Optimization Methods and Software
Vol. 00, No. 00, August 2014, 1–28

RESEARCH ARTICLE

The Second-Order Cone Eigenvalue Complementarity Problem

Luís M. Fernandes[a*‡] and Masao Fukushima[b†] and Joaquim J. Júdice[c‡] and Hanif D. Sherali[d]

[a]*Instituto Politécnico de Tomar and Instituto de Telecomunicações, Portugal*; [b] *Faculty of Science and Technology, Nanzan University, Seto, Aichi 489-0863, Japan*; [c]*Instituto de Telecomunicações, Portugal*; [d]*Grado Department of Industrial & Systems Engineering, Virginia Tech, Blacksburg, VA, USA*

(*Received 00 Month 200x; in final form 00 Month 200x*)

The Eigenvalue Complementarity Problem (EiCP) differs from the traditional eigenvalue problem in that the primal and dual variables belong to a closed and convex cone K and its dual, respectively, and satisfy a complementarity condition. In this paper we investigate the solution of the Second-Order Cone EiCP (SOCEiCP) where K is the Lorentz cone. We first show that the SOCEiCP reduces to a special Variational Inequality Problem on a compact set defined by K and a normalization constraint. This guarantees that SOCEiCP has at least one solution, and a new enumerative algorithm is introduced for finding a solution to this problem. The method is based on finding a global minimum of an appropriate nonlinear programming formulation NLP of the SOCEiCP using a special branching scheme along with a local nonlinear optimizer that computes stationary points on subsets of the feasible region of NLP associated with the nodes generated by the algorithm. A semi-smooth Newton's method is combined with this enumerative algorithm to enhance its numerical performance. Our computational experience illustrates the efficacy of the proposed techniques in practice.

Keywords: Eigenvalue Problems, Complementarity Problems, Nonlinear Programming, Global Optimization.

AMS Subject Classification: 90B60, 90C33, 90C30, 90C26

1. Introduction

The *Eigenvalue Complementarity Problem* (**EiCP**) [25, 28] consists of finding a real number λ and a vector $x \in \mathbb{R}^n\backslash\{0\}$ such that

$$\begin{aligned} w &= (\lambda B - A)x \\ x^T w &= 0, \\ w \geq 0, &\, x \geq 0, \end{aligned} \tag{1}$$

where $w \in \mathbb{R}^n$, $A, B \in \mathbb{R}^{n\times n}$, and B is positive definite (PD), i.e., $x^T Bx > 0$ for all $x \neq 0$. This problem has been introduced in [28] and finds many applications as described in [1, 24, 25].

*Corresponding author. Email: lmerca@co.it.pt

†The research of Masao Fukushima was partially supported by Grant-in-Aid for Scientific Research from Japan Society for the Promotion of Science.

‡The research of Luís M. Fernandes and Joaquim J. Júdice was partially supported in the scope of R&D Unit UID/EEA/50008/2013, financed by the applicable financial framework (FCT/MEC through national funds and when applicable co-funded by FEDER - PT2020 partnership agreement.)

The authors also thank two anonymous referees for their constructive and insightful comments.

ISSN: 1055-6788 print/ISSN 1029-4937 online
© 2014 Taylor & Francis
DOI: 10.1080/1055.6788.YYYY.xxxxxx
http://www.tandfonline.com

A number of algorithms have been proposed for the solution of the EiCP [2, 7, 10, 11, 15–17, 19, 21, 24, 29, 34]. The EiCP can be viewed as a mixed nonlinear complementarity problem [9], where the complementary vectors x and w belong to the cone $K = \mathbb{R}^n_+$ and its dual $K^* = \mathbb{R}^n_+$, respectively. Recently an extension of the EiCP has been introduced in [1] where K is the Lorentz cone. This so-called *Second-Order Cone Eigenvalue Complementarity Problem* (**SOCEiCP**) can be stated as follows:

Find a real number λ and a vector $x \in \mathbb{R}^n \backslash \{0\}$ such that

$$\begin{aligned} w &= (\lambda B - A)x \\ x^T w &= 0, \\ x \in K, &\quad w \in K^*, \end{aligned} \tag{2}$$

where $A, B \in \mathbb{R}^{n\times n}$, B is positive definite (PD), and K is the second-order or Lorentz cone defined by

$$K = K_1 \times K_2 \times \cdots \times K_r, \tag{3}$$

with

$$K_i = \{x^i \in \mathbb{R}^{n_i} \colon \|\bar{x}^i\| \leq x^i_0\} \subseteq \mathbb{R}^{n_i}, \quad i = 1, \ldots, r, \tag{4}$$

and where $r \geq 1$, $\sum_{i=1}^r n_i = n$, $x^i = (x^i_0, \bar{x}^i) \in \mathbb{R}^{n_i}$, $i = 1, \ldots, r$, and $x = (x^1, x^2, \ldots, x^r) \in \mathbb{R}^n$. Here, $\| \, . \, \|$ denotes the Euclidean norm and the dual cone K^* of K is defined by

$$K^* = \{y \in \mathbb{R}^n \colon y^T x \geq 0, \forall x \in K\}. \tag{5}$$

A number of semi-smooth Newton type algorithms have been discussed in [1] for finding a solution to the SOCEiCP. However, none of these methods possess global convergence and there is no guarantee that they converge to a solution of the SOCEiCP even if a line-search procedure is employed [1]. Numerical results reported in [1] indicate that these algorithms perform very well when they are successful.

In this paper, we investigate alternative approaches for finding a solution of the SOCEiCP. We start by showing the equivalence of the SOCEiCP to the following Variational Inequality Problem:

VI$(F, \bar{K})$: Find $x \in \bar{K}$ such that

$$F(x)^T (z - x) \geq 0, \, \forall z \in \bar{K},$$

where $\bar{K} = K \cap \Delta$, with

$$\Delta = \left\{ x \in \mathbb{R}^n : \sum_{i=1}^r x^i_0 = 1, \, x^i_0 \geq 0, \, i = 1, \ldots, r \right\}, \tag{6}$$

and where the function $F : \mathbb{R}^n \to \mathbb{R}^n$ is given by

$$F(x) = \frac{x^T A x}{x^T B x} B x - A x. \tag{7}$$

Since $B \in$ PD, this function is well-defined and continuous on $\bar{K}$. Furthermore, as $\bar{K}$ is a nonempty compact set, this VI$(F, \bar{K})$ has at least one solution [9] and thus so does SOCEiCP.

Due to the equivalence between SOCEiCP and VI$(F, \bar{K})$, local and fast algorithms for Variational Inequality Problems described in [9] can be used to solve the SOCEiCP. However, the mapping F is not monotone [9] and there is no theoretical guarantee that these algorithms converge to a solution of the SOCEiCP, even if line-search procedures are employed. As shown later in this paper, if both the matrices A and B are symmetric, the so-called Symmetric SOCEiCP is equivalent to finding a stationary point of the following nonlinear program:

$$\textbf{NLP}_1\textbf{:}\ \text{Maximize } f(x) = \frac{x^T Ax}{x^T Bx}$$
$$x \in \bar{K}$$

where $\bar{K} = K \cap \Delta$ and Δ is given by (6). In the asymmetric case (where at least one of the matrices A or B is asymmetric) this reduction no longer holds, but we can formulate another nonlinear programming problem for dealing with the SOCEiCP. As in [15], we introduce an additional vector $y = \lambda x$ to construct the following nonlinear program, where the latter relationship is accommodated within the objective function along with the complementarity constraint:

$$\textbf{NLP}_2\textbf{:}\ \text{Minimize } f(x, w, \lambda, y) = \|y - \lambda x\|^2 + (x^T w)^2 \tag{8}$$
$$\text{subject to } w - By + Ax = 0 \tag{9}$$
$$\|\bar{x}^i\|^2 \leq (x_0^i)^2, \quad i = 1, \dots, r \tag{10}$$
$$\|\bar{w}^i\|^2 \leq (w_0^i)^2, \quad i = 1, \dots, r \tag{11}$$
$$\sum_{i=1}^{r} (e^i)^T x^i - 1 = 0 \tag{12}$$
$$\sum_{i=1}^{r} (e^i)^T y^i - \lambda = 0 \tag{13}$$
$$x_0^i \geq 0, \quad i = 1, \dots, r \tag{14}$$
$$w_0^i \geq 0, \quad i = 1, \dots, r, \tag{15}$$

where $e^i = (1, 0, \dots, 0)^T \in \mathbb{R}^{n_i}$, $w^i = (w_0^i, \bar{w}^i) \in \mathbb{R}^{n_i}$, $y^i = (y_0^i, \bar{y}^i) \in \mathbb{R}^{n_i}$, $i = 1, \dots, r$, and $w = (w^1, w^2, \dots, w^r) \in \mathbb{R}^n$, $y = (y^1, y^2, \dots, y^r) \in \mathbb{R}^n$. (Note that $K^* = K$ here, as stated in Proposition 2.1 of Section 2 below.)

It is obvious that any global minimum of NLP$_2$ with an objective function value equal to zero provides a solution of the SOCEiCP. In this paper we first investigate when a stationary point of NLP$_2$ is a solution of SOCEiCP. We are able to prove that this occurs if and only if the Lagrange multipliers associated with the equalities (12) and (13) are equal to zero. This result is interesting as it shows that, contrary to the symmetric case, a stationary point of NLP$_2$ is not in general sufficient for finding a solution of SOCEiCP. Furthermore, the sufficient condition is not too strong and this suggests the use of an enumerative algorithm similar to [6] to find a solution of SOCEiCP. This method exploits a binary tree that is constructed by bracketing the interval $[0, 1]$ of the primal variables x_i, $i = 1, \dots, n$. Additionally, stationary points of an augmented NLP$_2$ are computed for each node in a systematic way until one is found having an objective function value equal to zero. This augmented NLP$_2$ is defined by using the well-known Reformulation-Linearization Technique [30, 32] as explained later in this paper. It is shown that this algorithm induces global convergence to a solution of SOCEiCP. Numerical experiments with this enumerative method indicates that it performs well for most small- and medium-scale instances. However, for certain test problems, the algorithm is only able to obtain a feasible solution in $\bar{K}$ with a small positive objective function value. Since semi-smooth Newton type algorithms for the SOCEiCP are local and fast algorithms, we follow the recommendation in [11] of combining the enumerative method with one of the semi-smooth Newton algorithms discussed in [1]. In this hybrid algorithm, the enumerative method is used as a safeguard and the process moves to the semi-smooth method when the current point $\bar{x}$ is close to a global minimum. By starting with this point $\bar{x}$, the semi-smooth algorithm is, in general, able to find a solution of the SOCEiCP. If the semi-smooth method fails, then the enumerative method is continued with the point $\bar{x}$ obtained as the initial point for the semi-smooth method. The process is repeated until a solution of the SOCEiCP is obtained by either the semi-smooth method or the enumerative method itself. Numerical experiments with this hybrid approach clearly indicate its efficacy in significantly enhancing the stand-alone enumerative method.

The remainder of this paper is organized as follows. The equivalence between the SOCEiCP and a Variational Inequality Problem and its consequences are presented in Section 2. The nonlinear programming formulation NLP$_2$ is discussed in Section 3. The enumerative method is described in Sections 4 and 5. A semi-smooth Newton's method and the hybrid algorithm are discussed in Section 6. Computational experience with the proposed algorithms is reported in Section 7, and conclusions are presented in the last section of the paper.

