Comput Optim Appl
DOI 10.1007/s10589-010-9388-5

A DC programming approach for solving the symmetric Eigenvalue Complementarity Problem

Hoai An Le Thi · Mahdi Moeini · Tao Pham Dinh · Joaquim Judice

Received: 18 January 2010

Abstract In this paper, we investigate a DC (Difference of Convex functions) programming technique for solving large scale Eigenvalue Complementarity Problems (EiCP) with real symmetric matrices. Three equivalent formulations of EiCP are considered. We first reformulate them as DC programs and then use DCA (DC Algorithm) for their solution. Computational results show the robustness, efficiency, and high speed of the proposed algorithms.

Keywords DC programming · DCA · Eigenvalue Complementarity Problem

1 Introduction

Historically, numerical computation of eigenvalues of a matrix returns to more than 150 years (see [10, 13, 43]). Computing eigenvalues is rather important in pure and

H.A. Le Thi (✉) · M. Moeini
Theoretical and Applied Computer Science Laboratory (LITA), EA 3097, Paul Verlaine University - Metz, Ile du Saulcy, 57045 Metz Cedex, France
e-mail: lethi@univ-metz.fr

M. Moeini
e-mail: moeini@univ-metz.fr

T. Pham Dinh
Laboratory of Modelling, Optimization & Operations Research, National Institute for Applied Sciences - Rouen, BP 08, Place Emile Blondel, 76131 Mont Saint Aignan Cedex, France
e-mail: pham@insa-rouen.fr

J. Judice
Instituto de Telecomunicações, Coimbra, Portugal
e-mail: Joaquim.Judice@co.it.pt

J. Judice
Department of Mathematics, University of Coimbra, Coimbra, Portugal

Published online: 22 December 2010

applied mathematics, physics, and engineering. A non-exhaustive list of applications where the calculation of the eigenvalues is required includes structural dynamics, electrical networks, quantum chemistry, chemical reactions, control theory and economics [6, 31, 37].

Some applications concerning the computation of eigenvalues are also studied in [9, 37, 39]. In most of the applications, the matrices are real valued and symmetric [31, 34, 37]. These matrices are very important in linear algebra and enjoy some interesting properties. For instance, all their eigenvalues are real [12].

Mathematically speaking, for a square matrix A, the eigenvalues of A are the roots of $det(A - \lambda I)$ called the characteristic polynomial of A. In realistic applications, the eigenvalues of A may be calculated for different aims. In some cases, the eigenvalues have an intrinsic meaning, e.g., for the expected long-time behavior of a dynamical system; in others they are just meaningless intermediate values of a computational method [18].

In some circumstances, it is advantageous to use an optimization approach to solve eigenvalue problems with real symmetric matrices. A classical optimization approach involves optimizing the Rayleigh quotient [26]. This function enjoys some interesting properties, namely any stationary point is an eigenvector and the corresponding eigenvalue is given by the Rayleigh quotient. Rayleigh quotient is not the only function that has been used in the optimization approaches for computing eigenvalues. Some alternatives have been presented and discussed in [1, 4, 5, 15, 26].

The Eigenvalue Complementarity Problem (EiCP) is an extension of the classical eigenvalue problem that has been widely studied [8, 9, 14–16, 34, 39, 46]. For given matrices A and B, EiCP consists of finding $\lambda \in \mathbb{R}$ and $\mathbf{x} \neq 0$ such that

$$\begin{cases} \mathbf{w} = (\lambda B - A)\mathbf{x}, \\ \mathbf{w} \geq \mathbf{0}, \\ \mathbf{x} \geq \mathbf{0}, \\ \mathbf{w}^T \mathbf{x} = 0. \end{cases}$$

EiCP is symmetric when A and B are both symmetric matrices. In this case the EiCP has been shown to be equivalent to finding a stationary point of a generalized Rayleigh quotient on the simplex [34].

In general the equivalent optimization problems are NP-complete [7, 34] and very difficult to solve efficiently, particularly when the dimension of the problem is large. The purpose of this paper is to present a new optimization approach for solving the symmetric EiCP when B is positive definite. This approach is used for solving Rayleigh quotient, quadratically constrained, and logarithmic optimization formulations of the symmetric EiCP [15, 26].

Our approach is a local deterministic method based on DC (Difference of Convex functions) programming. The DC Algorithm (DCA) was first introduced, in its preliminary form, by Pham Dinh Tao in 1985, and has been extensively developed since 1994 by Le Thi Hoai An and Pham Dinh Tao. It becomes now classic and popular (see e.g. [11, 20, 22, 24, 30, 36, 45] and references therein, and http://lita.sciences.univ-metz.fr/~lethi/).

It has been successfully applied to many large-scale (smooth or nonsmooth) nonconvex programs in various domains of applied sciences, for example *tomography* [45] and *machine learning* [24, 30, 36]. Numerical experiments show that DCA is in many cases more robust and efficient than standard methods (see e.g. [2, 3, 11, 17, 20–22, 24, 30, 32, 33, 38, 40–42, 45] and references therein).

Several equivalent formulations of EiCP are known in the literature and we have chosen the three most interesting for the use of DCA. Solving the EiCP amounts to finding a Karush-Kuhn-Tucker (KKT) point of the three considered optimization problems. Hence we do not need to find a global minimum for these problems, and using DCA should be a good choice regarding the following aspects:

– mathematical foundations of the algorithms,
– rate of convergence and running-time,
– ability to treating large-scale problems.

We first formulate the underlying optimization models in the form of DC programs in which a DC function is minimized over a closed convex set. Then, DCA is used to solve the DC programs. The generic DCA addresses a DC program which takes the form

$$\alpha = \inf\{f(x) := g(x) - h(x) : x \in \mathbb{R}^n\} \quad (P_{dc}) \tag{1}$$

where g, h are lower semicontinuous proper convex functions on $\mathbb{R}^n$ (the closed convex constraint set C is included by using its indicator function χ_C ($\chi_C(x) = 0$ if $x \in C, +\infty$ otherwise) as below. Such a function f is called DC function, and $g - h$ a DC decomposition of f while g and h are DC components of f. The construction of DCA involves the DC components g and h but not the function f itself. Each iteration k of DCA consists of computing

$$y^k \in \partial h(x^k), \qquad x^{k+1} \in \arg\min\{g(x) - h(x^k) - \langle x - x^k, y^k\rangle : x \in \mathbb{R}^n\} \quad (P_k).$$

Hence, for a DC program, each DC decomposition corresponds to a different version of DCA. Since a DC function f has an infinite number of DC decompositions which have a crucial impact on the speed of convergence, robustness, efficiency and quality of computed solution given by the DCA, the search for a good DC decomposition is very important. The development of an efficient algorithm based on the generic DCA scheme for a practical problem is thus a judicious question to be studied, and the answer depends on the specific structure of the problem being considered. This question is carefully studied in the current paper. Attempting to find g and h such that the computations of y^k and x^k are not expensive, we propose three DC decompositions that lead to DCA schemes in which the convex subproblems can be solved efficiently. In particular, in Algorithms 2 and 3 the convex subproblems are essentially projections of points onto the standard $n-1$-simplex.

Computational experiments show that DCA is quite efficient for the symmetric EiCP and compares favorably with one of the best existing algorithms, i.e., the Spectral Projected Gradient Algorithm (SPGA) [15].

The structure of the paper is as follows. The classical eigenvalue problem is reviewed and three formulations of the eigenvalue problem are presented in the Sect. 2. In Sect. 3 we give an outline of general DC programs and DCA. The DCA for these

Table 1 Notation

n	the dimension of the matrices;
$\lambda \in \mathbb{R}$	an eigenvalue of a matrix like A;
$\lambda_k \in \mathbb{R}$	k-th eigenvalue of a matrix like A;
$\mathbf{x}, \mathbf{w}, \mathbf{y}$	n-dimensional vectors in $\mathbb{R}^n$;
$\mathbf{e}$	$\mathbf{e} = (1, \ldots, 1)^T$;
$\mathbf{e}^j$	j-th canonical basic vector i.e., $\mathbf{e}^j = (0, \ldots, 1, \ldots, 0)^T$ in which 1 is located at the j-th position.
SPD	the set of symmetric positive definite matrices

three formulations are discussed in Sect. 4. Computational experiments are reported in Sect. 5 whereas the last section includes some conclusions and future work.

A full description of the notations used in this paper is given in Table 1. All quantities in boldface represent vectors in $\mathbb{R}^n$ unless otherwise noted. The transpose of a vector or a matrix will be denoted with the symbol ^{T}.

2 Eigenvalue problems and optimization

In this section we first give a general introduction to EiCP and then, present three equivalent optimization formulations of EiCP.

