constraints. The cpu time of SDPLR, SDPT3, and SeDuMi for the above two classes of problems increases from 128 to 159, 409 to 553, and 2281 to 2798 seconds respectively.

This observation was actually expected before running the test problems. The reason again goes back to the structure of the set of localization. A problem with more constraints usually gives rise to a smaller set of localization. In fact, the more constraints, the smaller the original set of localization, and therefore SOCCSM does not need to add too many cuts or surfaces before the desired accuracy is reached. For instance *dim* for a problem with $n_s = 2000$, $m = 10$, and n_l =100, 500, and 1000 is 112 (12 blocks of SOCCs and 1 LC), 65 (8 blocks of SOCCs and 1 LC), and 59 (7 blocks of SOCCs and 0 LC) respectively.

The computational results reported in this paper also outperform those of the matrix generation approach reported in Oskoorouchi and Goffin [24]. This shows that working with weaker but less expensive cuts (second-order cone) has computational advantages over strong but expensive cuts (semidefinite).

Figure 1 illustrates the convergence behavior of SOCCSM. It shows how the upper and lower bounds approach the optimal solution for a problem with $n_s = 300$, $m = 300$ and $n_l = 800$. As it appears in this figure the algorithm requires approximately 90 iterations before the gap between the upper and lower bounds falls within the desired accuracy. Table 1 shows that the

90 iterations are composed of 79 blocks of SOCCs and 11 linear cuts. A careful look at Figure 1 reveals an interesting observation. An approximate solution is reached around iteration 40 when the lower bound approaches the optimal point. This is a very typical behavior of ACCPM. The reason is that the upper bound is updated on a restricted version of the original problem (see Problem (35) in Section 6.1) and therefore it is not the tightest possible bound. While there is no other way to achieve a better bound, there might be a more appropriate way to define a stopping criterion. For example one can use the gap between the lower bound and the z value of the weighted analytic center (the $(m+1)$th component). Since the lower bound cut is associated with a weight equal to the total number of cuts, the analytic center does not approach this cut unless the localization set is reasonably small.

6.3 Maxcut and Lovasz Theta problems

Although SOCCSM is specifically designed for mixed linear-quadratic-semidefinite programming problems with bounded feasible region, however it can be modified to handle unbounded problems as well. The latter class of problems arises in many combinatorial applications such as the maxcut and Lovasz theta problems. We refer the reader to Helmberg [11] and references therein for details on semidefinite relaxations in combinatorial optimization.

First consider the semidefinite relaxation of the maxcut problem:

$$\begin{array}{ll} \min & -\frac{1}{4}L \bullet X \\ s.t. & \\ & diag(X) = \mathbf{1} \\ & X \succeq 0 \end{array} \tag{36}$$

where L is the Laplacian matrix of the graph. Notice that $diag(X) = \mathbf{1}$ implies that $I \bullet X = n_s$. Therefore this problem is a special case of Problem (29) where $C = -\frac{1}{4}L$, $A_i = \mathbf{1}_i\mathbf{1}_i^T \in \mathcal{M}^{n_s}$, for $i = 1, \dots, m$, $b = \mathbf{1}$, and $\mathbf{A}$, $\mathbf{c}$, A, and c are all zeros.

Problem (36) is in fact the unconstrained and unbounded version of the eigenvalue optimization problem (31). That is

$$\max_{y \in I\!R^n} n_s \lambda_{min}(C + Diag(y)) + \mathbf{1}^T y. \tag{37}$$