2. A Variational Inequality Formulation and Existence of a Solution

Consider again the SOCEiCP (2), where $x \in K$, $w \in K^*$, and K is given by (3)-(4). Let VI$(F, \bar{K})$ be the Variational Inequality Problem introduced in the previous section, where F is defined by (7) and where $\bar{K} = K \cap \Delta$, with Δ given by (6). Then the following results hold:

Proposition 2.1 *[3] $K^* = K$.*

Proposition 2.2 *[9] VI$(F, \bar{K})$ has at least one solution.*

Let

$$\tilde{K}_i = \{x^i \in \mathbb{R}^{n_i} : \|\bar{x}^i\|^2 \leq (x_0^i)^2\},\ i = 1, \ldots, r$$

and

$$\tilde{K} = (\tilde{K}_1 \times \ldots \times \tilde{K}_r) \cap \Delta. \tag{16}$$

Hence,

$$\bar{K} = \tilde{K}. \tag{17}$$

We next exhibit the reduction of SOCEiCP to VI$(F, \bar{K})$:

Theorem 2.3 *If $x \in \bar{K}$ is a solution of VI$(F, \bar{K})$ then $\left(x, \lambda = \frac{x^TAx}{x^TBx}\right)$ is a solution of SOCEiCP.*

Proof

(i) Consider first the case of $x_0^i \neq 0$ for all $i = 1, \ldots, r$. It follows from (17) and the definition of VI$(F, \bar{K})$ that x is a solution of VI$(F, \bar{K})$ if and only if $z = x$ is a minimizer of the following problem:

$$\text{Minimize } F(x)^T z$$
$$\text{subject to } z \in \tilde{K}.$$

Since the Linear Independence Constraint Qualification holds at x, we have that x satisfies the following KKT conditions ([4], Theorem 4.2.13):

$$F(x) = C\mu + E\alpha + \gamma e \tag{18}$$
$$\|\bar{x}^i\|^2 \leq (x_0^i)^2,\ i = 1, \dots, r \tag{19}$$
$$\mu_i \left(\|\bar{x}^i\|^2 - (x_0^i)^2\right) = 0,\ i = 1, \dots, r \tag{20}$$
$$\mu_i \geq 0,\ i = 1, \dots, r \tag{21}$$
$$x_0^i \geq 0,\ \alpha_i \geq 0,\ i = 1, \dots, r \tag{22}$$
$$x_0^i \alpha_i = 0,\ i = 1, \dots, r \tag{23}$$
$$e^T x = 1 \tag{24}$$

where

$$C = \begin{bmatrix} 2x_0^1 & 0 & \dots & 0 \\ -2\bar{x}^1 & 0^1 & \dots & 0^1 \\ 0 & 2x_0^2 & \dots & 0 \\ 0^2 & -2\bar{x}^2 & \dots & 0^2 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 2x_0^r \\ 0^r & 0^r & \dots & -2\bar{x}^r \end{bmatrix} \in \mathbb{R}^{n \times r},\quad E = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0^1 & 0^1 & \dots & 0^1 \\ 0 & 1 & \dots & 0 \\ 0^2 & 0^2 & \dots & 0^2 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 1 \\ 0^r & 0^r & \dots & 0^r \end{bmatrix} \in \mathbb{R}^{n \times r},$$

$$e = \begin{bmatrix} 1 \\ 0^1 \\ 1 \\ 0^2 \\ \vdots \\ 1 \\ 0^r \end{bmatrix} \in \mathbb{R}^n,\quad \mu = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_r \end{bmatrix} \in \mathbb{R}^r,\quad \alpha = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_r \end{bmatrix} \in \mathbb{R}^r.$$

Note that 0^i is a null vector of dimension n_i, $i = 1, \dots, r$, 0 is the real number and μ_i, α_i, γ are the Lagrange multipliers associated with the constraints $\|\bar{x}^i\|^2 \leq (x_0^i)^2$, $x_0^i \geq 0$, and $e^T x = 1$, respectively. Multiplying (18) by x^T and using (20), (23). and (24), we get

$$x^T F(x) = \gamma.$$

From (7),

$$x^T F(x) = \frac{x^T A x}{x^T B x} x^T B x - x^T A x = 0.$$

Hence $\gamma = 0$. Therefore, by letting $\lambda = \frac{x^T A x}{x^T B x}$ and $w = C\mu + E\alpha$, we obtain from (7), (18), and $\gamma = 0$ that

$$w = \lambda Bx - Ax.$$

Moreover, we have

$$w^T x = 0$$

and from (18), (20), and $w = F(x)$ that

$$\|\bar{w}^i\| - w_0^i = 2\mu_i \|\bar{x}^i\| - 2\mu_i x_0^i - \alpha_i = -\alpha_i \leq 0,\ i = 1, \ldots, r,$$

which means that $w \in K = K^*$. Hence (x, λ) is a solution of SOCEiCP.

(ii) Consider now the case of $x_0^j = 0$ for some j. Then $x^j = 0$ and, by (24), $r \geq 2$. We assume, for simplicity, that there is exactly one such j and that $j = 1$, i.e., $x = (x^1, u) \in \mathbb{R}^{n_1} \times \mathbb{R}^{n-n_1}$ with

$$x^1 = 0,\ u = (x^2, \ldots, x^r) \text{ and } x^i \neq 0,\ i = 2, \ldots, r.$$

(The case of more than one such j, but of course not all by virtue of (12), can be handled similarly.) Now, recall that

$$w = F(x) = \frac{x^T Ax}{x^T Bx} Bx - Ax = \lambda Bx - Ax$$

with

$$\lambda = \frac{x^T Ax}{x^T Bx}.$$

Now, decompose A and B according to the decomposition of x, i.e.

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \quad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

with $A_{11}, B_{11} \in \mathbb{R}^{n_1 \times n_1}$, $A_{22}, B_{22} \in \mathbb{R}^{(n-n_1)\times(n-n_1)}$. Then we can write

$$w = \begin{pmatrix} w^1 \\ v \end{pmatrix} = F(x) = F\begin{pmatrix} 0 \\ u \end{pmatrix} = \begin{bmatrix} \lambda B_{12}u - A_{12}u \\ \lambda B_{22}u - A_{22}u \end{bmatrix} = \begin{bmatrix} F_1(u) \\ F_2(u) \end{bmatrix}$$

and

$$\lambda = \frac{x^T Ax}{x^T Bx} = \frac{u^T A_{22}u}{u^T B_{22}u}.$$

Since $x = (0, u)$ is a solution of $\mathrm{VI}(F, \bar{K})$, we have

$$F(x)^T(z - x) = (w^1)^T z^1 + F_2(u)^T(y - u) \geq 0 \tag{25}$$

for all $z^1 \in \mathbb{R}^{n_1}$ and $y \in \mathbb{R}^{n-n_1}$ such that $(z^1, y) \in \bar{K}$. Then, putting $z^1 = 0$ in (25) shows that u is a solution of

$$\mathrm{VI}(F_2, \bar{\bar{K}}): \quad F_2(u)^T(y - u) \geq 0,\ \forall\, y \in \bar{\bar{K}},$$

where

$$\bar{\bar{K}} = (K_2 \times \ldots \times K_r) \cap \bar{\Delta} \subseteq \mathbb{R}^{n-n_1}$$

with

$$\bar{\Delta} = \{u \in \mathbb{R}^{n-n_1} : \sum_{j=2}^{r} x_0^j = 1,\ x_0^i \geq 0,\ i = 2, \ldots, r\}.$$

Moreover, it follows from Part (i) of the proof that $\left(u, \lambda = \frac{u^T A_{22} u}{u^T B_{22} u}\right)$ is a solution of

$$\begin{array}{lll} \text{SOCEiCP}_2 : & v = \lambda B_{22} u - A_{22} u & \\ & \|\bar{x}^i\| \leq x_0^i, & i = 2, \ldots, r \\ & \|\bar{w}^i\| \leq w_0^i, & i = 2, \ldots, r \\ & u^T v = 0, & \end{array}$$

where $u = (x^2, \ldots, x^r), v = (w^2, \ldots, w^r) \in \mathbb{R}^{n-n_1}$.

Therefore, we can deduce that $\left(x = \begin{pmatrix} 0 \\ u \end{pmatrix}, \lambda = \frac{u^T A_{22} u}{u^T B_{22} u}\right)$ is a solution of SOCEiCP, provided

$$\|\bar{w}^1\| \leq w_0^1 \tag{26}$$

where

$$w^1 = \begin{pmatrix} w_0^1 \\ \bar{w}^1 \end{pmatrix} = \lambda B_{12} u - A_{12} u.$$

In the following, we will show that (26) holds true. First, note that

$$F(x)^T x = \begin{pmatrix} w^1 \\ v \end{pmatrix}^T \begin{pmatrix} 0 \\ u \end{pmatrix} = v^T u = 0.$$

Thus, since $x = (0, u)$ is a solution of VI$(F, \bar{K})$, it satisfies

$$F(x)^T (y - x) = F(x)^T y \geq 0,\ \forall\, y \in \bar{K}. \tag{27}$$

In view of the homogeneity of the last inequality, (27) is equivalent to

$$F(x)^T y \geq 0,\ \forall\, y \in K. \tag{28}$$

Noticing $F(x) = (F_1(x), F_2(x)) = (w^1, v)$ and putting $y = (y^1, 0) \in \mathbb{R}^{n_1} \times \mathbb{R}^{n-n_1}$, we obtain from (28)

$$(w^1)^T y^1 \geq 0,\ \forall\, y^1 \in K_1,$$

which implies $w^1 \in K_1^* = K_1$, that is, (26) holds.

■

As a consequence of this theorem and Proposition 2.2, we can state the following existence result for the SOCEiCP [29]:

Theorem 2.4 *SOCEiCP has at least one solution.*

This theorem shows that the SOCEiCP can be solved by solving the Variational Inequality Problem VI($F, \bar{K}$). However, the mapping F is not monotone and it is quite difficult to solve this VI [9]. As in [7], we could design a projection algorithm to deal with the SOCEiCP. However, there is no guarantee that such an algorithm would converge to a solution of the SOCEiCP [7].

Consider now the symmetric SOCEiCP, that is, the case where A and B are both symmetric matrices. We still assume $B \in$ PD. Let f be the Rayleigh quotient function defined by

$$f(x) = \frac{x^T Ax}{x^T Bx}.$$

Then for each $x \in \bar{K}$, the gradient of f at x is given by

$$\nabla f(x) = -\frac{2}{x^T Bx}\left[\frac{x^T Ax}{x^T Bx}Bx - Ax\right].$$

The following result is a consequence of Theorem 2.3:

Theorem 2.5 *If x is a stationary point of*

$$\max\{f(x) : x \in \bar{K}\},$$

then $\left(x, \lambda = \frac{x^T Ax}{x^T Bx}\right)$ is a solution of SOCEiCP.

The equality (17) and Theorem 2.5 show that the symmetric SOCEiCP can be solved by computing a stationary point $\bar{x}$ of the Rayleigh quotient function on the set $\tilde{K}$ defined by (16). This is a nonlinear program of the form:

$$\begin{aligned}
\textbf{NLP}_1\textbf{:}\ \text{Maximize}\ f(x) &= \frac{x^T Ax}{x^T Bx} \\
\text{subject to}\ \|\bar{x}^i\|^2 &\leq (x_0^i)^2,\ i = 1, \ldots, r \\
\sum_{i=1}^{r} x_0^i &= 1 \\
x_0^i &\geq 0,\ i = 1, \ldots, r.
\end{aligned}$$

Since the feasible set of NLP$_1$ is a nonempty compact set, such a stationary point $\bar{x}$ always exists. Furthermore, the corresponding eigenvalue is given by $\lambda = \frac{\bar{x}^T A\bar{x}}{\bar{x}^T B\bar{x}}$.

It is interesting to note that, due to (17), this NLP$_1$ is a nonlinear program with a constraint set defined by linear constraints and r nonlinear inequalities. There are many efficient algorithms [4, 22] and codes (for instance, [13, 20, 33]) for computing a stationary point for this program. So, the symmetric SOCEiCP can be solved efficiently. As stated before, the asymmetric case is much more difficult to deal with. In the next section we consider another NLP formulation that proves to be very helpful for resolving this latter case.

3. A Nonlinear Programming Formulation for the Asymmetric SOCEiCP

As in [15], we introduce r vectors $y^i \in \mathbb{R}^{n_i}$ such that

$$y^i = \lambda x^i, \quad i = 1, \ldots, r, \tag{29}$$

in order to derive the following nonlinear programming formulation NLP$_2$ of the SOCEiCP given by (8)-(15). It is important to add that (12) and (13) are equivalent to $\sum_{i=1}^{r} x_0^i = 1$ and $\sum_{i=1}^{r} y_0^i = \lambda$, respectively. Note that, whereas the regions defined by the individual constraints (10) and (11) are nonconvex, their intersection with (14) represents a convex region due to (17). Hence, the feasible region of NLP$_2$ is convex (note that NLP$_1$ also shares this property). Moreover, the functions associated with the constraints (10) and (11) are differentiable everywhere. We remark here that a similar construct was used by Sherali and Al-Loughani [31] for the multi-facility Euclidean location problem. The following result holds:

PROPOSITION 3.1 *SOCEiCP has a solution (x, w, λ) if and only if (x, w, λ, y) is a global minimum of NLP$_2$ with $f(x, w, \lambda, y) = 0$.*

Since any global minimum of NLP$_2$ is a stationary point and a stationary point is much easier to compute, it is important to investigate when a stationary point of NLP$_2$ is a solution of the SOCEiCP. This is addressed by the next result.