For the symmetric matrices A and B, where B is positive definite, a solution of the Eigenvalue Complementarity Problem (EiCP) is an eigenpair $(\lambda, \mathbf{x})$, where $\lambda \in \mathbb{R}$ is a *Complementary Eigenvalue* and $\mathbf{x} \in \mathbb{R}^n$ is *Complementary Eigenvector* corresponding to λ. For a given eigenpair $(\lambda, \mathbf{x})$, the couple $(\lambda, \alpha\mathbf{x})$ is also an eigen-pair of (A, B), for all $\alpha > 0$, and in fact, the set of all complementary eigenvectors associated to a certain eigenvalue is a *cone* [15]. Hence, without loss of generality, we consider only the solutions satisfying $\langle \mathbf{e}, \mathbf{x} \rangle = 1$. Using this new constraint, EiCP becomes

$$\text{(EiCP):} \qquad \text{Find } \lambda \in \mathbb{R} \text{ such that } \begin{cases} \mathbf{w} = (\lambda B - A)\mathbf{x}, \\ \mathbf{w} \geq \mathbf{0}, \\ \mathbf{x} \geq \mathbf{0}, \\ \mathbf{w}^T\mathbf{x} = 0, \\ \langle \mathbf{e}, \mathbf{x} \rangle = 1. \end{cases}$$

Let $\phi(\mathbf{x})$ be a continuously differentiable function. For some *suitable* choices of $\phi(\mathbf{x})$, EiCP can be reduced to the following nonlinear program [15, 34]

$$\min \left\{ \phi(\mathbf{x}) : \mathbf{e}^T\mathbf{x} = 1, \mathbf{x} \geq \mathbf{0} \right\}. \tag{2}$$

In what follows, we present three equivalent formulations of EiCP that have been first discussed in [15, 34].

2.1 Rayleigh quotient formulation

In this case $\phi(\mathbf{x})$ consists of the generalized Rayleigh quotient given by

$$\phi(\mathbf{x}) := -\frac{\mathbf{x}^T A\mathbf{x}}{\mathbf{x}^T B\mathbf{x}}.$$

Then, any KKT point $\mathbf{x}^*$ of (2) gives a solution to EiCP $(\mathbf{x}^*, \lambda^*)$, where

$$\lambda^* = (\mathbf{x}^{*T} A\mathbf{x}^*)/(\mathbf{x}^{*T} B\mathbf{x}^*) \quad [15, 34].$$

2.2 Logarithmic formulation

An alternative to the generalized Rayleigh quotient is the following logarithmic function [4, 15, 26]

$$\phi(\mathbf{x}) := \ln(\mathbf{x}^T B\mathbf{x}) - \ln(\mathbf{x}^T A\mathbf{x})$$

where A has to be at least strictly copositive. This condition and the fact of $B \in SPD$, make possible the use of the logarithmic function.

As before, a KKT point $\mathbf{x}^*$ of (2) leads to the solution $(\mathbf{x}^*, \lambda^*)$ of EiCP such that

$$\lambda^* = (\mathbf{x}^{*T} A\mathbf{x}^*)/(\mathbf{x}^{*T} B\mathbf{x}^*) \quad [15].$$

2.3 Quadratic formulation

EiCP can also be reformulated as the following quadratically constrained quadratic programming problem

$$\min\left\{\frac{1}{2}\langle \mathbf{x}, -A\mathbf{x}\rangle : \langle \mathbf{x}, B\mathbf{x}\rangle = 1, \mathbf{x} \geq \mathbf{0}\right\}. \tag{3}$$

If there is an $0 \neq \overline{\mathbf{x}} \geq 0$ such that $\langle \overline{\mathbf{x}}, A\overline{\mathbf{x}}\rangle > 0$, then [19, 29] $\mathbf{x}^*$ is a KKT point of (3) if and only if $\mathbf{x}^*$ is a KKT point of

$$\min\left\{\frac{1}{2}\langle \mathbf{x}, -A\mathbf{x}\rangle : \langle \mathbf{x}, B\mathbf{x}\rangle \leq 1, \ \mathbf{x} \geq 0\right\} \tag{4}$$

Then $\langle \mathbf{x}^*, \lambda^*\rangle$ is a solution of EiCP, with $\lambda^* = \langle \mathbf{x}^*, A\mathbf{x}^*\rangle$ [15]. Since the feasible set of (3) is not convex, (3) is **in general** not a DC program and DCA cannot be applied to this formulation. Instead we consider the quadratic formulation (4).

3 General DC programs and DCA

Let $\Gamma_0(\mathbb{R}^n)$ denotes the convex cone of all lower semi-continuous proper convex functions on $\mathbb{R}^n$. Consider the following primal DC program

$$(P_{dc}) \qquad \beta_p = \inf\{F(x) := g(x) - h(x) \ : \ x \in \mathbb{R}^n\},$$

where $g, h \in \Gamma_0(\mathbb{R}^n)$.

A DC program (P_{dc}) is called a DC polyhedral program when either g or h is a polyhedral convex function (i.e., the pointwise supremum of a finite collection of affine functions). Note that a polyhedral convex function is almost always differentiable, say, it is differentiable everywhere except on a set of measure zero.

Let C be a nonempty closed convex set. Then, the problem

$$\inf\{f(x) := g(x) - h(x) : x \in C\}, \tag{5}$$

can be transformed into an unconstrained DC program by using the indicator function of C, i.e.,

$$\inf\{f(x) := \phi(x) - h(x) : x \in \mathbb{R}^n\}, \tag{6}$$

where $\phi := g + \chi_C$ is in $\Gamma_0(\mathbb{R}^n)$.

Let $g^*(y) := \sup\{\langle x, y\rangle - g(x) : x \in \mathbb{R}^n\}$ be the conjugate function of g. Then, the following program is called the dual program of (P_{dc}):

$$(D_{dc}) \qquad \beta_d = \inf\{h^*(y) - g^*(y : y \in \mathbb{R}^n\}. \tag{7}$$

Under the natural convention in DC programming, that is, $+\infty - (+\infty) = +\infty$, and by using the fact that every function $h \in \Gamma_0(\mathbb{R}^n)$ is characterized as a pointwise supremum of a collection of affine functions, say

$$h(x) := \sup\{\langle x, y\rangle - h^*(y) : y \in \mathbb{R}^n\},$$

it can be proved that $\beta_p = \beta_d$ [33]. There is a perfect symmetry between primal and dual DC programs, that is the dual of (D_{dc}) is (P_{dc}).

Recall that, for $\theta \in \Gamma_0(\mathbb{R}^n)$ and $x_0 \in dom\,\theta := \{x \in \mathbb{R}^n | \theta(x_0) < +\infty\}$, the subdifferential of θ at x_0, denoted $\partial\theta(x_0)$, is defined as [35]

$$\partial\theta(x_0) := \{y \in \mathbb{R}^n : \theta(x) \geq \theta(x_0) + \langle x - x_0, y\rangle, \forall x \in \mathbb{R}^n\} \tag{8}$$

which is a closed convex set in $\mathbb{R}^n$. It generalizes the derivative in the sense that θ is differentiable at x_0 if and only if $\partial\theta(x_0)$ is reduced to a singleton which is exactly $\{\nabla\theta(x_0)\}$.

The *modulus of strong convexity* of θ on C, denoted by $\rho(\theta, C)$ or $\rho(\theta)$ if $C = \mathbb{R}^n$, is given by

$$\rho(\theta, C) = \sup\{\rho \geq 0 : \theta - (\rho/2)\|\cdot\|^2 \text{ is convex on } C\}. \tag{9}$$

Clearly, θ is convex on C if and only if $\rho(\theta, C) \geq 0$. Furthermore θ is *strongly convex* on C if $\rho(\theta, C) > 0$.

The necessary local optimality condition for the primal DC program, (P_{dc}), is

$$\partial h(x^*) \subset \partial g(x^*). \tag{10}$$

The condition (10) is also sufficient for many important classes of DC programs, for example, for DC polyhedral programs, or when function f is locally convex at x^* [22, 32].

A point x^* satisfying the generalized Kuhn-Tucker condition

$$\partial h(x^*) \cap \partial g(x^*) \neq \emptyset \tag{11}$$

is called a critical point of $g - h$. It follows that if h is a polyhedral convex function, then a critical point of $g - h$ is almost always a local solution to (P_{dc}).

