Yokohama Publishers
ISSN 1349-8169 ONLINE JOURNAL

Pacific Journal of Optimization
Vol. 19, No. 4, pp. 579–606, 2023

THE SEQUENTIAL QUADRATIC PROGRAMMING FOR SYMMETRIC PARETO EIGENVALUE COMPLEMENTARITY PROBLEM*

Lin Zhu, Yuan Lei† and Jiaxin Xie

Abstract: In this article, a sequential quadratic programming (SQP) algorithm framework is proposed to solve the symmetric Pareto Eigenvalue Complementarity Problem (EiCP). In the algorithm framework, the search direction can be constructed by solving a quadratic programming subproblem in each iteration, and the precise step size can be obtained to ensure the reduction of the penalty function. Moreover, the global convergence of this algorithm is discussed. In order to improve the computational efficiency, a simple SQP algorithm is recommended to the large symmetric Pareto EiCP. Various numerical results of the symmetric Pareto EiCPs are reported to illustrate the efficiency of the proposed algorithms.

Key words: *pareto eigenvalue complementarity problem, sequential quadratic programming, quadratic programming subproblem, global convergence*

Mathematics Subject Classification: *65F15, 90C20, 90C33*

1 Introduction

The Eigenvalue Complementarity Problem (**EiCP**), arising in various areas of applied mathematics [25, 23, 24], is also called the cone-constrained eigenvalue problem. It reduces to the classical eigenvalue problem when the cone $\mathbb{K}$ coincides with the whole space $\mathbb{R}^n$. The Pareto **EiCP** corresponds to the case when the cone $\mathbb{K}$ coincides with the nonnegative cone $\mathbb{R}^n_+$, which consists in finding a scalar $\lambda \in \mathbb{R}$ and a vector $x \in \mathbb{R}^n_+ \setminus \{0\}$ such that

$$x \geq 0, \quad (A - \lambda B)x \geq 0, \quad x^T(A - \lambda B)x = 0, \tag{1.1}$$

where $A, B \in \mathbb{R}^{n\times n}$ are given matrices and B is positive definite. A wide variety of applications in science and engineering require the numerical calculation of Pareto **EiCP**, such as dynamic analysis of structural mechanics systems, frictional contact problems in mechanics, and so on [15, 16, 19, 20, 28, 13]. In the recent literature [27], it is proposed that the similarity between connected graphs can be measured by the complementarity eigenvalues of the adjacency matrix. Two distinct graphs G and H are viewed as highly similar if they share a large number of complementarity eigenvalues.

*The work was supported by National Natural Science Foundations of China (Nos. 11871205, 2001026, 12071019, 12126608) and the Fundamental Research Funds for the Central Universities.

†Corresponding author.

The scalar λ and the nonzero vector x satisfying system (1.1) are respectively called a Pareto eigenvalue and an associated Pareto eigenvector of the given matrix pair (A, B). The set

$$\sigma(A, B) = \{\lambda \in \mathbb{R} \mid (\lambda, x) \text{ solves (1.1) for some nonzero vectors } x \in \mathbb{R}^n_+\}$$

consisting of all Pareto eigenvalues is called the Pareto spectrum of (A, B). It is well known that Pareto **EiCP** always has a solution since it is equivalent to a Variational Inequality Problem with a continuous function on the ordinary simplex [5]. The number of Pareto eigenvalues may grow exponentially with the order of the problem [25, 21, 3, 26]. Therefore, it is not easy to solve the Pareto spectrum for medium and large scale instances. Similar to linear or nonlinear complementarity problems, the Pareto **EiCP** can be reformulated as an equivalent system of nonsmooth equations by using nonlinear complementarity functions (NCP- functions), and then semismooth Newton's method can be applied to solve Pareto **EiCP** [2, 1]. This kind of Newton-type method is efficient for solving the Pareto **EiCP** in general. However, these algorithms may fail in many instances.