PROPOSITION 3.2 *A stationary point $(x^*, w^*, \lambda^*, y^*)$ of NLP$_2$ is a solution of SOCEiCP if and only if $\gamma = \theta = 0$, where γ and θ are the Lagrange multipliers associated with the constraints (12) and (13), respectively.*

Proof Let $\alpha \in \mathbb{R}^n$, $\beta \in \mathbb{R}^r$, $\mu \in \mathbb{R}^r$, $\gamma \in \mathbb{R}$, $\theta \in \mathbb{R}$, $\delta \in \mathbb{R}^r$, and $\eta \in \mathbb{R}^r$ be the Lagrange multipliers associated with the constraints (9), (10), (11), (12), (13), (14), and (15), respectively. Denote

$$C = \begin{bmatrix} 2w_0^1 & 0 & \ldots & 0 \\ -2\bar{w}^1 & 0^1 & \ldots & 0^1 \\ 0 & 2w_0^2 & \ldots & 0 \\ 0^2 & -2\bar{w}^2 & \ldots & 0^2 \\ \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & \ldots & 2w_0^r \\ 0^r & 0^r & \ldots & -2\bar{w}^r \end{bmatrix} \in \mathbb{R}^{n\times r}, \; D = \begin{bmatrix} 2x_0^1 & 0 & \ldots & 0 \\ -2\bar{x}^1 & 0^1 & \ldots & 0^1 \\ 0 & 2x_0^2 & \ldots & 0 \\ 0^2 & -2\bar{x}^2 & \ldots & 0^2 \\ \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & \ldots & 2x_0^r \\ 0^r & 0^r & \ldots & -2\bar{x}^r \end{bmatrix} \in \mathbb{R}^{n\times r},$$

$$E = \begin{bmatrix} 1 & 0 & \ldots & 0 \\ 0^1 & 0^1 & \ldots & 0^1 \\ 0 & 1 & \ldots & 0 \\ 0^2 & 0^2 & \ldots & 0^2 \\ \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & \ldots & 1 \\ 0^r & 0^r & \ldots & 0^r \end{bmatrix} \in \mathbb{R}^{n\times r}, \; e = \begin{bmatrix} e^1 \\ e^2 \\ \vdots \\ e^r \end{bmatrix} \in \mathbb{R}^n,$$

where 0^i is a null vector of dimension n_i, $i = 1, \ldots, r$ and 0 is the zero real number. Then, $(x^*, w^*, \lambda^*, y^*)$ along with $(\alpha, \beta, \mu, \gamma, \theta, \delta, \eta)$ satisfies the following KKT conditions [4] for NLP$_2$:

$$2(x^T w)x = \alpha + C\mu + E\eta \tag{30}$$

$$2(x^T w)w - 2\lambda(y - \lambda x) = A^T\alpha + D\beta + E\delta + \gamma e \tag{31}$$

$$2(y - \lambda x) = -B^T\alpha + \theta e \tag{32}$$

$$-2x^T(y - \lambda x) = -\theta \tag{33}$$

$$\beta_i[\|\bar{x}^i\|^2 - (x_0^i)^2] = 0, \quad i = 1, \dots, r \tag{34}$$

$$\mu_i[\|\bar{w}^i\|^2 - (w_0^i)^2] = 0, \quad i = 1, \dots, r \tag{35}$$

$$\delta_i x_0^i = \eta_i w_0^i = 0, \quad i = 1, \dots, r \tag{36}$$

$$\beta_i \geq 0, \ \mu_i \geq 0 \ \ \delta_i \geq 0, \ \eta_i \geq 0 \quad i = 1, \dots, r, \tag{37}$$

where β_i, μ_i, δ_i, η_i are the ith components of vectors β, μ, δ, $\eta \in \mathbb{R}^r$, respectively. Multiplying (30), (31), and (32) by w^T, x^T and y^T respectively, and noting (12), (13), and (36), we have

$$2(x^T w)^2 = w^T\alpha + 2\sum_{i=1}^{r} \mu_i(-\|\bar{w}^i\|^2 + (w_0^i)^2),$$

$$2(x^T w)^2 - 2\lambda x^T(y - \lambda x) = \alpha^T A x + 2\sum_{i=1}^{r} \beta_i(-\|\bar{x}^i\|^2 + (x_0^i)^2) + \gamma$$

$$2y^T(y - \lambda x) = -\alpha^T B y + \theta\lambda.$$

Adding these three equalities and using (9), (34), and (35), we get

$$4(x^T w)^2 + 2(y - \lambda x)^T(y - \lambda x) = \gamma + \theta\lambda.$$

Hence $(x^*, w^*, \lambda^*, y^*)$ along with some (γ, θ) satisfies

$$2(x^T w)^2 + 2f(x, w, \lambda, y) = \gamma + \theta\lambda. \tag{38}$$

Now:

(i) If $\gamma = \theta = 0$, then

$$f(x^*, w^*, \lambda^*, y^*) = 0 \tag{39}$$

and (x^*, w^*, λ^*) is a solution of SOCEiCP.

(ii) Conversely, suppose that (x^*, w^*, λ^*) is a solution of SOCEiCP. Then it satisfies $(x^*)^T w^* = 0$ and $w^* = \lambda^* B x^* - A x^*$. On the other hand, since $(x^*, w^*, \lambda^*, y^*)$ is a stationary point of NLP$_2$, it particularly satisfies $w^* = By^* - Ax^*$, which along with $w^* = \lambda^* B x^* - A x^*$ implies that $By^* = \lambda^* B x^*$. Since B is positive definite, we obtain $y^* = \lambda^* x^*$, which together with $(x^*)^T w^* = 0$ yields $f(x^*, w^*, \lambda^*, y^*) = 0$. Thus we have $\theta = 0$ by (33), and so we get $\gamma = 0$ by (38).

■

Remark 1 Since the constraints (9), (12), (13), (14), and (15) are linear, then the Linear Independence Constraint Qualification holds at a stationary point (x, w, λ, y) for NLP$_2$ if and only if $x_0^i > 0$ and $w_0^i > 0$ for all $i = 1, \dots, r$.

Note that Proposition 3.2 addresses a given stationary point for NLP$_2$. However, it is important to note that the following result holds true without the need of verifying any constraint qualification, noting Proposition 3.1 and that the gradient vanishes at the resulting solution to NLP$_2$.

Proposition 3.3 *For any given solution (x^*, w^*, λ^*) to SOCEiCP, there corresponds a stationary point $(x^*, w^*, \lambda^*, y^*)$ of NLP$_2$.*

4. An Enumerative Algorithm for the SOCEiCP

The proposed enumerative method for solving SOCEiCP aims at finding a global minimum for a reformulation of NLP$_2$ to be derived below. We begin by imposing lower and upper bounds on the variables as explained in the next section, i.e.,

$$c \le x \le d \tag{40}$$

$$l \le \lambda \le u \tag{41}$$

$$L \le w \le U, \tag{42}$$

where $x = [x^i_j]$, $w = [w^i_j]$, $c = [c^i_j]$, $d = [d^i_j]$, $L = [L^i_j]$, and $U = [U^i_j]$, $j = 0, \ldots, n_i - 1$, $i = 1, \ldots, r$. These bounds will play a critical role in designing our algorithm and assuring its convergence as discussed in Propositions 4.1-4.2 and Theorem 4.3 below.

The proposed algorithm explores a binary tree that is constructed by using a branching strategy that is similar to that discussed for the enumerative method designed for the Inverse EiCP (IEiCP) [6]. In order to reduce the overall search in this process, Reformulation-Linearization Technique (RLT)-based constraints are generated, as discussed below.

First, recall that the vectors y^i, $i = 1, \ldots, r$, are given by (29). Next, n additional variables z are introduced as follows:

$$z^i_j \equiv x^i_j w^i_j, \;\; j = 0, 1, \ldots, n_i - 1, \;\; i = 1, \ldots, r. \tag{43}$$

Following [32], we define (nonnegative) bound-factors based on (40)–(42) as $(x - c)$, $(d - x)$, $(\lambda - l)$, $(u - \lambda)$, $(w - L)$, and $(U - w)$. Accordingly, we generate the so-called bound-factor RLT constraints [32] by constructing the following product restrictions:

$$[l \le \lambda \le u] * [c^i_j \le x^i_j \le d^i_j], \;\; j = 0, 1, \ldots, n_i - 1, \;\; i = 1, \ldots, r$$

$$[L^i_j \le w^i_j \le U^i_j] * [c^i_j \le x^i_j \le d^i_j], \;\; j = 0, 1, \ldots, n_i - 1, \;\; i = 1, \ldots, r,$$

which are subsequently linearized using (29) and (43), respectively. In the notation used above, the first set of constraints denotes nonnegatively restricted products of each of the two bound-factors associated with the λ–variable with each of the two bound-factors associated with the x^i_j–variable, for each $j = 0, 1, \ldots, n_i - 1$, $i = 1, \ldots, r$, and likewise for the second set of constraints. These are all linearized using (29) and (43), and yield the following $8n$ bound-factor restrictions:

$$y^i_j \geq -ud^i_j + \lambda d^i_j + ux^i_j,\ \ j = 0, 1, \dots, n_i - 1,\ \ i = 1, \dots, r \tag{44}$$

$$y^i_j \geq -lc^i_j + \lambda c^i_j + lx^i_j,\ \ j = 0, 1, \dots, n_i - 1,\ \ i = 1, \dots, r \tag{45}$$

$$y^i_j \leq -ld^i_j + \lambda d^i_j + lx^i_j,\ \ j = 0, 1, \dots, n_i - 1,\ \ i = 1, \dots, r \tag{46}$$

$$y^i_j \leq -uc^i_j + \lambda c^i_j + ux^i_j,\ \ j = 0, 1, \dots, n_i - 1,\ \ i = 1, \dots, r \tag{47}$$

$$z^i_j \geq -L^i_j c^i_j + w^i_j c^i_j + L^i_j x^i_j,\ \ j = 0, 1, \dots, n_i - 1,\ \ i = 1, \dots, r \tag{48}$$

$$z^i_j \geq -U^i_j d^i_j + w^i_j d^i_j + U^i_j x^i_j,\ \ j = 0, 1, \dots, n_i - 1,\ \ i = 1, \dots, r \tag{49}$$

$$z^i_j \leq -U^i_j c^i_j + w^i_j c^i_j + U^i_j x^i_j,\ \ j = 0, 1, \dots, n_i - 1,\ \ i = 1, \dots, r \tag{50}$$

$$z^i_j \leq -L^i_j d^i_j + w^i_j d^i_j + L^i_j x^i_j,\ \ j = 0, 1, \dots, n_i - 1,\ \ i = 1, \dots, r. \tag{51}$$

Furthermore, from (10), (11), and by the Cauchy-Schwarz inequality [4, 22], condition $x^T w = 0$ is equivalent to

$$(x^i)^T w^i = 0,\ \ i = 1, \dots, r. \tag{52}$$

Thus, noting (43), we remove the term $(x^T w)^2$ from the objective function and add r linear constraints

$$\sum_{j=0}^{n_i-1} z^i_j = 0,\ \ i = 1, \dots, r. \tag{53}$$

Alternatively, we can replace the r linear constraints in (53) by

$$\sum_{i=1}^{r} \sum_{j=0}^{n_i-1} z^i_j = 0. \tag{54}$$

Despite the constraint (54) being weaker than (53), we use it in our computations because we found it to be more effective due to the peculiarities of the solver.

We hence derive the following formulation that we shall exploit in designing the proposed enumerative algorithm:

$$\textbf{NLP}_3\textbf{:}\ \text{Minimize}\ \tilde{f}(x, w, \lambda, y, z) = \|y - \lambda x\|^2 + \|z - x \circ w\|^2 \tag{55}$$

$$\text{subject to}\ w - By + Ax = 0 \tag{56}$$

$$\|\bar{x}^i\|^2 \leq (x^i_0)^2, \quad i = 1, \dots, r \tag{57}$$

$$\|\bar{w}^i\|^2 \leq (w^i_0)^2, \quad i = 1, \dots, r \tag{58}$$

$$\sum_{i=1}^{r} x^i_0 - 1 = 0 \tag{59}$$

$$\sum_{i=1}^{r} y^i_0 - \lambda = 0 \tag{60}$$

$$(44) - (51), (54) \tag{61}$$

$$(40) - (42), \tag{62}$$

where $x \circ w \in \mathbb{R}^n$ is the Hadamard product of x and w defined as the vector with components $x^i_j w^i_j,\ j = 0, 1, \ldots, n_i - 1,\ i = 1, \ldots, r$. Note that, as indicated above, NLP$_3$ is a convex-constrained program with a nonconvex objective function.

As for the NLP$_2$, the following two results hold for the NLP$_3$:

Proposition 4.1 *SOCEiCP has a solution* $(\tilde{x}, \tilde{w}, \tilde{\lambda})$ *if and only if* $(\tilde{x}, \tilde{w}, \tilde{\lambda}, \tilde{y}, \tilde{z})$ *is a global minimum of NLP*$_3$ *with* $\tilde{f}(\tilde{x}, \tilde{w}, \tilde{\lambda}, \tilde{y}, \tilde{z}) = 0$.

Proposition 4.2 *For any given solution* (x^*, w^*, λ^*) *to SOCEiCP, there corresponds a stationary point* $(x^*, w^*, \lambda^*, y^*, z^*)$ *of NLP*$_3$.

Next, we propose an enumerative algorithm that seeks a solution to SOCEiCP by computing a global minimum for NLP$_3$. This is done by exploring a binary tree that is constructed by bracketing the intervals $[c^i_j, d^i_j]$ associated with the variables $x^i_j,\ j = 0, 1, \ldots, n_i - 1,\ i = 1, \ldots, r$. At each node of this tree, a stationary point for NLP$_3$ is computed. Then, either the corresponding objective function value is zero and, by Proposition 4.1, a solution of SOCEiCP is attained, or two new nodes are generated as described in the steps of the enumerative algorithm below, where the notation NLP$_3(k)$ is used to represent the program (55)–(62) corresponding to node k.