Based on local optimality conditions and duality in DC programming, the DC Algorithm (DCA) consists in constructing two sequences $\{x^k\}$ and $\{y^k\}$ of trial solutions for the primal and dual programs, respectively, such that the sequences $\{g(x^k) - h(x^k)\}$ and $\{h^*(y^k) - g^*(y^k)\}$ are decreasing, and $\{x^k\}$ (resp.$\{y^k\}$) converges to a primal feasible solutions $\tilde{x}$ (resp. a dual feasible solution $\tilde{y}$) satisfying the local optimality condition and

$$\tilde{x} \in \partial g^*(\tilde{y}), \qquad \tilde{y} \in \partial h(\tilde{x}). \tag{12}$$

DCA then yields the next simple scheme:

$$y^k \in \partial h(x^k); \qquad x^{k+1} \in \partial g^*(y^k). \tag{13}$$

In other words, these two sequences $\{x^k\}$ and $\{y^k\}$ are determined in the way that x^{k+1} and y^{k+1} are solutions of the convex primal program (P_l) and dual program (D_{l+1}), respectively. These are defined as

$$(P_k) \qquad \inf\{g(x) - h(x^k) - \langle x - x^k, y^k\rangle \ : \ x \in \mathbb{R}^n\}, \tag{14}$$

$$(D_{k+1}) \qquad \inf\{h^*(y) - g^*(y^k) - \langle y - y^k, x^{k+1}\rangle : y \in \mathbb{R}^n\}. \tag{15}$$

At each iteration, the DCA performs a double linearization with the use of the subgradients of h and g^*. In fact, in each iteration, one replaces in the primal DC program, (P_{dc}), the second component h by its affine minorization $h_k(x) := h(x^k) + \langle x - x^k, y^k\rangle$ to construct the convex program (P_k) whose solution set is nothing but $\partial g^*(y^k)$. Likewise, the second DC component g^* of the dual DC program, (D_{dc}), is replaced by its affine minorization $g_k^*(y) := g^*(y^k) + \langle y - y^k, x^{k+1}\rangle$ to obtain the convex program (D_{k+1}) whose $\partial h(x^{k+1})$ is the solution set. Hence DCA works with the convex DC components g and h but not with the DC function f itself.

Convergence properties of the DCA and its theoretical basis are described in [20, 22, 32, 33]. However, it is worthwhile to summarize the following properties for the sake of completeness:

i) The sequences $\{g(x^k) - h(x^k)\}$ and $\{h^*(y^k) - g^*(y^k)\}$ are decreasing and
   - $g(x^{k+1}) - h(x^{k+1}) = g(x^k) - h(x^k)$ if and only if $y^k \in \partial g(x^k) \cap \partial h(x^k)$, $y^k \in \partial g(x^{k+1}) \cap \partial h(x^{k+1})$ and $[\rho(g) + \rho(h)]\|x^{k+1} - x^k\| = 0$.
   - $h^*(y^{k+1}) - g^*(y^{k+1}) = h^*(y^k) - g^*(y^k)$ if and only if $x^{k+1} \in \partial g^*(y^k) \cap \partial h^*(y^k)$, $x^{k+1} \in \partial g^*(y^{k+1}) \cap \partial h^*(y^{k+1})$ and $[\rho(g^*) + \rho(h^*)]\|y^{k+1} - y^k\| = 0$.

   DCA terminates at the k^{th} iteration if either of the above equalities holds.

ii) If $\rho(g) + \rho(h) > 0$ (resp. $\rho(g^*) + \rho(h^*) > 0$), then the series $\{\|x^{k+1} - x^k\|^2\}$ (resp. $\{\|y^{k+1} - y^k\|^2\}$) converges (one says that DCA is asymptotically stable in this case).

iii) If the optimal value of problem (P_{dc}) is finite and the sequences $\{x^k\}$ and $\{y^k\}$ are bounded, then every limit point x^∞ (resp. y^∞) of the sequence $\{x^k\}$ (resp. $\{y^k\}$) is a critical point of $g - h$ (resp. $h^* - g^*$).
iv) DCA possesses linear convergence for general DC programs.
v) In polyhedral DC programs, the sequences DCA $\{x^k\}$ and $\{y^k\}$ contain finitely many elements and DCA possesses finite convergence.

4 DC formulations and DCA for EiCP

In this section, we show how to use DC programming and DCA for solving the above equivalent formulations of EiCP. We first reformulate them in the standard form of DC programming that is, minimizing a DC function over a convex set. Next we present the resulting DCA to solve the corresponding DC program.

4.1 Quadratic formulation

Consider the quadratic formulation of EiCP

$$\min\left\{F(\mathbf{x}) := \frac{1}{2}\langle \mathbf{x}, -A\mathbf{x}\rangle : \langle \mathbf{x}, B\mathbf{x}\rangle \leq 1, \mathbf{x} \geq \mathbf{0}\right\}, \tag{16}$$

where A and B are $n \times n$ symmetric matrices and B is a positive definite matrix.

Since $-A$ is not a positive semi-definite matrix [34], it is very difficult to solve (16) efficiently.

Let μ be a positive number such that $(\mu I + A)$ is a positive semi-definite matrix. To get a DC decomposition of $F(\mathbf{x})$ we first write

$$F(\mathbf{x}) := \frac{1}{2}\mu\|\mathbf{x}\|^2 - \frac{1}{2}\langle \mathbf{x}, (\mu I + A)\mathbf{x}\rangle. \tag{17}$$

Let

$$\Omega := \{\mathbf{x} \in \mathbb{R}^n : \langle \mathbf{x}, B\mathbf{x}\rangle \leq 1, \mathbf{x} \geq \mathbf{0}\}.$$

A DC formulation of the problem (16) can be

$$\min\{g(\mathbf{x}) - h(\mathbf{x}) : \mathbf{x} \in \mathbb{R}^n\}, \tag{18}$$

where

$$g(\mathbf{x}) := \frac{1}{2}\mu\,\|\mathbf{x}\|^2 + \chi_\Omega(\mathbf{x}), \tag{19}$$

$$h(\mathbf{x}) := \frac{1}{2}\langle \mathbf{x}, (\mu I + A)\mathbf{x}\rangle, \tag{20}$$

and the indicator function on Ω, χ_Ω is defined as

$$\chi_\Omega(\mathbf{x}) = \begin{cases} 0 & \text{if } \mathbf{x} \in \Omega, \\ +\infty & \text{otherwise.} \end{cases} \tag{21}$$

The DCA applied to this DC formulation can be described as follows.

Algorithm 1 - DCA for the quadratic formulation

1. Let $\mathbf{x}^0 \in \mathbb{R}^n$ be given and let ϵ be a sufficiently small positive number. Set $k := 0$.
2. Set $\mathbf{y}^k := (\mu I + A)\mathbf{x}^k$ and solve the convex quadratic program

$$\min\left\{\frac{1}{2}\mu \|\mathbf{x}\|^2 - \langle \mathbf{x}, \mathbf{y}^k\rangle : \langle \mathbf{x}, B\mathbf{x}\rangle \leq 1, \mathbf{x} \geq \mathbf{0}\right\}$$

   to obtain $\mathbf{x}^{k+1}$.
3. If $F(\mathbf{x}^k) - F(\mathbf{x}^{k+1}) \leq \epsilon$ then STOP and take $\mathbf{x}^{k+1}$ as an optimal solution; else set $k := k + 1$ and go to step 2.

The main operation at each iteration of the algorithm consists of solving a strictly convex separable quadratic program under a convex quadratic constraint with non-negative variables.

Theorem 1 (Convergence properties of Algorithm 1)

i) *Algorithms 1 generates a sequence* $\{x^k\}$ *such that the sequence* $\{g(x^k) - h(x^k)\}$ *is Decreasing.*
ii) *The sequence* $\{\|x^{k+1} - x^k\|^2\}$ *converges.*
iii) *The sequence* $\{x^k\}$ *converges to a point* x^* *that is a KKT point of the DC program.*
iv) $\langle \mathbf{x}^*, \lambda^*\rangle$ *is a solution of EiCP, where* $\lambda^* = \langle \mathbf{x}^*, A\mathbf{x}^*\rangle$.

Proof Since $\rho(g) > 0$, i) (resp. ii)) is a direct consequence of property i) (resp. property ii)) of the DCA convergence properties for a general DC program (see Sect. 3).

iii) is a direct consequence of the property i) of Theorem 2.4 of [23] concerning the convergence analysis of DCA with subanalytic functions. We emphasize here the convergence of the whole sequence $\{x^k\}$ generated by DCA, due to the subanalytic data, instead of the convergence of subsequences of $\{x^k\}$ in the general case.

iv) follows from iii). □

4.2 Logarithmic formulation

Consider the logarithmic formulation of EiCP

$$\min\{\ln(\langle \mathbf{x}, B\mathbf{x}\rangle) - \ln(\langle \mathbf{x}, A\mathbf{x}\rangle) : \langle \mathbf{e}, \mathbf{x}\rangle = 1, \mathbf{x} \geq \mathbf{0}\}, \tag{22}$$

where, as before, $B \in$ SPD and A is a symmetric strictly copositive matrix.