This paper is concerned with the symmetric Pareto **EiCP**, that is, the matrices A and B are symmetric, and B is positive definite. From a computational point of view, the symmetric Pareto **EiCP** is easier to solve. In addition, a large number of documents mention that each stationary point of the following nonlinear programming problem is the solution of symmetric Pareto **EiCP** [22, 14, 11, 4].

$$\begin{array}{ll} \min_{x\in\mathbb{R}^n} & f(x) = \frac{x^T Ax}{x^T Bx} \\ \text{s.t.} & x \in \Omega, \end{array} \tag{1.2}$$

where $\Omega = \{x \mid x \geq 0, e^T x = 1\}$ and $e \in \mathbb{R}^n$ is a vector of ones. Among them, the equality constraint $e^T x = 1$ is to ensure $x \neq 0$. As expected, a number of nonlinear programming algorithms [18] can be used to solve the symmetric Pareto **EiCP** by computing a stationary point of (1.2). In particular, some descent algorithms have been proposed for solving (1.2) by constructing a feasible descent direction [22, 11]. For example, the canonic steepest ascent method was slightly modified in [22] to solve the symmetric Pareto **EiCP** and the feasible direction at each step is the projection of the negative gradient on the nonnegative cone $\mathbb{R}^n_+$. The difference of convex function (DC) programming [14] may also be useful for computing a solution of the symmetric Pareto **EiCP** for large-scale instances. For the spectral projected gradient (SPG) method proposed in [11], the feasible descent direction with the spectral choice line search parameter is projected onto the compact set Ω in each step. Combining the idea of the SPG method and the block active set method described in [6], a block active set algorithm with spectral choice search direction (BAS) was proposed in [4]. Compared with the SPG algorithm, the BAS algorithm determines an exact optimal step size along the direction of the block active set algorithm in each iteration, and its efficiency has been proved by a large number of calculation results. Recently [10], a solution of symmetric Pareto **EiCP** can be computed as a stationary point $\bar{x}$ of the quadratic programming problem:

$$\begin{array}{ll} \min_{x\in\mathbb{R}^n} & f(x) = \frac{1}{2} x^T (A - \lambda B)x \\ \text{s.t.} & x \in \Omega, \end{array} \tag{1.3}$$

where $\lambda = \frac{\bar{x}^T A\bar{x}}{\bar{x}^T B\bar{x}}$. The alternating direction method of multipliers (ADMM) [10] and a sequential partial linearization (SPL) algorithm [7] are used to obtain an **EiCP** solution by solving the quadratic programming problem (1.3). In the numerical experiments in [10, 7], it can be found that the ADMM algorithm may be too slow for some instances, while the SPL algorithm performs better. In [12], a new sequential fractional linear quadratic program (FLQP) algorithm was proposed for computing a solution of symmetric **EiCP**, as this problem is equivalent to the computation of a stationary point of a fractional quadratic program on the simplex.

The sequential quadratic programming (SQP) method has proved highly effective for solving nonlinear programming problems with smooth nonlinear functions in the objective and constraints [18, 17, 9, 8]. The idea of the SQP method is to use a series of quadratic programming subproblems to solve the nonlinear programming problem, and the constraint condition of each quadratic programming subproblem is the linearization of the constraint condition in the original problem. In addition, the search direction can be constructed by solving the quadratic programming subproblem in each iteration. In this paper, we attempt to apply the SQP method for solving the symmetric Pareto **EiCP**, and make certain changes to the SQP method.

The structure of the paper is as follows. In Section 2, the detailed description of the SQP algorithm for solving the symmetric Pareto **EiCP** including the algorithm framework and important basic properties. At the end of this section, we will discuss the exact line search that ensures the reduction of the objective function. The global convergence of this algorithm is shown in Section 3. In Section 4, a simple SQP algorithm is introduced, and an effective method for solving the quadratic programming subproblem in each iteration is introduced. Numerical experiments are given in Section 5, and some conclusions and future work are introduced in the last section to conclude this article.
Notation : $\mathbb{S}^n$ represents the set of n order symmetric matrices, $\mathbb{S}^n_+$ represents the set of n order symmetric positive definite matrices.