Step 0 (Initialization) - Let $\epsilon > 0$ be a selected tolerance. Set $k = 1$, and find a stationary point $(\tilde{x}, \tilde{w}, \tilde{\lambda}, \tilde{y}, \tilde{z})$ of NLP$_3(1)$. (Note that NLP$_3(1)$ is feasible by Proposition 2.2, else, SOCEiCP would have no solution. Also, the derivation of the initial bounds on the variables used in formulating NLP$_3(1)$ will be discussed subsequently in Section 5.) If $\tilde{f}(\tilde{x}, \tilde{w}, \tilde{\lambda}, \tilde{y}, \tilde{z}) = 0$, then terminate with $(\tilde{x}, \tilde{w}, \tilde{\lambda}, \tilde{y}, \tilde{z})$ as a solution to SOCEiCP. Otherwise, let $L = \{1\}$ be the set of open nodes, set $UB(1) = \tilde{f}(\tilde{x}, \tilde{w}, \tilde{\lambda}, \tilde{y}, \tilde{z})$, and let $N = 1$ be the number of nodes generated.

Step 1 (Choice of node) - Select $k \in L$ such that

$$UB(k) = \min\{UB(i) \colon i \in L\},$$

and let $\tilde{c}$, $\tilde{d}$ be the vectors of lower and upper bounds for the variables x^i_j associated with this node k. Let $(\tilde{x}, \tilde{w}, \tilde{\lambda}, \tilde{y}, \tilde{z})$ be the stationary point that was previously found at this node. (Note that L is nonempty by Proposition 2.2, else SOCEiCP would have no solution.)

Step 2 (Branching rule) - Let

$$\psi = \max\left\{\frac{|\tilde{z}^i_j - \tilde{x}^i_j \tilde{w}^i_j|}{U^i_j - L^i_j}, \frac{|\tilde{y}^i_j - \tilde{\lambda}\tilde{x}^i_j|}{u - l} \colon j = 0, 1, \ldots, n_i - 1,\ i = 1, \ldots, r\right\} \quad (63)$$

and let the maximum in (63) be achieved by (i^*, j^*).

(i) If $\psi \leq \epsilon$, then $\tilde{\lambda}$ yields a complementary eigenvalue (within the tolerance ϵ) with $\tilde{x}$ being a corresponding eigenvector; terminate.

(ii) Else, partition the interval $[\tilde{c}^{i^*}_{j^*}, \tilde{d}^{i^*}_{j^*}]$ at node k into $[\tilde{c}^{i^*}_{j^*}, \hat{x}^{i^*}_{j^*}]$ and $[\hat{x}^{i^*}_{j^*}, \tilde{d}^{i^*}_{j^*}]$ to generate two new nodes,

$N+1$ and $N+2$, where $\hat{x}^{i^*}_{j^*}$ is given by

$$\hat{x}^{i^*}_{j^*} = \begin{cases} \tilde{x}^{i^*}_{j^*} & \text{if } \min\{(\tilde{x}^{i^*}_{j^*} - \tilde{c}^{i^*}_{j^*}), (\tilde{d}^{i^*}_{j^*} - \tilde{x}^{i^*}_{j^*})\} \geq 0.1(\tilde{d}^{i^*}_{j^*} - \tilde{c}^{i^*}_{j^*}) \\ \frac{\tilde{c}^{i^*}_{j^*} + \tilde{d}^{i^*}_{j^*}}{2} & \text{otherwise.} \end{cases} \tag{64}$$

Step 3 (Solve, Update, and Queue) - For each of $t = N+1$ and $t = N+2$, find a stationary point $(\tilde{x}, \tilde{w}, \tilde{\lambda}, \tilde{y}, \tilde{z})$ of Problem NLP$_3(t)$. If NLP$_3(t)$ is feasible, set $L \leftarrow L \cup \{t\}$ and $UB(t) = \tilde{f}(\tilde{x}, \tilde{w}, \tilde{\lambda}, \tilde{y}, \tilde{z})$. Otherwise, set $L \leftarrow L \backslash \{k\}$ and return to Step 1.

The convergence result and its proof for the foregoing enumerative algorithm closely follows that for the IEiCP as described in [6]. We detail this below for the sake of completeness.

Theorem 4.3 *The enumerative algorithm for NLP$_3$ run with $\epsilon = 0$ either terminates finitely with a solution to SOCEiCP, or else, an infinite branch-and-bound (B&B) tree is generated such that along any infinite branch of this tree, any accumulation point of the stationary points obtained for NLP$_3$ solves SOCEiCP.*

Proof The case of finite termination is obvious. Hence, suppose that an infinite B&B tree is generated, and consider any infinite branch. Denote, for simplicity, $\zeta \equiv (x, w, \lambda, y, z)$ and let $\{\zeta^s\}_S$, with $s \in S$, be a sequence of stationary points of NLP$_3$ that correspond to nodes on this infinite branch. Then, by taking a subsequence if necessary, we may assume

$$\{\zeta^s\}_S \to \zeta^* \text{ and } \{[c^s, d^s]\}_S \to [c^*, d^*],$$

where $[c^s, d^s]$ denotes the vector of bounds on x at node $s \in S$ of the B&B tree. We will show that ζ^* yields a solution to SOCEiCP.

Note that along the infinite branch under consideration, there exists some index-pair $(\hat{i}, \hat{j})$ such that we branch on the interval for $x^{\hat{i}}_{\hat{j}}$ infinitely often. Let this correspond to nodes indexed by $s \in S_1 \subseteq S$. By the partitioning rule (64), since the interval length for $x^{\hat{i}}_{\hat{j}}$ decreases by a geometric ratio of at most 0.9 over $s \in S_1$, we have in the limit that

$$c^{*\hat{i}}_{\hat{j}} = d^{*\hat{i}}_{\hat{j}} = x^{*\hat{i}}_{\hat{j}} = \nu^*, \text{ say.} \tag{65}$$

Furthermore, from (65) and the RLT bound-factor constraints (44)–(51), we have in the limit that

$$y^{*\hat{i}}_{\hat{j}} = \lambda^* \nu^* = \lambda^* x^{*\hat{i}}_{\hat{j}} \tag{66}$$

and

$$z^{*\hat{i}}_{\hat{j}} = \nu^* w^{*\hat{i}}_{\hat{j}} = x^{*\hat{i}}_{\hat{j}} w^{*\hat{i}}_{\hat{j}}. \tag{67}$$

Moreover, by the selection of the index-pair $(\hat{i}, \hat{j})$ for $s \in S_1$, via (63), we get in the limit as $s \to \infty$, $s \in S_1$, that

$$z^{*i}_j = x^{*i}_j w^{*i}_j \quad \text{and} \quad y^{*i}_j = \lambda^* x^{*i}_j, \quad j = 0, 1, \ldots, n_i - 1, \; i = 1, \ldots, r. \tag{68}$$

Consequently, the set of constraints (56) yield from (68) that, in the limit, $w^* - By^* + Ax^* = 0$, i.e.,

$$w^* = \lambda^* Bx^* - Ax^*. \tag{69}$$

Furthermore, by (54) and (68), we get

$${x^*}^T w^* = 0. \tag{70}$$

Likewise, from (57), (58), (14), (15), and (68), we get

$$x^* \in K \text{ and } w^* \in K. \tag{71}$$

Thus, (69)–(71) imply that the (x^*, w^*, λ^*)–part of ζ^* represents a solution to SOCEiCP.
■

Remark 2 Note that by the proof of Theorem 4.3, only one set from each pair {(44),(45)}, {(46),(47)} and {(48),(49)}, {(50),(51)} of the RLT bound-factor constraints is necessary to assure convergence of the proposed algorithm to a solution to SOCEiCP. However, we retain both pairs of constraints from each set since they better guide the algorithm to converge more efficiently, similar to our experience in [6].

5. Computation of lower and upper bounds for the variables

The enumerative algorithm described in the previous section requires lower and upper bounds for the original variables of the nonlinear programming formulation of the SOCEiCP that have a major importance for the construction of the so-called RLT constraints. In this section we introduce a few procedures for the computation of such bounds. Note that these techniques should be used as a preprocessing for the enumerative algorithm.

5.1. *Lower and upper bounds* $[c, d]$ *for the* x*–variables*

It is easy to see that any feasible vector x for the formulation NLP$_2$ belongs to the set

$$\Delta = \{x \in \mathbb{R}^n \colon \sum_{i=1}^{r} x_0^i = 1,\ x_0^i \geq 0,\ -1 \leq x_j^i \leq 1,\ j = 1, \ldots, n_i - 1,\ i = 1, \ldots, r\}. \tag{72}$$

Accordingly, lower and upper bounds $[c_j^i, d_j^i]$ for the variables x_j^i are given as follows:

$$c_0^i = 0,\ d_0^i = 1, \quad i = 1, \ldots, r \tag{73}$$

$$c_j^i = -1,\ d_j^i = 1, \quad j = 1, \ldots, n_i - 1,\ i = 1, \ldots, r. \tag{74}$$

5.2. *Upper bound* u *for the variable* λ

If (x, w, λ) is a solution to SOCEiCP, then $x \neq 0$ and $0 = x^T w = \lambda x^T Bx - x^T Ax$. Since $B \in \text{PD}$, then $x^T Bx \neq 0$ and

$$\lambda = \frac{x^T Ax}{x^T Bx}. \tag{75}$$

Therefore, an upper bound u for λ can be found by letting

$$u \geq \max_{x \in \Delta} \frac{x^T A x}{x^T B x}.$$

But for any $x \in \Delta$,

$$x^T A x \leq |x^T A x| \leq \sum_{i=1}^{n} \sum_{j=1}^{n} |a_{ij}||x_i||x_j| \leq \sum_{i=1}^{n} \sum_{j=1}^{n} |a_{ij}| \equiv \mu, \text{ say.} \tag{76}$$

Then an upper bound u is given by

$$u = \mu/\eta, \tag{77}$$

where

$$\eta \equiv \min \left\{ \frac{1}{2} x^T (B + B^T) x \colon x \in \Delta \right\}. \tag{78}$$

Hence, computing the upper bound u essentially requires the solution of a strictly convex quadratic program with simple constraints.

5.3. *Lower bound l for the variable λ*

Consider the following linear program (LP):

$$\begin{aligned} \textbf{LP:} \text{ Minimize } & \sum_{i=1}^{r} y_0^i \\ \text{subject to } & w = By - Ax \\ & x \in \Delta \\ & L \leq w \leq U, \end{aligned}$$

where L and U are, respectively, some finite lower and upper bounds on the $w-$variables (these are derived in Section 5.4 below based on the above upper bound u on λ). Then the following result holds true:

Proposition 5.1 *Problem LP has an optimal solution.*

Proof Since there is at least an $\bar{x} \in \Delta$ and a $\bar{w}$ such that $L \leq \bar{w} \leq U$, and since $B \in$ PD, we have that a feasible solution for LP is given by $(\bar{x}, \bar{w}, \bar{y})$, where $\bar{y}$ is the unique solution to the linear system $By = \bar{w} + A\bar{x}$. Now, suppose that there exists a nonzero recession direction (d_x, d_y, d_w) for the feasible region of LP. Due to the definition of Δ and since $L \leq w \leq U$, we have $d_x = d_w = 0$. Hence, $d_y \neq 0$ satisfies $Bd_y = 0$, which is a contradiction since $B \in$ PD. Thus the feasible region of LP is nonempty and bounded, and hence LP has an optimal solution. ∎

Therefore, noting that LP is based on a relaxation of the constraints of NLP$_2$, and that $\sum_{i=1}^{r} y_0^i = \lambda$, we can take the optimal objective value of LP as a lower bound l for λ.

5.4. *Lower and Upper bounds $[L, U]$ for the w–variables*

Note that $w_0^i \geq 0 \equiv L_0^i,\ \ i = 1, \ldots, r$. On the other hand, using the equations

$$w = \lambda Bx - Ax,$$

we have

$$w_0^i = \sum_{j=1}^{n} (\lambda b_{t_i j} - a_{t_i j}) x_j,\ \ i = 1, \ldots, r,$$

where $t_1 = 1$ and $t_i = 1 + \sum_{k=1}^{i-1} n_k$, $i = 2, \ldots, r$. Note that t_i is the index for the rows of the matrices A and B that correspond to the variable w_0^i, $i = 1, \ldots, r$. Since $x \in \Delta$ and $\lambda \leq u$, where $u > 0$ by (76)–(78), then

$$U_0^i \equiv \sum_{j=1}^{n} u|b_{t_i j}| + |a_{t_i j}|, \quad i = 1, \ldots, r,$$

yield valid upper bounds for the variables $w_0^i, i = 1, \ldots, r$.
Since

$$\|\bar{w}^i\| \leq w_0^i, \quad i = 1, \ldots, r,$$

then

$$L_j^i \equiv -U_0^i \leq w_j^i \leq U_0^i \equiv U_j^i,\ \ j = 1, \ldots, n_i - 1,\ \ i = 1, \ldots, r,$$

yield valid lower and upper bounds for the remaining variables w_j^i with $j \neq 0$.