Define

$$L_{AB}(\mathbf{x}) := \ln(\mathbf{x}^T B\mathbf{x}) - \ln(\mathbf{x}^T A\mathbf{x})$$

and write $L_{AB}(\mathbf{x})$ in the form

$$L_{AB}(\mathbf{x}) = \frac{1}{2}\mu \|\mathbf{x}\|^2 - \left[\frac{1}{2}\mu \|\mathbf{x}\|^2 + \ln(\mathbf{x}^T A\mathbf{x}) - \ln(\mathbf{x}^T B\mathbf{x})\right]. \tag{23}$$

For a sufficiently large value of μ, the function $[\frac{1}{2}\mu\|\mathbf{x}\|^2 + \ln(\mathbf{x}^T A\mathbf{x}) - \ln(\mathbf{x}^T B\mathbf{x})]$ is convex and (23) is rewritten as a new DC decomposition of $L_{AB}(\mathbf{x})$.

Therefore the DC formulation corresponding to (23) is rewritten as

$$\min\{g(\mathbf{x}) - h(\mathbf{x}) : \mathbf{x} \in \mathbb{R}^n\}, \tag{24}$$

where

$$g(\mathbf{x}) := \frac{1}{2}\mu \|\mathbf{x}\|^2 + \chi_\Theta(\mathbf{x}), \tag{25}$$

$$h(\mathbf{x}) := \left[\frac{1}{2}\mu \|\mathbf{x}\|^2 + \ln(\mathbf{x}^T A\mathbf{x}) - \ln(\mathbf{x}^T B\mathbf{x})\right] \tag{26}$$

and

$$\Theta := \{\mathbf{x} \in \mathbb{R}^n : \langle \mathbf{e}, \mathbf{x}\rangle = 1, \mathbf{x} \geq \mathbf{0}\}$$

with χ_Θ being the indicator function on Θ. Then DCA takes the following form:

Algorithm 2 - DCA for the logarithmic formulation

1. Let $\mathbf{x}^0 \in \mathbb{R}^n$ be given and let ϵ be a sufficiently small positive number. Set $k := 0$.
2. Set $\mathbf{y}^k := (\mu I + \frac{2A}{\langle \mathbf{x}^k, A\mathbf{x}^k\rangle} - \frac{2B}{\langle \mathbf{x}^k, B\mathbf{x}^k\rangle})\mathbf{x}^k$ and solve the convex quadratic program

$$\min\left\{\frac{1}{2}\mu \|\mathbf{x}\|^2 - \langle \mathbf{x}, \mathbf{y}^k\rangle : \langle \mathbf{e}, \mathbf{x}\rangle = 1, \mathbf{x} \geq \mathbf{0}\right\}$$

   to obtain $\mathbf{x}^{k+1}$.
3. If $L_{AB}(\mathbf{x}^k) - L_{AB}(\mathbf{x}^{k+1}) \leq \epsilon$ then STOP and take $\mathbf{x}^{k+1}$ as an optimal solution; else set $k := k + 1$ and go to step 2.

The DC decomposition (24)–(26) leads to the Algorithm 2 whose main effort in each iteration consists of solving a strictly convex quadratic program on the $n-1$-simplex Θ. This is nothing else the projection of a point onto the standard simplex Θ for which a number of quite efficient algorithms exist (see e.g. [15]).

Estimation of μ Since the function $[-\ln(\mathbf{x}^T B\mathbf{x})]$ is convex and the sum of two convex functions is also convex, it is sufficient to take μ such that $[\frac{1}{2}\mu\|\mathbf{x}\|^2 + \ln(\mathbf{x}^T A\mathbf{x})]$ becomes convex. To this aim, it is sufficient that μ be greater than the spectral radius of the Hessian matrix of $\Lambda(x) := \ln(\mathbf{x}^T A\mathbf{x})$, i.e., $\mu \geq \rho(\nabla^2\Lambda(\mathbf{x}))$ for all $x \in \Theta$. The gradient and Hessian of $\Lambda(\mathbf{x})$ are respectively

$$\nabla\Lambda(\mathbf{x}) = \frac{2A\mathbf{x}}{\mathbf{x}^T A\mathbf{x}}, \tag{27}$$

and

$$\nabla^2\Lambda(x) = \frac{2A}{\mathbf{x}^T A\mathbf{x}} - \frac{4(A\mathbf{x})(A\mathbf{x})^T}{(\mathbf{x}^T A\mathbf{x})^2}. \tag{28}$$

But $(A\mathbf{x})(A\mathbf{x})^T$ is positive semi-definite for all $\mathbf{x} \in \mathbb{R}^n$, as

$$\mathbf{y}^T(A\mathbf{x})(A\mathbf{x})^T\mathbf{y} = (\mathbf{y}^T A\mathbf{x})(\mathbf{x}^T A\mathbf{y}) = (\mathbf{x}^T A\mathbf{y})^T(\mathbf{x}^T A\mathbf{y}) = \|\mathbf{x}^T A\mathbf{y}\|^2 \geq 0.$$

Furthermore, for all $\mathbf{x} \in \mathbb{R}^n$, $\nabla^2\Lambda(\mathbf{x})$ is a real symmetric matrix so it is a Hermitian matrix. We use the following result, known as the *monotonicity theorem* [12], in order to estimate $\rho(\nabla^2\Lambda(\mathbf{x}))$:

Theorem 2 *Let Λ and Γ be two $n \times n$ Hermitian matrices. Assume that Γ is PSD and the eigenvalues of Λ and $\Lambda + \Gamma$ are arranged in increasing order. Then*

$$\lambda_k(\Lambda) \leq \lambda_k(\Lambda + \Gamma) \quad \textit{for all } k = 1, \ldots, n,$$

where $\lambda_k(.)$ shows the k-th eigenvalue. Consequently,

$$\rho(\Lambda) \leq \rho(\Lambda + \Gamma),$$

where $\rho(.)$ is the spectral radius of the matrix $(.)$.

Proof See [12]. □

In order to apply this theorem for estimating μ, first note that $\nabla^2\Lambda(x)$ is a Hermitian matrix and $\frac{4(A\mathbf{x})(A\mathbf{x})^T}{(\mathbf{x}^T A\mathbf{x})^2}$ is positive semi-definite. Hence

$$\rho(\nabla^2\Lambda(\mathbf{x})) \leq \rho\left(\nabla^2\Lambda(\mathbf{x}) + \frac{4(A\mathbf{x})(A\mathbf{x})^T}{(\mathbf{x}^T A\mathbf{x})^2}\right), \quad \text{for all } \mathbf{x} \in \mathbb{R}^n.$$

But,

$$\frac{2A}{\mathbf{x}^T A\mathbf{x}} = \nabla^2\Lambda(\mathbf{x}) + \frac{4(A\mathbf{x})(A\mathbf{x})^T}{(\mathbf{x}^T A\mathbf{x})^2}.$$

Then

$$\rho(\nabla^2\Lambda(\mathbf{x})) \leq \rho\left(\frac{2A}{\mathbf{x}^T A\mathbf{x}}\right), \quad \text{for all } \mathbf{x} \in \mathbb{R}^n.$$

Hence, $\rho(\Lambda) \leq \rho(|\Lambda|)$ for any $n \times n$ matrix Λ, and

$$\rho\left(\frac{2A}{\mathbf{x}^T A\mathbf{x}}\right) \leq \rho\left(\frac{2|A|}{\mathbf{x}^T A\mathbf{x}}\right) \leq \rho\left(\frac{2|A|}{\alpha}\right),$$

where $\alpha := \min\{\mathbf{x}^T A\mathbf{x} : \langle \mathbf{e}, \mathbf{x}\rangle = 1, \mathbf{x} \geq \mathbf{0}\}$. Note that A must be a positive semi-definite matrix for α to be computed relatively easily. In fact, finding α for a general strictly copositive matrix is a NP-hard problem [28].

4.3 Generalized Rayleigh quotient formulation

Consider the Rayleigh quotient formulation of the EiCP

$$\max\left\{\frac{\langle \mathbf{x}, A\mathbf{x}\rangle}{\langle \mathbf{x}, B\mathbf{x}\rangle} : \langle \mathbf{e}, \mathbf{x}\rangle = 1, \mathbf{x} \geq \mathbf{0}\right\}, \tag{29}$$

where A and B are two symmetric matrices and $B \in SPD$.