Table 3: Performance of SOCCSM on the max-cut and Lovasz theta problems

name	n_s	m	LC	SOCC	dim	f	cpu
mcp100	100	100	91	22	172	-2.269e+02	12
mcp124-1	124	124	126	35	243	-1.423e+02	27
mcp124-2	124	124	106	28	205	-2.707e+02	23
mcp124-3	124	124	79	25	160	-4.693e+02	16
mcp124-4	124	124	31	20	96	-8.673e+02	13
mcp250-1	250	250	228	59	428	-3.184e+02	217
mcp250-2	250	250	148	53	321	-5.337e+02	142
mcp250-3	250	250	42	45	205	-9.844e+02	92
mcp250-4	250	250	18	41	164	-1.688e+03	55
mcp500-1	500	500	425	98	800	-6.001e+02	1046
mcp500-2	500	500	296	84	623	-1.073e+03	1120
mcp500-3	500	500	65	60	303	-1.852e+03	426
mcp500-4	500	500	48	42	226	-3.579e+03	225
toruspm3-8-50	512	512	736	10	779	-5.289e+02	2419
maxG11	800	800	522	305	1613	-6.310e+02	12653
theta1	50	103	204	1	206	-2.303e+01	22
theta2	100	497	1056	31	1179	-3.300e+01	2516
theta3	150	1105	1699	79	1949	-4.241e+01	5642

The Lovasz theta problem is the following semidefinite programming relaxation of the independent set problem:

$$\begin{aligned} \min \quad & -\mathbf{1}\,\mathbf{1}^T \bullet X \\ s.t. \quad & \\ & E_{ij} \bullet X = 0, \quad \text{for } \{i,j\} \in E \\ & I \bullet X = 1 \\ & X \succeq 0 \end{aligned} \tag{38}$$

where E is the edge set and E_{ij} is a matrix with one in ij-th and ji-th positions and zero elsewhere. The equivalent eigenvalue optimization of the dual semidefinite program to the Lovasz theta problem reads

$$\max \lambda_{min}(-\mathbf{1}\,\mathbf{1}^T + \sum_{ij\in E} E_{ij} y_{ij}). \tag{39}$$

Both Problems (37) and (39) are unconstrained versions of Problem (31). A minor modification in Algorithm 1 is required to handle these problems.

Since the set of localization must be bounded, we start from an artificial ball constraint $\|y\| \leq \beta$, where $\beta > 0$ is arbitrarily selected. Of course, the smaller the set of localization, the faster the convergence. Therefore we initialize from $\beta = 1$. When an iteration gets close to this boundary we increase the radius of the ball constraint by a factor of, say 1.5. While this approach gives flexibility to the algorithm to further minimize the objective function and solve the unbounded problem, it controls the size of the localization set.

An alternative approach is to impose the box constraint $l \leq y \leq u$, where l and u are arbitrary m-vectors ($l < u$). At each iteration, if y_i gets close to its upper or lower bounds, the artificial bound is multiplied by 1.5 and moved away. There are advantages and disadvantages with each of the above choices. The 2-norm constraint can be cast as a single second-order cone constraint and easily incorporated into our algorithm. Clearly this has an advantage over the $2m$ linear box constraints, especially when m is large. However, updating the ball constraint when the current query point approaches its boundary will enlarge the entire localization set. Consequently more cuts are needed to solve the problem, and it may result in slower overall convergence.

Table 3 illustrates our computational results on some challenging examples of the maxcut and Lovasz theta problems selected from SDPLIB [4]. In this table the column under "f" illustrates the value of the best lower bound obtained by the algorithm.

The test problems selected from SDPLIB are amongst problems that are known to be difficult to solve. Comparing *dim* and the cpu time of each class of problem set, Table 3 further confirms that SOCCSM works better on dense problems than it does on sparse ones. This is because the sparse problems start with a rather large set of localization, and therefore too many cuts are required before the desired accuracy is reached.

We report these results to show the performance of our algorithm on sparse problems, although SOCCSM performs poorly on most of these problems.

We do not compare our cpu time in Table 3 with that of other software. This is because SDPLR and SDPT3 have proven to work efficiently with the sparse problems. Since all of the test problems in this table are somewhat sparse, these software packages have an advantage in this area and of course work better. However, the results are somewhat comparable to, and sometimes better than, those reported in Sivaramakrishnan et al. [27].