6. A hybrid algorithm for the SOCEiCP

6.1. *A semi-smooth algorithm*

In this section, we start by discussing a semi-smooth algorithm introduced in [1] for the SOCEiCP based on the so-called natural residual function. We decided to choose this function due to its simplicity and generality. Consider again the SOCEiCP (2), where x and w belong to the cone K given by (3)-(4). As noted in (52), the following equivalence holds:

$$x \in K,\ w \in K,\ x^T w = 0 \quad \Longleftrightarrow \quad x^i \in K_i,\ w^i \in K_i,\ (x^i)^T w^i = 0, \quad i = 1, \ldots, r. \tag{79}$$

According to [1], SOCEiCP can be reformulated as the following system of equations

$$\Phi(z) = \Phi(x, w, \lambda) := \begin{bmatrix} \varphi^1(x^1, w^1) \\ \vdots \\ \varphi^r(x^r, w^r) \\ \lambda Bx - Ax - w \\ \sum_{i=1}^{r} (e^i)^T x^i - 1 \end{bmatrix} = 0. \tag{80}$$

where $\varphi^i \colon \mathbb{R}^{n_i} \times \mathbb{R}^{n_i} \to \mathbb{R}^{n_i}$ satisfies

$$\varphi^i(x^i, w^i) = 0 \quad \Longleftrightarrow \quad x^i \in K_i,\ w^i \in K_i,\ (x^i)^T w^i = 0 \tag{81}$$

for each $i = 1, \ldots, r$.

The steps of the semi-smooth algorithm for the solution of the system (80) are as follows:

Semi-Smooth Algorithm

Step 0 (Initialization) - Let $\bar{z} = (\bar{x}, \bar{w}, \bar{\lambda})$ be an initial point such that $\bar{x} \in \Delta$, and let ϵ_1 and ϵ_2 be selected positive tolerances.

Step 1 (Newton direction) - Compute the search direction $d = (d_x, d_w, d_\lambda)$ by

$$J(\bar{x}, \bar{w}, \bar{\lambda}) \begin{bmatrix} d_x \\ d_w \\ d_\lambda \end{bmatrix} = -\Phi(\bar{x}, \bar{w}, \bar{\lambda}), \tag{82}$$

where $J(\bar{x}, \bar{w}, \bar{\lambda})$ is the Clarke generalized Jacobian of Φ at $\bar{z}$ [8]. If the matrix $J(\bar{z})$ is singular, stop with an unsuccessful termination.

Step 2 (Update) - Find a new point $\tilde{z} = (\tilde{x}, \tilde{w}, \tilde{\lambda})$ by

$$\tilde{x} = \bar{x} + d_x, \quad \tilde{w} = \bar{w} + d_w, \quad \tilde{\lambda} = \bar{\lambda} + d_\lambda$$

and let $\bar{x} = \tilde{x}$, $\bar{w} = \tilde{w}$, and $\bar{\lambda} = \tilde{\lambda}$. If

$$\|\bar{w} - \bar{\lambda} B\bar{x} + A\bar{x}\| < \epsilon_1 \text{ and } \|\Phi(\bar{x}, \bar{w}, \bar{\lambda})\| < \epsilon_2$$

holds, then stop with a solution $(\bar{x}, \bar{w}, \bar{\lambda})$ of SOCEiCP. Otherwise, go to Step 1.

It follows from the description of the steps of the algorithm that all the iterates $\bar{x}$ satisfy the normalization constraint (12). Furthermore, the main work per iteration is the computation of the generalized Jacobian $J = J(\bar{x}, \bar{w}, \bar{\lambda})$ and the solution of a linear system of equations (82) with J. This matrix should be nonsingular and there is no guarantee that this property holds in each iteration. As discussed later, this is one of the drawbacks of this algorithm. On the positive side, it is possible to have explicit simple formulas for the generalized Jacobian when the so-called natural residual function is used. Next we discuss these formulas.

Consider the natural residual function φ^i_{NR} associated with the second-order cone K_i, which is defined as in [1] by

$$\varphi^i_{\mathrm{NR}}(x^i, w^i) := x^i - P_{K_i}(x^i - w^i),$$

where $P_{K_i}(s^i)$ is the projection of a vector $s^i = (s^i_0, \bar{s}^i) \in \mathbb{R} \times \mathbb{R}^{n_i - 1}$ onto the second-order cone K_i for each $i = 1, \ldots, r$, i.e.,

$$P_{K_i}(s^i) = \arg\min_{x^i \in K_i} \|x^i - s^i\|.$$

As shown in [12], each one of the residual functions φ^i_{NR}, $i = 1, \ldots, r$, satisfies (81).

Furthermore, the generalized Jacobian of Φ at $z = (x, w, \lambda)$ is given by

$$J = \left[\begin{array}{ccccccc} I_{n_1} - V^1 & 0 & 0 & V^1 & 0 & 0 & \\ 0 & \ddots & 0 & 0 & \ddots & 0 & 0 \\ 0 & 0 & I_{n_r} - V^r & 0 & 0 & V^r & \\ & & & & & & \\ \multicolumn{3}{c}{\lambda B - A} & \multicolumn{3}{c}{-I_n} & Bx \\ \multicolumn{3}{c}{(e^1)^T \dots (e^r)^T} & \multicolumn{3}{c}{0} & 0 \end{array}\right], \tag{83}$$

where I_{n_i} denotes the identity matrix of order n_i and the matrices V^i are computed as described below.

Next, we discuss explicit formulas for the matrices V^i, $i = 1, \dots, r$. Since all these matrices have similar forms, we only consider the case of $r = 1$, i.e.,

$$K = \{(x_0, \bar{x}) \in \mathbb{R} \times \mathbb{R}^{n-1} \colon \|\bar{x}\| \le x_0\}.$$

Clearly, the mapping P_K is Lipschitz continuous and hence has a subdifferential everywhere. Moreover, it can be shown [14] that the projection mapping P_K has a property called semi-smoothness which is useful in establishing superlinear convergence of a generalized Newton method [9, 26]. Explicit formulas of the B-subdifferential $\partial_B P_K$ and the Clarke subdifferential ∂P_K are, for example, available in [14, 18, 23]. In the following, we list representations of the B-subdifferential $\partial_B P_K(y)$, which depend on the point $y = (y_0, \bar{y})$ under consideration. Let $\lambda_i, i = 1, 2$, denote the spectral values of y [14, 18]. Note that P_K is in fact continuously differentiable at y in cases (a)-(c).

(a) If $0 < \lambda_1 \le \lambda_2$, i.e., $y_0 > \|\bar{y}\|$, then

$$\partial_B P_K(y) = \{I_n\}\,.$$

(b) If $\lambda_1 \le \lambda_2 < 0$, i.e., $y_0 < -\|\bar{y}\|$, then

$$\partial_B P_K(y) = \{0\}\,.$$

(c) If $\lambda_1 < 0 < \lambda_2$, i.e., $-\|\bar{y}\| < y_0 < \|\bar{y}\|$, then

$$\partial_B P_K(y) = \left\{ \frac{1}{2} \begin{pmatrix} 1 & \bar{v}^T \\ \bar{v} & H \end{pmatrix} \right\},$$

where $\bar{v} = \frac{\bar{y}}{\|\bar{y}\|}$ and $H = \left(1 + \frac{y_0}{\|\bar{y}\|}\right) I_{n-1} - \frac{y_0}{\|\bar{y}\|} \bar{v}\bar{v}^T$.

(d) If $\lambda_1 = 0 < \lambda_2$, i.e., $y_0 = \|\bar{y}\| \neq 0$, then

$$\partial_B P_K(y) = \left\{ I_n, \frac{1}{2} \begin{pmatrix} 1 & \bar{v}^T \\ \bar{v} & H \end{pmatrix} \right\},$$

where $\bar{v} = \frac{\bar{y}}{\|\bar{y}\|}$ and $H = 2I_{n-1} - \bar{v}\bar{v}^T$. In practice, we choose I_n from $\partial_B P_K(y)$.

(e) If $\lambda_1 < 0 = \lambda_2$, i.e., $y_0 = -\|\bar{y}\| \neq 0$, then

$$\partial_B P_K(y) = \left\{ 0, \frac{1}{2} \begin{pmatrix} 1 & \bar{v}^T \\ \bar{v} & H \end{pmatrix} \right\},$$

where $\bar{v} = \frac{\bar{y}}{\|\bar{y}\|}$ and $H = \bar{v}\bar{v}^T$. In practice, we choose 0 from $\partial_B P_K(y)$.

(f) If $\lambda_1 = \lambda_2 = 0$, i.e., $y_0 = \|\bar{y}\| = 0$, then

$$\partial_B P_K(y) = \{I_n, 0\} \cup \left\{ \frac{1}{2} \begin{pmatrix} 1 & \bar{v}^T \\ \bar{v} & H \end{pmatrix} \;\middle|\; \begin{array}{l} \bar{v} \in \mathbb{R}^{n-1} \text{ such that } \|\bar{v}\| = 1, \\ H = (1+\rho)I_{n-1} - \rho\bar{v}\bar{v}^T \\ \text{with } \rho \in \mathbb{R} \text{ such that } |\rho| \le 1 \end{array} \right\}.$$

In practice, we choose 0 from $\partial_B P_K(y)$ in this last case.

Consider now the system of nonsmooth equations with $r = 1$:

$$\Phi(z) := \begin{pmatrix} \varphi_{\mathrm{NR}}(x, w) \\ \lambda Bx - Ax - w \\ e^T x - 1 \end{pmatrix} = 0,$$

where $z = (x, w, \lambda)$ and $\varphi_{\mathrm{NR}}(x, w) = x - P_K(x - w)$. Then a generalized Jacobian of the function Φ with respect to the variable z is given by

$$J = \begin{bmatrix} I_n - V & V & 0 \\ \lambda B - A & -I_n & Bx \\ e^T & 0 & 0 \end{bmatrix},$$

where V is a generalized Jacobian of P_K at $y = x - w$, as given by one of the matrices $\partial_B P_K(y)$ specified above.

As discussed in [1], there is no guarantee that the semi-smooth algorithm will converge to a solution of the SOCEiCP. Actually, the algorithm may fail to attain a solution due to lack of convergence or because the generalized Jacobian is singular at a particular iteration. On the positive side, whenever the algorithm converges, then it usually provides an accurate solution in a relatively small number of iterations. An attractive characteristic of this semi-smooth method is its ability to start with any point. As in [11], we propose a hybrid algorithm that combines the good features of the enumerative and semi-smooth algorithms. The simple enumerative algorithm is applied first. If the value of the merit function of the nonlinear formulation is relatively small at a current point $(\bar{x}, \bar{w}, \bar{\lambda}, \bar{y}, \bar{z})$, the procedure switches to the semi-smooth method with the initial point $(\bar{x}, \bar{w}, \bar{\lambda})$. Then, either this latter algorithm terminates successfully with a solution of the SOCEiCP or it fails, in which case we revert to the enumerative algorithm by continuing with the previously aborted stage in the algorithmic process. As in [11], the switch from the enumerative method to the semi-smooth method is done by using a relaxed set of tolerances in the stopping criterion of the first algorithm. The steps of the proposed hybrid algorithm are presented below.

Hybrid Algorithm for finding a complementary eigenvalue.

Step 0 - Let $\bar{\epsilon}$ a positive tolerance for switching from the enumerative algorithm to the semi-smooth method, and let *nmaxit* be the maximum number of iterations permitted to be performed by the semi-smooth method.

Step 1 - Apply Step 1 of the enumerative method with a positive tolerance $\epsilon < \bar{\epsilon}$. Let $(\bar{x}, \bar{y}, \bar{w}, \bar{\lambda})$ be the stationary point associated with the node k, and compute ψ as in Step 2 of the enumerative method.

(i) If $\psi < \epsilon$, stop with a solution of the SOCEiCP.

(ii) If $\psi < \bar{\epsilon}$, go to Step 2.

(iii) Generate two new nodes as discussed in the enumerative method. Repeat Step 1.

Step 2 - Apply the semi-smooth method. If the algorithm terminates with a solution (x^*, w^*, λ^*) of SOCEiCP, stop. Otherwise the semi-smooth method terminates without success (singular generalized Jacobian or number of iterations reaches *nmaxit*); go to Step 1 (iii) with node k and the solution $(\bar{x}, \bar{y}, \bar{w}, \bar{\lambda})$ given at the beginning of this step.

7. Computational Experience

In this section, we report some computational experience with the algorithms discussed in the previous sections. All the tests have been performed on a Pentium IV (Intel) with Hyperthreading, 3.0 GHz CPU, 2GB RAM computer, using the operating system Linux. The algorithms were implemented in the General Algebraic Modeling System (GAMS) language (Rev 118 Linux/Intel) [5]. Four sets of test problems were constructed for the SOCEiCP. First, matrices E and F were randomly generated with elements uniformly distributed in the intervals $[0, 1]$ and $[-1, 1]$, respectively. In the first set of test problems $A = E$ and $B = F + D$, where D is a diagonal matrix with positive diagonal elements such that B is an asymmetric strictly row diagonally dominant and PD matrix. These problems are denoted by RNB(k, m, n), where k and m are the end-points of the chosen interval for generating the matrix elements, and n represents the order of the matrices. The second set of test problems, denoted by RNI(k, m, n), differs from the first set in the matrix B, which was set equal to the identity. In the third set of test problems, the matrices A and B are given by $E^T E$, and $F^T F$, respectively, (B is symmetric PD). These problems are denoted by RSB(k, m, n), and where k, m, and n have the same meaning of the previous sets of test problems. In the last set of test problems, we consider the matrix B as the identity matrix and $A = F^T F$. These problems are denoted by RSI(k, m, n). For all test problems, we consider dimensions: $n = 5, 10, 20, 30, 40$, and 50.