For finding a DC reformulation of (29), we start by writing

$$\frac{\langle \mathbf{x}, A\mathbf{x}\rangle}{\langle \mathbf{x}, B\mathbf{x}\rangle} = \frac{\langle \mathbf{x}, A\mathbf{x}\rangle + \nu\langle \mathbf{x}, B\mathbf{x}\rangle - \nu\langle \mathbf{x}, B\mathbf{x}\rangle}{\langle \mathbf{x}, B\mathbf{x}\rangle} = \frac{\langle \mathbf{x}, (A+\nu B)\mathbf{x}\rangle}{\langle \mathbf{x}, B\mathbf{x}\rangle} - \nu,$$

where ν can be any real number. Now let ν be a sufficiently large positive number such that the matrix $C := (A + \nu B)$ is positive definite. Then the problem (29) is equivalent to the following fractional programming problem

$$\max\left\{\frac{\langle \mathbf{x}, C\mathbf{x}\rangle}{\langle \mathbf{x}, B\mathbf{x}\rangle} : \langle \mathbf{e}, \mathbf{x}\rangle = 1, \mathbf{x} \geq \mathbf{0}\right\}, \tag{30}$$

where both $C := (A + \nu B)$ and B are positive definite matrices. Hence, solving (30) amounts to solving the next problem

$$\max\left\{\ln(\langle \mathbf{x}, C\mathbf{x}\rangle) - \ln(\langle \mathbf{x}, B\mathbf{x}\rangle) : \langle \mathbf{e}, \mathbf{x}\rangle = 1, \mathbf{x} \geq \mathbf{0}\right\} \tag{31}$$

or again

$$\min\left\{\ln(\langle \mathbf{x}, B\mathbf{x}\rangle) - \ln(\langle \mathbf{x}, C\mathbf{x}\rangle) : \langle \mathbf{e}, \mathbf{x}\rangle = 1, \mathbf{x} \geq \mathbf{0}\right\}. \tag{32}$$

Like (22), (32) is a DC program of the form

$$\min\left\{L_{CB}(\mathbf{x}) := g(\mathbf{x}) - h(\mathbf{x}) : \mathbf{x} \in \mathbb{R}^n\right\}, \tag{33}$$

where

$$g(\mathbf{x}) := \frac{1}{2}\mu\,\|\mathbf{x}\|^2 + \chi_\Theta(\mathbf{x}), \tag{34}$$

$$h(\mathbf{x}) := \left[\frac{1}{2}\mu\,\|\mathbf{x}\|^2 + \ln(\mathbf{x}^T C\mathbf{x}) - \ln(\mathbf{x}^T B\mathbf{x})\right]. \tag{35}$$

Here μ can be chosen as $\mu \geq \rho(\frac{2|C|}{\beta})$, with $\beta := \min\{\mathbf{x}^T C\mathbf{x} : \langle \mathbf{e}, \mathbf{x}\rangle = 1, \mathbf{x} \geq \mathbf{0}\}$.

We find again a similar DC formulation of (24) where A is replaced by C. Therefore the DCA applied to (33) is nothing else but Algorithm 2 in which A is replaced by C. It is described as follows.

Algorithm 3 - DCA for the Rayleigh quotient formulation

1. Let $\mathbf{x}^0 \in \mathbb{R}^n$ be given and let ϵ be a sufficiently small positive number. Set $k := 0$.
2. Set $\mathbf{y}^k := (\mu I + \frac{2C}{\langle \mathbf{x}^k, C\mathbf{x}^k \rangle} - \frac{2B}{\langle \mathbf{x}^k, B\mathbf{x}^k \rangle})\mathbf{x}^k$ and solve the convex quadratic program

$$\min \left\{ \frac{1}{2}\mu \|\mathbf{x}\|^2 - \langle \mathbf{x}, \mathbf{y}^k \rangle : \langle \mathbf{e}, \mathbf{x} \rangle = 1, \mathbf{x} \geq \mathbf{0} \right\}$$

   to obtain $\mathbf{x}^{k+1}$.
3. If $L_{CB}(\mathbf{x}^k) - L_{CB}(\mathbf{x}^{k+1}) \leq \epsilon$ then STOP and take $\mathbf{x}^{k+1}$ as an optimal solution; else set $k := k + 1$ and go to step 2.

Theorem 3 (Convergence properties of Algorithms 2 and 3)

i) *Algorithms 2, 3 generate a sequence* $\{x^k\}$ *such that the corresponding sequence* $\{g(x^k) - h(x^k)\}$ *is decreasing.*
ii) *The series* $\{\|x^{k+1} - x^k\|^2\}$ *converges.*
iii) *The sequence* $\{x^k\}$ *converges to a point* x^* *that is a KKT point of the corresponding DC program (24) or (33).*
iv) $\langle \mathbf{x}^*, \lambda^* \rangle$ *is a solution of EiCP, where* $\lambda^* = (\mathbf{x}^{*\mathbf{T}} A\mathbf{x}^*)/(\mathbf{x}^{*T} B\mathbf{x}^*)$.

Proof Since $\rho(g) > 0$, i) and ii) are a direct consequence DCA convergence properties i) and ii) for a general DC program (see Sect. 3), while iii) is, like iii) of Theorem 1, due to the subanalytic data.

iv) it follows from the equivalence between EiGP and (2) (see Sect. 2). □

5 Computational experiments

5.1 Data and parameters

DC algorithms for solving the Symmetric Eigenvalue Complementarity Problem were implemented in C++. The performance of the DCA is compared with C++ implementation of Spectral Projected Gradient Algorithm (SPGA) [15]. All computational experiments were run on a Pentium IV 3 GHz of 1.00 GB RAM.

Except for the experiment concerning the effect of ten different initial points (Table 8), the procedures presented in [34] have been used to find an initial point for DCA and SPGA. In fact, the vectors $\mathbf{x} = (1/n)\mathbf{e}$ and $\mathbf{x} = \mathbf{e}^1$ were used as the initial solution of SPGA and DCA, respectively. These are the choices for which the corresponding algorithms have the best performance.

The DCA also requires a suitable value for the parameters $\mu, \nu \geq 0$. For each experiment, a suitable value has been used.

The first experiments have been carried out on two different types of matrices. For the matrix A we have defined two alternative choices $A1$ and $A2$ as follows:

- Given the $n \times n$ lower-triangular matrix M_n

$$M_n = \begin{cases} 1: & \text{if } i = j, \\ 0: & \text{if } i < j, \\ 2: & \text{if } i > j. \end{cases}$$

  Then $A1 := (M_n)(M_n)^T$. Hence $A \in SPD$.
- The symmetric positive definite matrix $A2$ is defined as follows

$$\begin{pmatrix} 6 & -4 & 1 & 0 & 0 & 0 & \cdots & 0 \\ -4 & 6 & -4 & 1 & 0 & 0 & \cdots & 0 \\ 1 & -4 & 6 & -4 & 1 & 0 & \cdots & 0 \\ 0 & 1 & -4 & 6 & -4 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 1 & -4 & 6 \end{pmatrix}$$

  In all of our experiments B is the identity matrix.

5.2 Computational results

The performance of the DCA algorithms presented in Sect. 4 is measured by the CPU time taken to solve the non convex optimization problems and the quality of the provided solutions. Efficiency of the DCA is compared with SPGA. This latter algorithm has shown to be much more efficient than interior-point (IP) and active-set methods for solving large-scale EiCPs [15]. We also believe that SPGA is more efficient than the semi-smooth Newton (SSN) algorithm discussed in [1], as this latter method shares essentially the same benefits and drawbacks of IP methods for solving large-scale complementarity problems. In fact, IP and SSN algorithms are usually fast to converge but require the solution of a large linear system of equations in each iteration. In order to have a comparable quality of the optimal eigenvalues and preventing premature convergence, the precision ε has been set equal to 10^{-6} for DCA and 10^{-9} for SPGA. If we take $\varepsilon = 10^{-6}$ for SPGA, it terminates before getting a good solution and the quality of the corresponding eigenvalue is not good.

In Algorithm 1 (DCA applied to the quadratic formulation of EiCP), we need to solve a quadratically constrained quadratic programming problem at each iteration. The standard solver CPLEX version 10.1 has been used to solve this subproblem. In the Algorithms 2 and 3, a C++ implementation of the Block Pivotal Principle Pivoting Algorithm [15] has been used to find the solution of the strictly convex separable quadratic program on the simplex at each iteration. The Block Pivotal Principle Pivoting Algorithm is a simple algorithm that is quite efficient to our purpose.

The following tables present the optimal eigenvalues (λ^*), number of iterations (iter.), the suitable value of μ (and ν for the generalized Rayleigh formulation of EiCP) for DCA, and CPU time in second taken to solve the different formulations of EiCP (by DCA or by SPGA).