7 Conclusions

We presented an analytic center cutting surface method that uses mixed linear and multiple second-order cone cuts. We discussed both theory and application of this technique. We proved that the analytic center can be recovered after adding p second-order cone cuts in $O(p\log(p+1))$ Newton steps and that the overall algorithm is fully polynomial. This result is an extension of the complexity results of ACCPM. In particular it extends the complexity result of Oskoorouchi and Goffin [23] from single to multiple SOCCs.

We implemented Algorithm 1 on mixed linear-quadratic-semidefinite programming problems with bounded feasible region. We discovered that SOCCSM can efficiently handle randomly generated problems with large and dense coefficient matrices. The numerical results presented in this paper revealed three interesting observations: 1) a comparison of the CPU time between our algorithm and SDPLR, SDPT3, and SeDuMi shows that SOCCSM outperforms primal-dual interior point methods on dense problems; 2) SOCCSM performs better on problems with constraints, in fact the more constraints, the faster the convergence; and 3) comparing the numerical results of this paper and that of Oskoorouchi and Goffin [24], illustrates the computational benefits of utilizing the second-order cone relaxation on semidefinite cuts.

References

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

[2] D. S. Atkinson and P. M. Vaidya, *A cutting plane algorithm that uses analytic centers*, Mathematical Programming, series B, 69 (1995), pp. 1–43.

[3] V. L. Basescu and J. E. Mitchell, *An analytic center cutting plane approach for conic programming*, Rensselaer Polytechnic Institute, Troy, NY 12180, June 2005.

[4] B. Borchers, *SDPLIB 1.2, A library of semidefinite programming problems*, Optimization Methods and Software 11(1999) , pp. 613-623.

[5] S. Burer and C. Choi, *Computational enhancements in low-rank semidefinite programming*, Optimization Methods and Software, 21(3), pp. 493–512, 2006.

[6] S. Burer and R.D.C. Monteiro, *A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization*, Mathematical Programming (series B), 95(2), pp. 329-357, 2003.

[7] S. Burer and R.D.C. Monteiro, *Local Minima and Convergence in Low-Rank Semidefinite Programming*, Mathematical Programming (series A), 103(3), pp. 427–444, 2005.

[8] S. K. Chua, K. C. Toh, and G. Y. Zhao, *An analytic center cutting plane method with deep cuts for semidefinite feasibility problems*, Journal of Optimization Theory and Applications, 123 (2004), pp. 291–318.

[9] J.-L. Goffin, Z.-Q. Luo, and Y. Ye, *Complexity analysis of an interior cutting plane for convex feasibility problems*, SIAM Journal on Optimization, 6 (1996), pp. 638–652.

[10] J.-L. Goffin and J.-P. Vial, *Multiple cuts in the analytic center cutting plane methods*, SIAM Journal on Optimization, 11 (2000), pp. 266–288.

[11] C. Helmberg, *Numerical evaluation of SB method*, Mathematical Programming, 95 (2003) pp. 381–406.

[12] C. Helmberg, *Semidefinite programming for Combinatorial Optimization*, Habilitationsschrift, ZIB-Report 00-34, October 2000.

[13] C. Helmberg and F. Rendl, *A spectral bundle method for semidefinite programming*, SIAM Journal on Optimization, 10(3) (2000) pp. 673–696.

[14] S. Kim, M. Kojima and M. Yamashita, *Second Order Cone Programming Relaxation of a Positive Semidefinite Constraint*, Optimization Methods and Software, Vol.18 (5) 535-541 (2003).

[15] K. Krishnan and J. E. Mitchell, *A unifying framework for several cutting plane methods for semidefinite programming*, Optimization Methods and Software, Vol.21 (1) 57-74 (2006).

[16] K. Krishnan and J. E. Mitchell, *A semidefinite programming based polyhedral cut-and-price approach for the maxcut problem*, Computational Optimization and Applications, Vol.33 (1) 51-71 (2006).

[17] A. S. Lewis and M. L. Overton, *Eigenvalue Optimization*, Acta Numerica, (1996), pp. 149–190.