The numerical results of the experiments on the solution of all these test problems are reported in Tables 1 to 5. In these tables, we use the following notation:

- Val - value of the objective function;
- $\lambda-$ value of the eigenvalue computed by the algorithms ($--$ when the algorithms are not able to compute an eigenvalue);
- Nnod - total number of nodes generated;
- c - integer number such that $x^T w = \beta.10^{\mathrm{c}}$ for some $\beta \in (0.1, 1]$, where $(\bar{x}, \bar{\lambda}, \bar{w})$ is the best approximate global minimum computed by the algorithm;
- Fe - integer number such that $\|\bar{w} - \bar{\lambda} B\bar{x} - A\bar{x}\|_\infty = \beta.10^{\mathrm{Fe}}$ for some $\beta \in (0.1, 1]$, where $(\bar{x}, \bar{\lambda}, \bar{w})$ is the best approximate global minimum computed by the algorithm;
- Ntime - number of times that the Semi-Smooth Algorithm is called;
- Iter - total number of iterations for the Semi-Smooth Algorithm (with the number of iterations in each call displayed separately in the summation);
- $*-$ enumerative algorithm was not able to compute an eigenvalue within 300 nodes (in Table 3).
- $*-$ BARON was not able to compute an eigenvalue within, 1,000 seconds (in Table 4).

In the first set of experiments, we investigate whether a stationary point of NLP$_2$ gives a solution of the SOCEiCP. To do this, we alternatively used two well-known efficient codes MINOS [20] and IPOPT [33] to compute a stationary point for all the NLP$_2$ instances generated as discussed before. The numerical results of these experiments are displayed in Tables 1 and 2 and confirm our expectation that these codes are in some cases able to compute a solution for the SOCEiCP, but not in general. The code IPOPT performed better than MINOS for this purpose. Furthermore, both codes efficiently computed stationary points with small objective function values. This is a very important feature in the design of the enumerative method (henceforth, we use IPOPT for computing stationary points in this method).

Table 1. Performance of MINOS for computing a stationary point of NLP_2.

	$r = 1$		$r = 2$		$r = 3$	
Problem	Val	λ	Val	λ	Val	λ
RNI(0, 1, 5)	0.000	−0.119	0.000	−0.097		
RNI(0, 1, 10)	0.000	0.823	0.000	0.255	0.001	−−
RNI(0, 1, 20)	0.196	−−	0.078	−−	0.054	−−
RNI(0, 1, 30)	0.000	1.010	0.000	1.058	0.020	−−
RNI(0, 1, 40)	0.000	0.961	0.111	−−	0.046	−−
RNI(0, 1, 50)	0.000	1.665	0.052	−−	0.072	−−
RNI(−1, 1, 5)	0.000	0.895	0.000	0.895		
RNI(−1, 1, 10)	0.000	0.944	0.002	−−	0.000	0.777
RNI(−1, 1, 20)	0.000	1.392	0.109	−−	0.001	−−
RNI(−1, 1, 30)	0.000	2.489	1.310	−−	0.031	−−
RNI(−1, 1, 40)	0.000	1.820	0.010	−−	0.018	−−
RNI(−1, 1, 50)	0.000	3.336	0.015	−−	0.110	−−
RSI(0, 1, 5)	0.000	0.604	0.000	0.075		
RSI(0, 1, 10)	0.134	−−	0.000	0.823	0.001	−−
RSI(0, 1, 20)	0.133	−−	0.001	−−	0.033	−−
RSI(0, 1, 30)	0.075	−−	0.005	−−	0.015	−−
RSI(0, 1, 40)	0.257	−−	0.024	−−	0.052	−−
RSI(0, 1, 50)	0.288	−−	0.282	−−	0.355	−−
RSI(−1, 1, 5)	0.000	2.878	0.000	2.729		
RSI(−1, 1, 10)	0.616	−−	0.098	−−	0.001	−−
RSI(−1, 1, 20)	0.002	−−	0.010	−−	0.036	−−
RSI(−1, 1, 30)	1.652	−−	0.285	−−	0.078	−−
RSI(−1, 1, 40)	3.865	−−	5.570	−−	0.919	−−
RSI(−1, 1, 50)	35.643	−−	4.925	−−	14.788	−−
RNB(0, 1, 5)	0.001	−−	0.000	−0.037		
RNB(0, 1, 10)	0.001	−−	0.000	0.046	0.001	−−
RNB(0, 1, 20)	0.003	−−	0.001	−−	0.001	−−
RNB(0, 1, 30)	0.000	0.078	0.001	−−	0.001	−−
RNB(0, 1, 40)	0.001	−−	0.001	−−	0.001	−−
RNB(0, 1, 50)	0.000	0.068	0.001	−−	0.001	−−
RNB(−1, 1, 5)	0.000	0.338	0.000	0.354		
RNB(−1, 1, 10)	0.000	0.144	0.001	−−	0.001	−−
RNB(−1, 1, 20)	0.005	−−	0.002	−−	0.002	−−
RNB(−1, 1, 30)	0.001	−−	0.006	−−	0.001	−−
RNB(−1, 1, 40)	0.000	0.094	0.001	−−	0.001	−−
RNB(−1, 1, 50)	0.003	−−	0.001	−−	0.002	−−
RSB(0, 1, 5)	0.001	−−	0.000	0.012		
RSB(0, 1, 10)	0.000	0.082	0.001	−−	0.000	0.397
RSB(0, 1, 20)	0.000	0.038	0.000	0.003	0.000	0.021
RSB(0, 1, 30)	0.000	0.002	0.000	0.001	0.002	−−
RSB(0, 1, 40)	0.000	0.003	0.000	0.004	0.000	0.004
RSB(0, 1, 50)	0.000	0.002	0.000	0.002	0.000	0.002
RSB(−1, 1, 5)	0.001	−−	0.000	0.205		
RSB(−1, 1, 10)	0.001	−−	0.001	−−	0.000	0.035
RSB(−1, 1, 20)	0.000	0.004	0.001	−−	0.000	0.010
RSB(−1, 1, 30)	0.000	0.005	0.000	0.004	0.000	0.004
RSB(−1, 1, 40)	0.000	0.009	0.000	0.012	0.000	0.010
RSB(−1, 1, 50)	0.001	−−	0.000	0.007	0.000	0.041

The numerical results of the enumerative method implemented using IPOPT for computing the required stationary points are reported in Table 3. We have used the tolerance $\epsilon = 10^{-5}$ for the stopping criterion of the algorithm. The numerical results reported in Table 3 clearly indicate that the enumerative algorithm is able to efficiently compute a solution of SOCEiCP or at least an approximate global minimum of NLP_2 with a small objective function value (very close to zero). In fact, the algorithm usually enumerates a very small number of nodes, and often terminates at the root node itself.

Table 4 examines the performance of BARON [27] for solving NLP_2 corresponding to all the SOCEiCP test problems. The code implements a "branch-and-reduce" algorithm and aims at finding a solution of SOCEiCP by computing a global minimum of NLP_2. However, many of the globally optimal solutions reported by BARON are not true solutions of SOCEiCP, because the values of Fe are relatively large. In contrast, the enumerative method does not declare such (approximate) global minima as solutions of SOCEiCP. In fact, when the enumerative algorithm is only able to find an approximate global optimizer that is not declared as a solution of SOCEiCP, the values of Fe are similar to those delivered by the global minima given by BARON (see the results marked with * in Table 3 and compare the corresponding values of Fe with those obtained by

Table 2. Performance of IPOPT for computing a stationary point of NLP$_2$.

	$r = 1$		$r = 2$		$r = 3$	
Problem	Val	λ	Val	λ	Val	λ
RNI$(0,1,5)$	0.000	-0.119	0.000	-0.040		
RNI$(0,1,10)$	0.001	$--$	0.000	0.252	0.001	$--$
RNI$(0,1,20)$	0.000	0.507	0.001	$--$	0.011	$--$
RNI$(0,1,30)$	0.000	1.015	0.000	1.065	0.003	$--$
RNI$(0,1,40)$	0.000	0.961	0.001	$--$	0.000	1.100
RNI$(0,1,50)$	0.000	1.665	0.000	1.386	0.000	1.096
RNI$(-1,1,5)$	0.268	$--$	0.016	$--$		
RNI$(-1,1,10)$	0.000	0.944	0.000	0.583	0.000	0.809
RNI$(-1,1,20)$	0.470	$--$	0.004	$--$	0.001	$--$
RNI$(-1,1,30)$	0.000	2.490	1.310	$--$	0.031	$--$
RNI$(-1,1,40)$	0.000	1.821	0.010	$--$	0.000	2.694
RNI$(-1,1,50)$	2.419	$--$	0.000	2.279	0.074	$--$
RSI$(0,1,5)$	0.001	$--$	0.000	0.160		
RSI$(0,1,10)$	0.134	$--$	0.000	0.823	0.000	0.623
RSI$(0,1,20)$	0.191	$--$	0.001	$--$	0.033	$--$
RSI$(0,1,30)$	0.075	$--$	0.003	$--$	0.007	$--$
RSI$(0,1,40)$	0.257	$--$	0.024	$--$	0.043	$--$
RSI$(0,1,50)$	0.288	$--$	0.489	$--$	0.409	$--$
RSI$(-1,1,5)$	0.160	$--$	0.000	2.728		
RSI$(-1,1,10)$	0.616	$--$	0.096	$--$	0.000	1.258
RSI$(-1,1,20)$	0.507	$--$	0.448	$--$	0.003	$--$
RSI$(-1,1,30)$	1.652	$--$	0.285	$--$	0.058	$--$
RSI$(-1,1,40)$	3.865	$--$	4.660	$--$	0.293	$--$
RSI$(-1,1,50)$	18.902	$--$	3.461	$--$	0.495	$--$
RNB$(0,1,5)$	0.001	$--$	0.000	-0.015		
RNB$(0,1,10)$	0.000	0.278	0.000	0.046	0.001	$--$
RNB$(0,1,20)$	0.002	$--$	0.001	$--$	0.000	0.096
RNB$(0,1,30)$	0.000	0.078	0.000	0.079	0.001	$--$
RNB$(0,1,40)$	0.001	$--$	0.001	$--$	0.001	$--$
RNB$(0,1,50)$	0.000	0.068	0.000	0.057	0.001	$--$
RNB$(-1,1,5)$	0.022	$--$	0.002	$--$		
RNB$(-1,1,10)$	0.000	0.144	0.000	0.172	0.001	$--$
RNB$(-1,1,20)$	0.005	$--$	0.001	$--$	0.002	$--$
RNB$(-1,1,30)$	0.001	$--$	0.006	$--$	0.001	$--$
RNB$(-1,1,40)$	0.000	0.094	0.001	$--$	0.000	0.124
RNB$(-1,1,50)$	0.000	0.136	0.000	0.072	0.001	$--$
RSB$(0,1,5)$	0.000	0.075	0.000	0.012		
RSB$(0,1,10)$	0.000	0.089	0.000	0.060	0.000	0.010
RSB$(0,1,20)$	0.000	0.006	0.000	0.032	0.000	0.035
RSB$(0,1,30)$	0.000	0.002	0.000	0.001	0.000	0.002
RSB$(0,1,40)$	0.000	0.003	0.000	0.004	0.000	0.004
RSB$(0,1,50)$	0.000	0.001	0.000	0.002	0.000	0.002
RSB$(-1,1,5)$	0.000	0.206	0.000	0.314		
RSB$(-1,1,10)$	0.001	$--$	0.000	0.031	0.000	0.035
RSB$(-1,1,20)$	0.000	0.004	0.000	0.006	0.000	0.009
RSB$(-1,1,30)$	0.000	0.005	0.000	0.004	0.001	$--$
RSB$(-1,1,40)$	0.000	0.009	0.000	0.012	0.000	0.010
RSB$(-1,1,50)$	0.001	$--$	0.000	0.007	0.000	0.048

BARON). Furthermore, as for the enumerative method, there are some instances where BARON terminates without declaring that the best computed feasible solution is a global minimizer of NLP$_2$. In one instance BARON was not even able to find a feasible solution for NLP$_2$ (see problem RSB$(0, 1, 50)$ with $r = 3$). Finally, as discussed below, the enumerative method can be beneficially combined with a semi-smooth method as proposed in the hybrid algorithm discussed in Section 6.