Table 2 reports the performance of DCA to solve the logarithmic formulation of EiCP for the pairs $(A, B) := (Ai, B)$ for $i = 1, 2$ and different dimensions.

Table 2 Results obtained by DCA (Algorithm 2) and SPGA for solving the logarithmic formulation of EiCP with $(A, B) := (Ai, I)$ for $i = 1, 2$

		SPGA			DCA			
A, B	Dim	λ^*	CPU	iter.	λ^*	CPU	iter.	μ
$A1, I$	100	16210.722642	0.000000	7	16210.719322	0.000000	21	100
$A1, I$	200	64844.890473	0.000000	7	64844.858223	0.000000	30	200
$A1, I$	300	145901.836816	0.000000	7	145901.793809	0.000000	40	300
$A1, I$	400	259381.561669	0.000000	7	259381.473284	0.000000	49	400
$A1, I$	500	405284.065028	0.000000	7	405283.923758	0.000000	58	500
$A1, I$	600	583609.346892	0.000000	7	583609.149952	0.000000	67	600
$A1, I$	700	794357.407263	0.000000	7	794357.160989	0.000000	76	700
$A1, I$	800	1037528.246137	0.000000	7	1037527.974410	0.000000	85	800
$A1, I$	900	1313121.863514	0.000000	7	1313121.600676	0.000000	94	900
$A1, I$	1000	1621138.259395	0.000000	7	1621137.550826	0.016000	102	1000
$A2, I$	100	7.985417	0.000000	181	7.995345	0.000000	927	20
$A2, I$	200	7.996202	0.062000	634	7.996045	0.015000	996	20
$A2, I$	300	7.998274	0.125000	1043	7.996045	0.031000	996	20
$A2, I$	400	7.998968	0.219000	1520	7.996045	0.047000	996	20
$A2, I$	500	7.999279	0.250000	1422	7.996045	0.047000	996	20
$A2, I$	600	7.999305	0.281000	1346	7.996045	0.047000	996	20
$A2, I$	700	7.999356	0.437000	1744	7.996045	0.063000	996	20
$A2, I$	800	7.999418	0.594000	2132	7.996045	0.093000	996	20
$A2, I$	900	7.999467	0.797000	2527	7.996045	0.078000	996	20
$A2, I$	1000	7.999518	1.140000	3256	7.996045	0.078000	996	20
$A2, I$	1500	7.999496	2.125000	3878	7.996045	0.125000	996	20
$A2, I$	2000	7.999509	3.562000	4554	7.996045	0.172000	996	20
$A2, I$	3000	7.999522	5.890000	4736	7.996045	0.266000	996	20
$A2, I$	4000	7.999540	10.437000	5666	7.995481	0.312000	932	20
$A2, I$	10000	7.999596	38.375000	7866	7.994835	0.703000	829	19
Range	–	–	0.0–38.4	7–7866	–	0.0–0.7	21–996	–

The second set of experiments displayed in Table 3 corresponds to the results provided by DCA for solving the Rayleigh Quotient with the matrices defined by the pairs $(A, B) := (Ai, I)$, $i = 1, 2$.

According to Tables 2 and 3, the performance of SPGA and DCA are comparable in case $A = A1$. In fact, the number of iterations of DCA is greater than that of SPGA but the CPU time remains almost the same. When $A = A2$, DCA is more efficient and more robust than SPGA. The CPU time of DCA remains less than one second even for the large dimensions, which is not the case for SPGA. It is also interesting to note that the precision of the eigenvalues found by the two formulations are the same when DCA is employed, but this no longer holds for SPGA. So DCA seems to be more robust and to provide more accurate eigenvalues than SPGA. It is showed in [15] that SPGA has a better performance in comparison with the commercial softwares LOGO

Table 3 Results obtained by DCA (Algorithm 3) and SPGA for solving the Rayleigh Quotient formulation of EiCP with $(A, B) := (Ai, I)$ for $i = 1, 2$. The suitable value of ν for these matrices is 0.0

		SPGA			DCA			
A, B	Dim	λ^*	CPU	iter.	λ^*	CPU	iter.	μ
$A1, I$	100	16210.722715	0.000000	8	16210.719322	0.000000	21	100
$A1, I$	200	64844.890838	0.000000	8	64844.858223	0.000000	30	200
$A1, I$	300	145901.837710	0.000000	8	145901.793809	0.000000	40	300
$A1, I$	400	259381.563329	0.000000	8	259381.473284	0.000000	49	400
$A1, I$	500	405284.067694	0.000000	8	405283.923758	0.000000	58	500
$A1, I$	600	583609.350804	0.000000	8	583609.149952	0.000000	67	600
$A1, I$	700	794357.412661	0.000000	8	794357.160989	0.000000	76	700
$A1, I$	800	1037528.253263	0.000000	8	1037527.974410	0.000000	85	800
$A1, I$	900	1313121.872611	0.000000	8	1313121.600676	0.000000	94	900
$A1, I$	1000	1621138.270704	0.000000	8	1621137.550826	0.015000	102	1000
$A2, I$	100	7.996204	0.047000	615	7.995345	0.000000	927	20
$A2, I$	200	7.999006	0.141000	1581	7.996045	0.016000	996	20
$A2, I$	300	7.999507	0.266000	2331	7.996045	0.015000	996	20
$A2, I$	400	7.999509	0.281000	1945	7.996045	0.032000	996	20
$A2, I$	500	7.999524	0.438000	2426	7.996045	0.047000	996	20
$A2, I$	600	7.999610	0.735000	3399	7.996045	0.062000	996	20
$A2, I$	700	7.999638	1.062000	4300	7.996045	0.062000	996	20
$A2, I$	800	7.999637	1.328000	4640	7.996045	0.062000	996	20
$A2, I$	900	7.999633	1.578000	4868	7.996045	0.078000	996	20
$A2, I$	1000	7.999640	1.922000	5273	7.996045	0.094000	996	20
$A2, I$	1500	7.999619	3.062000	5333	7.996045	0.141000	996	20
$A2, I$	2000	7.999652	5.782000	7078	7.996045	0.172000	996	20
$A2, I$	3000	7.999651	9.516000	7541	7.996045	0.281000	996	20
$A2, I$	4000	7.999654	15.046000	8130	7.995481	0.344000	932	20
$A2, I$	5000	7.999666	18.953000	8745	7.995480	0.406000	922	20
$A2, I$	10000	7.999683	43.578000	9777	7.994835	0.734000	829	19
Range	–	–	0.0–43.6	8–9777	–	0.0–0.73	21–996	–

[44] and MINOS [27], so we may conclude that DCA is better than the mentioned codes for solving EiCP.

Table 4 includes the numerical results of DCA applied to the quadratically constrained formulation of EiCP (Algorithm 1) with $(A, B) := (A1, I)$ and $(A, B) := (A2, I)$. We observe, from this table, that the CPU time of Algorithm 1 is dramatically greater than that of Algorithms 2 and 3. This is due to the solution of quadratically constrained quadratic programming subproblems by CPLEX solver as the operations of writing the subproblem in CPLEX format as well as of solving it are quite time consuming. It seems that the use of an efficient algorithm to solve the quadratically constrained formulation of EiCP can decrease drastically the CPU time of Algorithm 1. Another issue of Algorithm 1 is the fact that it is quite sensitive to

Table 4 Results obtained by DCA (Algorithm 1) for solving the Quadratically constrained formulation of EiCP with $(A, B) := (Ai, I)$ for $i = 1, 2$

		DCA			
A, B	Dim	λ^*	CPU	iter.	μ
$A1, I$	100	16210.721961	0.094000	5	50
$A1, I$	200	64844.890796	0.235000	6	50
$A1, I$	300	145901.820956	0.250000	5	50
$A1, I$	400	259381.455304	0.797000	11	50
$A1, I$	500	405283.708020	0.422000	5	50
$A1, I$	600	583609.070680	0.719000	5	100
$A1, I$	700	794357.217068	0.453000	4	110
$A1, I$	800	1037527.982273	0.516000	4	121
$A1, I$	900	1313121.723681	0.609000	4	125
$A1, I$	1000	1621137.119452	0.640000	4	124
$A2, I$	100	7.993198	28.813000	1438	5
$A2, I$	200	7.993541	60.047000	1500	5
$A2, I$	300	7.993333	100.125000	1458	5
$A2, I$	400	7.990916	84.547000	1071	5
$A2, I$	500	7.993522	137.671000	1500	5
$A2, I$	600	7.992261	147.953000	1257	5
$A2, I$	700	7.992252	166.078000	1256	5
$A2, I$	800	7.993525	232.562000	1500	5
$A2, I$	900	7.992216	209.156000	1249	5
$A2, I$	1000	7.993336	279.922000	1460	5
Range	–	–	0.09–280	4–1500	–

the value of μ. This does not occur with Algorithms 2 and 3, which seem to be less dependent on this value.