[18] Z.-Q. Luo and J. Sun, *An analytic center based on column generation algorithm for convex quadratic feasibility problems*, SIAM Journal on Optimization, 9(1) (1998) pp. 217–235.

[19] Z.-Q. Luo and J. Sun, *A Polynomial Cutting Surfaces Algorithm for the Convex Feasibility Problem Defined by Self-Concordant Inequalities*, Computational Optimization and Applications 15 (2000) 167-191.

[20] H.J. Lüthi and B. Büeler, *The Analytic Center Quadratic Cut Method (ACQCM) for Strongly Monotone Variational Inequality Problems*, SIAM Journal on Optimization, 10(2) (2000), pp. 415–426.

[21] J. E. Mitchell and M.J. Todd, *Solving combinatorial optimization problems using Karmarkar's algorithm*, Mathematical Programming, 56 (1992), pp. 245–284.

[22] M. R. Oskoorouchi and J. L. Goffin, *The analytic center cutting plane method with semidefinite cuts*, SIAM Journal on Optimization, 13 (4) (2003), 1029–1053.

[23] M. R. Oskoorouchi and J. L. Goffin, *An interior point cutting plane method for the convex feasibility problem with second-order cone inequalities*, Mathematics of Operations Research, Vol. 30, No. 1, February 2005, pp. 127–149.

[24] M. R. Oskoorouchi and J. L. Goffin, *A matrix generation approach for eigenvalue optimization*, Mathematical Programming, Ser. A, forthcoming.

[25] M. L. Overton, *Large-scale optimization of eigenvalues*, SIAM Journal on Optimization, 2 (1992), pp. 88–120

[26] F. Sharifi Mokhtarian and J.-L. Goffin, *An analytic center quadratic cut method for the convex quadratic feasibility problem*, Math. Program., Ser. A 93 (2002) 2, pp. 305–325.

[27] K. Sivaramakrishnan, G. Plaza, and T. Terlaky, *A conic interior point decomposition approach for large scale semidefinite programming*, Technical Report, Department of Mathematics, North Carolina State University, Raleigh, NC, 27695, December 2005.

[28] G. Sonnevend, *New algorithms in convex programming based on a notation of center and on rational extrapolations*, International Series of Numerical Mathematics, Birkhauser Verlag, Basel, Switzerland, 84 (1988), pp. 311–327.

[29] J. F. Sturm, *Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones*, Optimization Methods and Software, Vol. 11-12, pp. 625–653, 1999.

[30] J. F. Sturm, *Implementation of Interior Point Methods for Mixed Semidefinite and Second Order Cone Optimization Problems*, Optimization Methods and Software, Volume 17, Number 6, pp. 1105–1154, 2002.

[31] J. Sun, K. C. Toh, and G. Y. Zhao, *An analytic center cutting plane method for semidefinite feasibility problems*, Mathematics of Operations Research, Volume: 27 (2), May 2002 pp. 332–346

[32] K. C. Toh, M. J. Todd, and R. H. Tutuncu, *SDPT3–a Matlab software package for semidefinite programming, version 2.1*, Optimization Methods and Software, 11 (1999), pp. 545–581.

[33] K. C. Toh, G. Y. Zhao, and J. Sun, *A multiple-cut analytic center cutting plane method for semidefinite feasibility problems*, SIAM Journal on Optimization, 12 (2002) 4, pp. 1126–1146

[34] R.H Tutuncu, K.C. Toh, and M.J. Todd, *Solving semidefinite-quadratic-linear programs using SDPT3*, Mathematical Programming Ser. B, 95, pp. 189–217, (2003).

[35] Y. Ye, *A potential reduction algorithm allowing column generation*, SIAM Journal on Optimization, 2 (1992), pp. 7–20.

[36] Y. Ye, *Complexity analysis of the analytic center cutting plane method that uses multiple cuts*, Mathematical Programming, 78 (1997), pp. 85–104.