To summarize the numerical study thus far, the enumerative method is in general very efficient in either finding a solution of the SOCEiCP (i.e., computing a global minimum of NLP$_2$) or at least determining an approximate global minimum of NLP$_2$ with a very small objective function value. In fact, this is what motivated us to investigate the hybrid method discussed in Section 6. The numerical results for the hybrid algorithm (with $\epsilon_1 = 10^{-4}$, $\epsilon_2 = 10^{-4}$, $\bar{\epsilon} = 10^{-1}$, and $nmaxit = 100$) are reported in Table 5 and clearly show the efficacy of this approach. In fact, the hybrid algorithm was able to solve all the instances with a reasonable small amount of effort. In many cases the algorithm terminated at the root node itself where the enumerative method found a stationary point with a small objective function value and switched to the semi-smooth method, which was then able to find a solution to the SOCEiCP. In other instances, the semi-smooth method was called more than once (see the columns for Ntime and Iter). However, the number of calls yet is small and the hybrid method was able to solve all the SOCEiCP instances by either terminating with the semi-smooth method, or resorting to the enumerative method itself.

Table 3. Performance of the enumerative algorithm for solving SOCEiCP.

	$r = 1$				$r = 2$				$r = 3$			
Problem	Nnod	λ	c	Fe	Nnod	λ	c	Fe	Nnod	λ	c	Fe
RNI$(0, 1, 5)$	*	−−	−3	−4	1	2.711	−5	−9				
RNI$(0, 1, 10)$	1	3.849	−6	−12	1	4.146	−5	−10	1	4.675	−5	−11
RNI$(0, 1, 20)$	1	7.169	−5	−10	1	1.868	−5	−12	1	8.672	−5	−11
RNI$(0, 1, 30)$	1	4.955	−5	−11	1	3.773	−5	−11	1	12.084	−5	−10
RNI$(0, 1, 40)$	2	13.160	−4	−7	1	14.343	−5	−12	1	15.262	−5	−12
RNI$(0, 1, 50)$	1	1.665	−5	−10	1	17.637	−5	−11	1	18.344	−4	−10
RNI$(-1, 1, 5)$	*	−−	−6	−3	1	0.895	−5	−10				
RNI$(-1, 1, 10)$	1	0.944	−6	−10	1	0.942	−5	−10	1	1.078	−5	−7
RNI$(-1, 1, 20)$	1	1.393	−5	−10	5	1.581	−5	−8	16	1.310	−5	−10
RNI$(-1, 1, 30)$	1	2.490	−5	−9	2	3.236	−5	−9	1	2.071	−5	−9
RNI$(-1, 1, 40)$	4	1.832	−5	−9	*	−−	−1	−4	1	2.695	−5	−10
RNI$(-1, 1, 50)$	1	3.336	−5	−11	1	2.293	−5	−10	1	2.420	−5	−10
RSI$(0, 1, 5)$	2	6.864	−4	−10	1	8.276	−5	−11				
RSI$(0, 1, 10)$	2	5.160	−4	−11	1	3.426	−5	−12	1	24.287	−5	−11
RSI$(0, 1, 20)$	1	28.458	−5	−13	53	3.297	−5	−10	1	18.943	−5	−11
RSI$(0, 1, 30)$	9	5.791	−5	−10	*	−−	−2	−3	1	189.966	−5	−12
RSI$(0, 1, 40)$	2	142.720	−4	−12	6	119.672	−5	−12	2	107.298	−4	−12
RSI$(0, 1, 50)$	3	10.921	−5	−11	*	−−	−2	−5	*	−−	−1	−4
RSI$(-1, 1, 5)$	1	4.727	−5	−11	1	3.538	−5	−10				
RSI$(-1, 1, 10)$	1	7.881	−6	−12	1	7.371	−5	−11	1	7.080	−5	−8
RSI$(-1, 1, 20)$	*	−−	−2	−4	6	14.188	−5	−11	1	18.619	−5	−12
RSI$(-1, 1, 30)$	*	−−	−2	−5	1	26.044	−5	−12	1	37.967	−5	−11
RSI$(-1, 1, 40)$	*	−−	0	−2	*	−−	−6	−2	*	−−	−2	−5
RSI$(-1, 1, 50)$	1	52.483	−4	−10	*	−−	−2	−6	5	45.497	−4	−8
RNB$(0, 1, 5)$	*	−−	−2	−3	1	0.494	−5	−10				
RNB$(0, 1, 10)$	1	0.196	−5	−10	1	0.046	−5	−8	1	0.537	−5	−9
RNB$(0, 1, 20)$	*	−−	−2	−2	9	0.061	−5	−9	*	−−	−2	−3
RNB$(0, 1, 30)$	1	0.079	−5	−7	1	0.079	−5	−7	*	−−	−4	−5
RNB$(0, 1, 40)$	1	0.050	−5	−6	9	0.205	−5	−7	1	0.053	−5	−9
RNB$(0, 1, 50)$	1	0.068	−5	−6	1	0.056	−4	−6	1	0.044	−5	−9
RNB$(-1, 1, 5)$	1	0.338	−6	−10	1	0.354	−5	−8				
RNB$(-1, 1, 10)$	1	0.144	−6	−10	1	0.171	−5	−8	*	−−	−3	−3
RNB$(-1, 1, 20)$	1	0.137	−5	−8	17	0.137	−5	−9	1	0.134	−5	−7
RNB$(-1, 1, 30)$	31	0.147	−5	−10	1	0.215	−5	−9	*	−−	−2	−2
RNB$(-1, 1, 40)$	1	0.094	−5	−7	2	0.129	−5	−6	1	0.124	−5	−7
RNB$(-1, 1, 50)$	1	0.136	−5	−6	3	0.094	−5	−7	3	0.105	−5	−8
RSB$(0, 1, 5)$	*	−−	−3	−2	2	0.006	−5	−8				
RSB$(0, 1, 10)$	1	0.082	−5	−9	5	0.052	−5	−6	1	0.015	−5	−5
RSB$(0, 1, 20)$	*	−−	−4	−2	11	0.109	−5	−7	7	0.050	−5	−6
RSB$(0, 1, 30)$	*	−−	−6	−1	1	0.282	−5	−7	21	0.028	−5	−5
RSB$(0, 1, 40)$	*	−−	−4	−1	*	−−	−4	−1	*	−−	−4	−1
RSB$(0, 1, 50)$	1	0.233	−5	−8	100	0.160	−5	−7	*	−−	−5	−1
RSB$(-1, 1, 5)$	1	0.205	−6	−8	1	0.205	−5	−9				
RSB$(-1, 1, 10)$	*	−−	−6	−1	*	−−	−3	−2	1	0.035	−5	−7
RSB$(-1, 1, 20)$	*	−−	−6	−2	*	−−	−4	−3	*	−−	−4	−1
RSB$(-1, 1, 30)$	*	−−	−4	−1	*	−−	−4	−1	2	0.178	−4	−7
RSB$(-1, 1, 40)$	*	−−	−6	−1	*	−−	−4	−2	*	−−	−3	−1
RSB$(-1, 1, 50)$	*	−−	−4	0	*	−−	−4	−1	44	0.077	−5	−4

8. Conclusions

In this paper we have investigated the Second-Order Cone Eigenvalue Complementarity Problem (SOCEiCP). We showed that the SOCEiCP reduces to a Variational Inequality Problem on a compact and convex set. This reduction guarantees a solution to the SOCEiCP. Furthermore, the symmetric SOCEiCP can be solved by computing a stationary point on a convex set defined by linear and nonlinear constraints. For the asymmetric case, a nonlinear programming (NLP) formulation for the SOCEiCP was introduced. We demonstrated that a stationary point for this NLP often provides a solution to the SOCEiCP, but not always. We therefore proposed an enumerative algorithm for solving the asymmetric SOCEiCP, which aims at finding a global minimum for the NLP formulation. The method was proven to globally converge to a solution of the SOCEiCP, and was shown to perform well in practice. However, for some instances, the enumerative algorithm was only able to find an approximate global minimum of NLP. A hybrid algorithm was thus conceived that combines the enumerative method with a semi-smooth Newton's method, which thereby significantly enhanced the enumerative method. The reported numerical results exhibit that the hybrid algorithm indeed performs very well in practice, and is robust and efficient in comparison with using the commercial software BARON to solve the NLP formulation. The solution of other Eigenvalue Complementarity Problems arising in different applications is an interesting topic for future research.

Table 4. Performance of BARON for solving SOCEiCP.

	$r=1$				$r=2$				$r=3$			
Problem	Nnod	λ	c	Fe	Nnod	λ	c	Fe	Nnod	λ	c	Fe
RNI$(0,1,5)$	1	2.388	-6	-7	0	-0.044	-6	-7				
RNI$(0,1,10)$	0	1.359	-6	-7	0	0.253	-5	-6	10	0.116	-6	-7
RNI$(0,1,20)$	0	0.508	-5	-7	505	7.783	-6	-8	1	0.998	-3	-5
RNI$(0,1,30)$	0	1.012	-5	-7	0	1.061	-6	-8	28	12.084	-7	-9
RNI$(0,1,40)$	1	0.961	-6	-8	28	14.343	-6	-7	0	1.103	-5	-7
RNI$(0,1,50)$	10	1.664	-5	-7	*	$--$	-1	-2	*	$--$	-2	-2
RNI$(-1,1,5)$	0	0.895	-6	-7	0	0.895	-10	-9				
RNI$(-1,1,10)$	10	0.945	-6	-7	0	0.587	-5	-6	37	1.070	-5	-6
RNI$(-1,1,20)$	0	1.391	-5	-7	82	1.570	-5	-7	370	1.310	-5	-8
RNI$(-1,1,30)$	0	2.489	-6	-7	10	3.233	-6	-8	*	$--$	-2	-2
RNI$(-1,1,40)$	0	1.819	-6	-7	*	$--$	-2	-2	0	2.691	-6	-8
RNI$(-1,1,50)$	0	3.336	-6	-8	17	2.295	-4	-7	*	$--$	-1	-2
RSI$(0,1,5)$	0	2.437	-6	-7	1	0.152	-7	-8				
RSI$(0,1,10)$	0	5.160	-7	-7	0	0.823	-5	-6	1	24.287	-6	-8
RSI$(0,1,20)$	496	75.604	-6	-8	*	$--$	-2	-2	154	4.069	-6	-8
RSI$(0,1,30)$	307	5.870	-5	-7	226	183.581	-6	-9	*	$--$	-2	-2
RSI$(0,1,40)$	*	$--$	-1	-2	*	$--$	-2	-3	*	$--$	-1	-2
RSI$(0,1,50)$	*	$--$	-6	-1	*	$--$	-1	-1	*	$--$	-5	-1
RSI$(-1,1,5)$	0	2.878	-5	-8	0	2.728	-6	-7				
RSI$(-1,1,10)$	271	7.881	-6	-8	*	$--$	-2	-3	0	1.302	-6	-6
RSI$(-1,1,20)$	442	17.747	-6	-8	199	12.679	-5	-7	28	18.619	-5	-9
RSI$(-1,1,30)$	*	$--$	-2	-4	*	$--$	-1	-1	*	$--$	-1	-2
RSI$(-1,1,40)$	*	$--$	-6	-1	33	32.344	-4	-6	*	$--$	-6	-1
RSI$(-1,1,50)$	*	$--$	-6	0	*	$--$	-6	-1	*	$--$	-5	0
RNB$(0,1,5)$	28	0.453	-6	-7	0	-0.009	-6	-5				
RNB$(0,1,10)$	0	0.278	-5	-6	0	0.206	-6	-5	0	0.023	-5	-5
RNB$(0,1,20)$	0	0.057	-5	-5	*	$--$	-3	-2	*	$--$	-3	-2
RNB$(0,1,30)$	19	0.078	-5	-5	68	0.079	-5	-5	*	$--$	-3	-1
RNB$(0,1,40)$	*	$--$	-3	-1	*	$--$	-3	-1	*	$--$	-3	-1
RNB$(0,1,50)$	10	0.068	-5	-4	*	$--$	-3	-1	*	$--$	-3	-1
RNB$(-1,1,5)$	0	0.338	-7	-6	0	0.354	-8	-7				
RNB$(-1,1,10)$	1	0.144	-6	-6	703	0.169	-6	-6	118	0.191	-6	-6
RNB$(-1,1,20)$	1	0.137	-5	-6	622	0.136	-5	-6	172	0.134	-6	-5
RNB$(-1,1,30)$	1	0.147	-5	-4	0	0.215	-5	-5	*	$--$	-3	-1
RNB$(-1,1,40)$	20	0.094	-5	-5	*	$--$	-3	-1	*	$--$	-3	-1
RNB$(-1,1,50)$	11	0.136	-5	-4	*	$--$	-3	-1	*	$--$	-3	-1
RSB$(0,1,5)$	0	0.075	-6	-6	10	0.012	-6	-5				
RSB$(0,1,10)$	28	0.082	-7	-4	1	0.052	-7	-5	1	0.015	-7	-4
RSB$(0,1,20)$	287	0.006	-5	-1	109	0.002	-6	-1	1	0.010	-4	-1
RSB$(0,1,30)$	63	0.002	-6	-1	82	0.001	-7	-2	1	0.003	-5	-1
RSB$(0,1,40)$	1	0.003	-8	-1	1	0.004	-6	-1	9	0.004	-6	-1
RSB$(0,1,50)$	1	0.003	-6	-1	1	0.002	-6	-1	*	$--$	∞	∞
RSB$(-1,1,5)$	1	0.206	-7	-7	0	0.205	-6	-6				
RSB$(-1,1,10)$	1	0.183	-5	-4	3493	0.261	-5	-5	1	0.033	-8	-6
RSB$(-1,1,20)$	1627	0.022	-4	-1	1011	0.022	-4	-1	1	0.010	-4	-1
RSB$(-1,1,30)$	82	0.005	-6	-1	1	0.005	-5	-1	1	0.003	-6	-1
RSB$(-1,1,40)$	1	0.009	-5	-1	23	0.012	-6	0	1	0.010	-6	-1
RSB$(-1,1,50)$	*	$--$	-6	0	1	0.007	-6	-1	1	0.018	-6	0

Table 5. Performance of the hybrid algorithm for solving SOCEiCP.