In order to evaluate and compare the efficiency of the algorithms for more general cases we generated some random matrices $A3$. These matrices have different dimensions ranging from 100 to 1000, and were generated by using the *rand*() function of C++ such that their elements are uniformly distributed in the interval $[-6, 6]$. Because of the presence of logarithmic function, we have to construct the matrices so that the logarithmic terms are well-defined. To this aim, we added some multiplications of the identity matrix to the generated matrices. As before, B is the identity matrix. The numerical results of this experiment are reported in Tables 5 and 6 and confirm the performance of DCA for the structured matrices mentioned before.

In the last experiment we are interested in the effect of the initial point of the algorithms on the solution of the EiCP found by the algorithms. To this aim, we consider the matrix $A2$ with Dimensions 100, 500,1000 and apply DCA and SPGA to solve the logarithmic formulation of EiCP from ten different initial points. The notations of the 10 different initial points are presented in Table 7. The numerical results are presented in Table 8. Here "N. diff. λ^* " stands for the number of different eigenvalues provided by each algorithm, and we say that the two eigenvalues λ^* are different if the absolute value of their difference is greater than 10^{-6}.

Table 5 Results obtained by DCA (Algorithm 2) and SPGA for solving the logarithmic formulation of EiCP with $(A, B) := (A3, I)$

		SPGA			DCA			
A, B	Dim	λ^*	CPU	iter.	λ^*	CPU	iter.	μ
$A3, I$	100	44.257826	0.062000	189	46.160388	0.047000	336	150
$A3, I$	200	71.889149	0.063000	72	71.886184	0.078000	193	200
$A3, I$	300	88.102569	0.172000	66	88.101715	0.156000	155	300
$A3, I$	400	103.172541	0.532000	119	103.171000	0.484000	279	400
$A3, I$	500	120.084353	1.063000	162	120.331321	1.640000	611	400
$A3, I$	600	129.614127	1.203000	125	129.612462	1.157000	303	400
$A3, I$	700	138.074630	5.781000	456	137.923316	2.187000	422	500
$A3, I$	800	144.952300	2.078000	132	144.772355	2.985000	447	500
$A3, I$	900	159.344141	3.469000	172	159.341712	3.453000	409	600
$A3, I$	1000	160.563726	9.954000	400	160.518692	4.625000	447	650
Range	–	–	0.06–9.95	66–456	–	0.05–4.63	155–611	–

Table 6 Results obtained by DCA (Algorithm 3) and SPGA for solving the Rayleigh Quotient formulation of EiCP with $(A, B) := (A3, I)$

		SPGA			DCA			
A, B	Dim	λ^*	CPU	iter.	λ^*	CPU	iter.	μ
$A3, I$	100	44.257824	0.031000	150	46.160388	0.047000	336	150
$A3, I$	200	71.889166	0.078000	68	71.886184	0.078000	193	200
$A3, I$	300	88.102574	0.172000	79	88.101715	0.156000	155	300
$A3, I$	400	103.172538	0.484000	118	103.171000	0.484000	279	400
$A3, I$	500	120.336075	2.328000	369	120.331321	1.640000	611	400
$A3, I$	600	129.614267	0.609000	72	129.612462	1.157000	303	400
$A3, I$	700	138.074658	5.766000	486	137.923316	2.187000	422	500
$A3, I$	800	144.952199	2.093000	136	144.772355	2.985000	447	500
$A3, I$	900	159.344101	2.781000	150	159.341712	3.453000	409	600
$A3, I$	1000	160.563778	10.391000	435	160.518692	4.625000	447	650
Range	–	–	0.03–10.39	68–486	–	0.05–4.63	155–611	–

We observe from the numerical results in Table 8 that:

- DCA provides 6 different constrained eigenvalues while the number of different constrained eigenvalues provided by SPGA varies from 4 to 9;
- Even if the obtained λ^* is the same for different initial points, the corresponding eigenvectors are not necessary the same and in some cases they are completely different;
- The starting point $(1/n)e^1$ was chosen as an infeasible point to the problem but both of the algorithms give a feasible eigenvector.

Table 7 The notation of the initial points

Initial point	Description
$\mathbf{e}^i$	i-th canonical basic vector i.e., $\mathbf{e}^i = (0, \ldots, 1, \ldots, 0)^T$ in which 1 is located at the i-th position
$\mathbf{e}^{i,j}$	the n-vector for which all elements are 0 unless the i-th and j-th elements which are equal to $\frac{1}{2}$
$\mathbf{e}^{i,j,k}$	the n-vector for which all elements are 0 unless the i-th, j-th, and k-th elements which are equal to $\frac{1}{3}$
$(1/n)\mathbf{e}$	the n-vector $\frac{1}{n}(1, \ldots, 1)$
$(1/n)\mathbf{e}^1$	the n-vector $(\frac{1}{n}, 0, \ldots, 0)$

Table 8 Results obtained by DCA (Algorithm 2) and SPGA for solving the logarithmic formulation of EiCP with $(A, B) := (A2, I)$ and different initial points

A, B	n	Initial point	SPGA		DCA	
			λ^*	N. diff. λ^*	λ^*	N. diff. λ^*
$A2, I$	100	$\mathbf{e}^1$	7.996204	4	7.995345	6
$A2, I$	100	$\mathbf{e}^{50}$	7.996205		7.995870	
$A2, I$	100	$\mathbf{e}^{100}$	7.996204		7.995345	
$A2, I$	100	$\mathbf{e}^{1,2}$	7.996204		7.995352	
$A2, I$	100	$\mathbf{e}^{99,100}$	7.996204		7.995352	
$A2, I$	100	$\mathbf{e}^{1,2,3}$	7.996204		7.995345	
$A2, I$	100	$\mathbf{e}^{1,50,100}$	7.996205		7.995868	
$A2, I$	100	$\mathbf{e}^{98,99,100}$	7.996204		7.995345	
$A2, I$	100	$(1/n)\mathbf{e}$	7.985417		7.985249	
$A2, I$	100	$(1/n)\mathbf{e}^1$	7.996203		7.995348	
$A2, I$	500	$\mathbf{e}^1$	7.999527	8	7.996045	6
$A2, I$	500	$\mathbf{e}^{250}$	7.999682		7.997233	
$A2, I$	500	$\mathbf{e}^{500}$	7.999547		7.996045	
$A2, I$	500	$\mathbf{e}^{1,2}$	7.999547		7.996044	
$A2, I$	500	$\mathbf{e}^{499,500}$	7.999539		7.996044	
$A2, I$	500	$\mathbf{e}^{1,2,3}$	7.999552		7.996045	
$A2, I$	500	$\mathbf{e}^{1,250,500}$	7.999650		7.997205	
$A2, I$	500	$\mathbf{e}^{498,499,500}$	7.999561		7.996045	
$A2, I$	500	$(1/n)\mathbf{e}$	7.999279		7.988518	
$A2, I$	500	$(1/n)\mathbf{e}^1$	7.999552		7.996041	
$A2, I$	1000	$\mathbf{e}^1$	7.999644	9	7.996045	6
$A2, I$	1000	$\mathbf{e}^{500}$	7.999742		7.997233	
$A2, I$	1000	$\mathbf{e}^{1000}$	7.999612		7.996045	
$A2, I$	1000	$\mathbf{e}^{1,2}$	7.999612		7.996044	
$A2, I$	1000	$\mathbf{e}^{999,1000}$	7.999630		7.996044	
$A2, I$	1000	$\mathbf{e}^{1,2,3}$	7.999627		7.996045	
$A2, I$	1000	$\mathbf{e}^{1,500,1000}$	7.999749		7.997205	
$A2, I$	1000	$\mathbf{e}^{998,999,1000}$	7.999616		7.996045	
$A2, I$	1000	$(1/n)\mathbf{e}$	7.999518		7.988518	
$A2, I$	1000	$(1/n)\mathbf{e}^1$	7.999626		7.996041	

6 Conclusion

In this paper, we present a new approach based on DC programming and DCA for solving the Symmetric Eigenvalue Complementarity Problem (EiCP). We develop DCA to solve different formulations of EiCP. The nice effect of DC decompositions are well exploited in our approaches, especially in Algorithms 2 and 3 whose main work per iteration relies on the computation of a projection of a point onto the standard $n-1$ simplex. Computational experiments show that DCA is quite efficient and in most of the cases DCA outperforms SPGA, one among the best existing algorithms for EiCP. We intend to investigate in a near future the application of DCA to the asymmetric EICP [16]. On the other hand, the design of an efficient procedure for solving the quadratically constrained quadratic programming subproblems required by Algorithm 1 is also one of our current research areas.