	$r=1$				$r=2$				$r=3$			
Problem	Nnod	λ	Ntime	Iter	Nnod	λ	Ntime	Iter	Nnod	λ	Ntime	Iter
RNI$(0,1,5)$	3	2.388	3	$2\times100+98$	1	2.711	0	0				
RNI$(0,1,10)$	1	3.849	0	0	1	4.146	0	0	1	4.675	0	0
RNI$(0,1,20)$	1	7.169	0	0	1	1.868	0	0	1	8.672	0	0
RNI$(0,1,30)$	1	4.955	0	0	1	3.773	0	0	1	12.084	0	0
RNI$(0,1,40)$	1	13.160	1	1	1	14.343	0	0	1	15.262	0	0
RNI$(0,1,50)$	1	1.665	0	0	1	17.637	0	0	1	18.344	0	0
RNI$(-1,1,5)$	1	0.895	1	4	1	0.895	0	0				
RNI$(-1,1,10)$	1	0.944	0	0	1	0.942	0	0	1	1.078	0	0
RNI$(-1,1,20)$	1	1.393	0	0	1	1.578	1	16	15	1.310	15	$14\times100+16$
RNI$(-1,1,30)$	1	2.490	0	0	1	3.233	1	2	1	2.071	0	0
RNI$(-1,1,40)$	1	1.821	1	4	1	2.609	1	19	1	2.695	0	0
RNI$(-1,1,50)$	1	3.336	0	0	1	2.293	0	0	1	2.420	0	0
RSI$(0,1,5)$	1	6.864	1	1	1	8.276	0	0				
RSI$(0,1,10)$	1	5.160	1	1	1	3.426	0	0	1	24.287	0	0
RSI$(0,1,20)$	1	28.458	0	0	3	3.297	3	$2\times100+14$	1	18.943	0	0
RSI$(0,1,30)$	1	5.791	1	30	1	6.906	1	91	1	189.966	0	0
RSI$(0,1,40)$	2	142.720	1	100	1	9.797	1	82	1	313.323	1	96
RSI$(0,1,50)$	3	10.921	2	2×100	1	11.710	1	33	1	12.342	1	49
RSI$(-1,1,5)$	1	4.727	0	0	1	3.538	0	0				
RSI$(-1,1,10)$	1	7.881	0	0	1	7.371	0	0	1	7.080	0	0
RSI$(-1,1,20)$	9	17.747	9	$8\times100+94$	1	10.499	1	29	1	18.619	0	0
RSI$(-1,1,30)$	6	34.079	6	$5\times100+42$	1	26.044	0	0	1	37.967	0	0
RSI$(-1,1,40)$	26	35.461	24	$23\times100+17$	21	34.596	4	$3\times100+19$	14	34.678	14	$13\times100+54$
RSI$(-1,1,50)$	1	52.483	0	0	11	45.841	7	$6\times100+10$	5	45.497	0	0
RNB$(0,1,5)$	8	0.453	8	$7\times100+87$	1	0.494	0	0				
RNB$(0,1,10)$	1	0.196	0	0	1	0.046	0	0	1	0.537	0	0
RNB$(0,1,20)$	1	0.057	1	16	1	0.147	1	6	4	0.487	4	$3\times100+81$
RNB$(0,1,30)$	1	0.079	0	0	1	0.079	0	0	1	0.058	1	35
RNB$(0,1,40)$	1	0.050	0	0	1	0.037	1	13	1	0.053	0	0
RNB$(0,1,50)$	1	0.068	0	0	1	0.056	0	0	1	0.044	0	0
RNB$(-1,1,5)$	1	0.338	0	0	1	0.354	0	0				
RNB$(-1,1,10)$	1	0.144	0	0	1	0.171	0	0	10	0.191	10	$9\times100+23$
RNB$(-1,1,20)$	1	0.137	0	0	2	0.137	2	$100+47$	1	0.134	0	0
RNB$(-1,1,30)$	2	0.147	2	$100+37$	1	0.215	0	0	1	0.136	1	6
RNB$(-1,1,40)$	1	0.094	0	0	1	0.129	1	17	1	0.124	0	0
RNB$(-1,1,50)$	1	0.136	0	0	1	0.094	1	21	1	0.105	1	6
RSB$(0,1,5)$	1	0.075	1	8	1	0.007	1	6				
RSB$(0,1,10)$	1	0.082	0	0	1	0.096	1	10	1	0.015	0	0
RSB$(0,1,20)$	1	0.046	1	12	3	0.039	3	$2\times100+41$	6	0.050	6	$5\times100+3$
RSB$(0,1,30)$	2	0.035	2	$100+14$	1	0.282	0	0	1	0.028	1	68
RSB$(0,1,40)$	1	0.026	1	15	10	0.024	10	$9\times100+65$	2	0.025	2	$100+17$
RSB$(0,1,50)$	1	0.233	0	0	1	0.240	1	8	2	0.020	2	$100+58$
RSB$(-1,1,5)$	1	0.205	0	0	1	0.205	0	0				
RSB$(-1,1,10)$	1	0.219	1	14	2	0.261	2	$100+76$	1	0.035	0	0
RSB$(-1,1,20)$	9	0.024	8	8×100	3	0.080	3	$2\times100+92$	2	0.203	2	$100+66$
RSB$(-1,1,30)$	2	0.129	2	$100+42$	10	0.118	10	$9\times100+39$	2	0.178	1	100
RSB$(-1,1,40)$	1	0.910	1	45	15	0.091	15	$14\times100+73$	3	0.077	3	$2\times100+28$
RSB$(-1,1,50)$	2	0.080	2	$100+62$	7	0.074	7	$6\times100+63$	6	0.077	6	$5\times100+41$

References

[1] S. Adly and H. Rammal, *A new method for solving second-order cone eigenvalue complementarity problem*, to appear in Journal of Optimization Theory and Applications .

[2] S. Adly and A. Seeger, *A nonsmooth algorithm for cone-constrained eigenvalue problems*, Computational Optimization and Applications 49 (2011), pp. 299–318.

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

[4] M.S. Bazaraa, H.D. Sherali, and C.M. Shetty, *Nonlinear Programming: Theory and Algorithms, 3rd edition*, John Wiley & Sons, New York 2006.

[5] A. Brooke, D. Kendrick, A. Meeraus, and R. Raman, *Gams a User's Guide*, GAMS Development Corporation, Washington 1998.

[6] C. Brás, J.J. Júdice, and H.D. Sherali, *On the solution of the inverse eigenvalue complementarity problem*, Journal of Optimization Theory and Applications 162 (2014), pp. 88–106.

[7] C. Brás, M. Fukushima, J. Júdice, and S. Rosa, *Variational inequality formulation of the asymmetric eigenvalue complementarity problem and its solution by means of gap functions*, Pacific Journal of Optimization 8 (2012), pp. 197–215.

[8] F.H. Clarke, *Optimization and nonsmooth analysis*, Society for Industrial and Applied Mathematics 1990, Available at http://books.google.pt/books?id=bVse-WvJvLYC.

[9] F. Facchinei and J. Pang, *Finite-dimensional variational inequalities and complementarity problems*, Springer, New York 2003.

[10] L.M. Fernandes, J.J. Júdice, H.D. Sherali, and M.A. Forjaz, *On an enumerative algorithm for solving eigenvalue complementarity problems*, Computational Optimization and Applications 59 (2014), pp. 113–134.

[11] L.M. Fernandes, J.J. Júdice, H.D. Sherali, and M. Fukushima, *On the computation of all eigenvalues for the eigenvalue complementarity problem*, Journal of Global Optimization 59 (2014), pp. 307–326.

[12] M. Fukushima, Z. Luo, and P. Tseng, *Smoothing functions for second-order-cone complementarity problems*, SIAM Journal on Optimization 12 (2002), pp. 436–460.

[13] P. Gill, W. Murray, and M. Saunders, *SNOPT: An SQP algorithm for large-scale constrained optimization*, SIAM Journal on Optimization 12 (2002), pp. 979–1006.

[14] S. Hayashi, N. Yamashita, and M. Fukushima, *A combined smoothing and regularization method for monotone second-order cone complementarity problems*, SIAM Journal on Optimization 15 (2005), pp. 593–615.

[15] J. Júdice, H.D. Sherali, and I. Ribeiro, *The eigenvalue complementarity problem*, Computational Optimization and Applications 37 (2007), pp. 139–156.

[16] J. Júdice, M. Raydan, S. Rosa, and S. Santos, *On the solution of the symmetric eigenvalue complementarity problem by the spectral projected gradient algorithm*, Numerical Algorithms 44 (2008), pp. 391–407.

[17] J. Júdice, H.D. Sherali, I. Ribeiro, and S. Rosa, *On the asymmetric eigenvalue complementarity problem*, Optimization Methods and Software 24 (2009), pp. 549–586.

[18] C. Kanzow, I. Ferenczi, and M. Fukushima, *On the local convergence of semismooth Newton methods for linear and nonlinear second-order cone programs without strict complementarity*, SIAM Journal on Optimization 20 (2009), pp. 297–320.

[19] H. Le Thi, M. Moeini, T. Pham Dinh, and J. Júdice, *A DC programming approach for solving the symmetric eigenvalue complementarity problem*, Computational Optimization and Applications 51 (2012), pp. 1097–1117.

[20] B. Murtagh and A. Saunders, *MINOS 5.0 User's Guide. Technical Report SOL 83-20*, Department of Operations Research, Stanford University 1983.

[21] Y.S. Niu, T. Pham, H.A. Le Thi, and J. Júdice, *Efficient DC programming approaches for the asymmetric eigenvalue complementarity problem*, Optimization Methods and Software 28 (2013), pp. 812–829.

[22] J. Nocedal and S.J. Wright, *Numerical optimization, 2nd edition*, Springer, New York 2006.

[23] J.S. Pang, D. Sun, and J. Sun, *Semismooth homeomorphisms and strong stability of semidefinite and lorentz complementarity problems*, Mathematics of Operations Research 28 (2003), pp. 39–63.

[24] A. Pinto da Costa and A. Seeger, *Cone-constrained eigenvalue problems: theory and algorithms*, Computational Optimization and Applications 45 (2010), pp. 25–57.

[25] A. Pinto da Costa, J. Martins, I. Figueiredo, and J. Júdice, *The directional instability problem in systems with frictional contacts*, Computer Methods in Applied Mechanics and Engineering 193 (2004), pp. 357–384.

[26] L. Qi and J. Sun, *A nonsmooth version of Newton's method*, Mathematical Programming 58 (1993), pp. 353–367.

[27] N. Sahinidis and M. Tawarmalani, *Baron 7.2.5: Global optimization of mixed-integer nonlinear*

programs, User's Manual 2005.

[28] A. Seeger, *Eigenvalue analysis of equilibrium processes defined by linear complementarity conditions*, Linear Algebra and its Applications 292 (1999), pp. 1–14.

[29] A. Seeger and M. Torki, *On eigenvalues induced by a cone constraint*, Linear Algebra and its Applications 372 (2003), pp. 181 – 206.

[30] H.D. Sherali and W. Adams, *A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems*, Kluwer Academic Publishers, Dordrecht 1999.

[31] H.D. Sherali and I. Al-loughani, *Equivalent primal and dual differentiable reformulations of the Euclidean multifacility location problem*, IIE Transactions 30 (1998), pp. 1065–1074.

[32] H.D. Sherali and C. Tuncbilek, *A global optimization algorithm for polynomial programming problems using a reformulation-linearization technique*, Journal of Global Optimization 2 (1992), pp. 101–112.

[33] A. Wächter and L.T. Biegler, *On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming*, Mathematical Programming 106 (2006), pp. 25–57, Available at http://dx.doi.org/10.1007/s10107-004-0559-y.

[34] M.S. Zhou Yihuiand Gowda, *On the finiteness of the cone spectrum of certain linear transformations on Euclidean Jordan algebras*, Linear Algebra and its Applications 431 (2009), pp. 772–782.