References

1. Le Thi, H.A., Pham Dinh, T.: A nonsmooth algorithm for cone-constrained eigenvalue problems. Comput. Optim. Appl. (2010, to appear)
2. Le Thi, H.A., Pham Dinh, T.: Large scale molecular optimization from distance matrices by a DC optimization approach. SIAM J. Control Optim. **14**(1), 77–116 (2003)
3. Le Thi, H.A., Pham Dinh, T., Yen, N.D.: Properties of two DC Algorithms for quadratic programming. J. Glob. Optim. (2010, in press)
4. Auchmuty, G.: Unconstrained variational principles for eigenvalues of real symmetric matrices. SIAM J. Math. Anal. **20**(5), 1186–1207 (1989)
5. Auchmuty, G.: Globally and rapidly convergent algorithms for symmetric eigenproblems. SIAM J. Matrix Anal. Appl. **12**(4), 690–706 (1991)
6. Chatelin, F.: Eigenvalues of Matrices. Wiley, New York (1993)
7. Chung, S.: NP-completeness of the linear complementarity problem. J. Optim. Theory Appl. **60**(3), 393–399 (1989)
8. Costa, A.P., Seeger, A.: Numerical solution of cone-constrained eigenvalue problems. Comput. Optim. Appl. **28**(1), 37–61 (2009)
9. Costa, A.P., Figueiredo, I.N., Judice, J., Martins, J.A.C.: A complementarity eigenproblem in the stability analysis of finite dimensional elastic systems with frictional contact. In: Ferris, M., Pang, J.S., Mangasarian, O. (eds.) Complementarity: Applications, Algorithms and Extensions, pp. 67–83. Kluwer Academic, New York (2001)
10. Golub, G.H., van der Vorst, H.: Eigenvalue computation in the 20th century. J. Comput. Appl. Math. **123**(1–2), 35–65 (2000)
11. Harrington, J.E., Hobbs, B.F., Pang, J.S., Liu, A., Roch, G.: Collusive game solutions via optimization. Math. Program. **104**(1–2), 407–435 (2005)
12. Horn, R.A., Johnson, C.R.: Matrix Analysis. Cambridge University Press, Cambridge (1990)
13. Jacobi, C.G.J.: Uber ein leichtes verfahren die in der theorie der sacularstorungen vorkommenden gleichungen numerisch aufzulosen. J. Reine Angew. Math. **30**, 51–94 (1846)
14. Judice, J., Sherali, H.D., Ribeiro, I.: The eigenvalue complementarity problem. Comput. Optim. Appl. **37**, 139–156 (2007)
15. Judice, J., Raydan, M., Rosa, S., Santos, S.: On the solution of the symmetric eigenvalue complementarity problem by the spectral projected gradient algorithm. Numer. Algorithms **47**, 391–407 (2008)
16. Judice, J., Sherali, H.D., Ribeiro, I., Rosa, S.: On the asymmetric eigenvalue complementarity problem. Optim. Methods Softw. **24**, 549–568 (2009)
17. Kanno, Y., Ohsaki, M.: Optimization-based stability analysis of structures under unilateral constraints. Int. J. Numer. Methods Eng. **77**, 90–125 (2009)
18. Kressner, D.: Numerical Methods for General and Structured Eigenvalue Problems. Springer, Berlin (2005)
19. Lavilledieu, P., Seeger, A.: Existence de valeurs propres pour les systèmes multivoques: résultats anciens et nouveaux. Ann. Sci. Math. Qué. **2001**, 47–70 (2001)

20. Le Thi, H.A.: Contribution à l'optimisation non convexe et l'optimisation globale: Théorie, Algorithmes et Applications. Habilitation à Diriger des Recherches, Université de Rouen (1997)
21. Le Thi, H.A., Pham Dinh, T.: A continuous approach for globally solving linearly constrained quadratic zero-one programming problems. Optimization **50**(1–2), 93–120 (2001)
22. Le Thi, H.A., Pham Dinh, T.: The DC (Difference of Convex functions) programming and DCA revisited with DC models of real world non convex optimization problems. Ann. Oper. Res. **133**, 23–46 (2005)
23. Le Thi, H.A., Huynh, V.N., Pham Dinh, T.: Convergence Analysis of DC Algorithm for DC programming with subanalytic data. Ann. Oper. Res. Technical Report, LMI, INSA-Rouen, August 2009
24. Liu, Y., Shen, X., Doss, H.: Multicategory ψ-learning and support vector machine: Computational tools. J. Comput. Graph. Stat. **14**, 219–236 (2005)
25. Matrix market. A visual repository of test data for use in comparative studies of algorithms for numerical linear algebra
26. Mongeau, M., Torki, M.: Computing eigenelements of real symmetric matrices via optimization. Comput. Optim. Appl. **29**, 263–287 (2004)
27. Murtagh, B.A., Saunders, M.A.: Minos 5.1 user's guide. Technical Report 83-20R, Department of Operations Research, Stanford University (1987)
28. Murty, K.G.: Linear Complementarity, Linear and Nonlinear Programming. Heldermann, Berlin (1988)
29. Naumann, J., Wenk, H.U.: On eigenvalue problems for variational inequalities. An application to nonlinear plate buckling. Rend. Mat. **9**(6), 439–463 (1976)
30. Neumann, J., Schnörr, C., Steidl, G.: Combined SVM-based feature selection and classification. Mach. Learn. **61**, 129–150 (2005)
31. Parlett, B.N.: The Symmetric Eigenvalue Problem. Classics in Applied Mathematics, vol. 20. SIAM, Philadelphia (1997)
32. Pham Dinh, T., Le Thi, H.A.: Convex analysis approach to d.c. programming: Theory, algorithms and applications. Acta Math. Vietnam. **22**(1), 289–355 (1997). Dedicated to Professor Hoang Tuy on the occasion of his 70th birthday
33. Pham Dinh, T., Le Thi, H.A.: DC optimization algorithms for solving the trust region subproblem. SIAM J. Control Optim. **8**, 476–505 (1998)
34. Queiroz, M., Judice, J., Humes, C.: The symmetric eigenvalue complementarity problem. Math. Comput. **73**(248), 1849–1863 (2004)
35. Rockafellar, R.T.: Convex Analysis, 1st edn. Princeton University Press, Princeton (1970)
36. Ronan, C., Fabian, S., Jason, W., Léon, B.: Trading convexity for scalability. In: Proceedings of the 23rd International Conference on Machine Learning ICML 2006, Pittsburgh, Pennsylvania, pp. 201–208 (2006)
37. Saad, Y.: Numerical methods for large eigenvalue problems. Manchester University Press, Manchester (1992)
38. Schnörr, C.: Signal and image approximation with level-set constraints. Computing **81**, 137–160 (2007)
39. Seeger, A.: Eigenvalue analysis of equilibrium processes defined by linear complementarity conditions. Linear Algebra Appl. **292**, 1–14 (1999)
40. Sriperumbudur, B.K., Torres, D.A., Lanckriet, G.R.G.: Sparse eigen methods by DC programming. In: ICML (2007)
41. Pham Dinh, T., Le Thi, H.A., Akoa, F.: Combining DCA and interior point techniques for large-scale nonconvex quadratic programming. Optim. Methods Softw. **23**(4), 609–629 (2008)
42. Thiao, M., Pham Dinh, T., Le Thi, H.A.: A DC programming approach for sparse eigenvalue problem. In: ICML (2010)
43. van der Vorst, H., Golub, G.H.: 150 years old and still alive: eigenproblems. In: The State of the Art in Numerical Analysis. Inst. Math. Appl. Conf. Ser. New Ser, vol. 63, pp. 93–119. Oxford University Press, New York (1997)
44. Vanderbei, R.J.: LOQO user's manual, version 3.10. Technical Report SOR-97-08, Princeton University (2003)
45. Weber, S., Schnörr, C., Schüle, T., Hornegger, J.: Binary tomography by iterating linear programs. In: Klette, R., Kozera, R., Noakes, L., Weickert, J. (eds.) Computational Imaging and Vision—Geometric Properties from Incomplete Data. Kluwer Academic, Dordrecht (2005)
46. Zhou, Y., Gowda, M.S.: On the finiteness of the cone spectrum of certain linear transformations on Euclidean Jordan algebras. Linear Algebra Appl. **431**(5–7), 772–782 (2009)