$\boxed{2}$ The SQP Method for Symmetric Pareto EiCP

$\boxed{2.1}$ The sequential quadratic programming algorithm

In this section, we present an SQP algorithm framework for solving the symmetric Pareto **EiCP**. Currently, the nonlinear programming problem (1.2) is usually used as the optimization model for solving the symmetric Pareto **EiCP**. However, the objective function $f(x)$ in (1.2) is a generalized Rayleigh quotient. It is not an easy task to construct an appropriate quadratic programming subproblem in each iteration. In addition, the objective function in the optimization problem (1.3) contains the parameter λ, which is unknown at the beginning. To avoid the above problems, we consider the following quadratic programming problem:

$$\begin{array}{cl} \min_{x\in\mathbb{R}^n} & f(x) = \frac{1}{2}x^TAx \\ \text{s.t.} & \frac{1}{2}x^TBx = 1, \\ & x \geq 0. \end{array} \tag{2.1}$$

The Lagrangian function of the optimization problem (2.1) can be formulated as:

$$L(x, \lambda, z) = \frac{1}{2}x^TAx - \lambda(\frac{1}{2}x^TBx - 1) - z^Tx, \tag{2.2}$$

where the scalar $\lambda \in \mathbb{R}$ and the vector $z \in \mathbb{R}^n_+$ are respectively the Lagrange multipliers of the equality and inequality constraints in (2.1). Obviously, the stationary point of this problem satisfies the Karush-Kuhn-Tucker conditions:

$$\begin{cases} (A - \lambda B)x = z, \\ z \geq 0, x \geq 0, \\ z^Tx = 0, \\ \frac{1}{2}x^TBx - 1 = 0. \end{cases} \tag{2.3}$$

It follows from (2.3) that (λ, x) is a solution of the symmetric Pareto **EiCP**.

In this section, we introduce the SQP method to compute a stationary point of the quadratic programming problem (2.1). The key of Algorithm SQP is to construct a subproblem at the current point and obtain the search direction by solving the subproblem. Next, we construct the following subproblem at the current iteration point x_k:

$$\begin{array}{rc} \min_{d\in\mathbb{R}^n} & g_k(d) = \frac{1}{2}d^TMd + (Ax_k)^Td \\ \text{s.t.} & x_k^TBd + \frac{1}{2}x_k^TBx_k - 1 = 0, \\ & d + x_k \geq 0, \end{array} \tag{2.4}$$

where M is a symmetric positive definite matrix. Obviously, the quadratic programming subproblem (2.4) is strictly convex function. Generally speaking, the symmetric positive definite matrix M is usually an approximation to the Hessian of the Lagrangian function (2.2) in the sequential quadratic programming algorithm and the matrix M of each subproblem is different. The Lagrangian Hessian matrix of the problem (2.1) is $A - \lambda B$. Therefore, the most straightforward idea is that we can find an appropriate value $\lambda^{'}$ to approximate the Lagrangian multiplier λ. Choosing $\lambda^{'} = \frac{x_k^TAx_k}{x_k^TBx_k}$ is a good strategy, which is similar to Algorithm SPL. But it is difficult to guarantee that the matrix $M = A - \lambda^{'}B$ is positive definite. Solving non-convex subproblems becomes the hard part of the problem. In addition, if the matrix M is updated with iterations, we can hardly guarantee that it is bounded. We need a trade-off between a good update strategy and a good subproblem. In this paper, we fix the symmetric positive definite matrix M in the objective function (2.4) and analyze its convergence. This strategy not only reduces the difficulty of solving the subproblem, but also ensures that the matrix M must be bounded. We describe the selection of the matrix M in detail in Section 5 of the paper. Below, we present the basic properties of the quadratic programming subproblem (2.4).

Theorem 2.1. *Define the penalty function as*

$$P_\sigma(x) := f(x) + \sigma|\frac{1}{2}x^TBx - 1|, \tag{2.5}$$

where $\sigma > 0$ is the penalty parameter. If the current iteration point $x_k \geq 0$ and $x_k \neq 0$, then the quadratic programming subproblem (2.4) is well defined. Let d_k be the solution of (2.4), then we have

1. *d_k is a descent direction of the penalty function $P_\sigma(x)$ at x_k as long as $\sigma \geq |\lambda_k|$, where $\lambda_k \in \mathbb{R}$ is the Lagrange multiplier of the equality constraint in (2.4),*
2. *$x_k + \alpha d_k \geq 0$ for all $\alpha \in [0, 1]$,*
3. *if $d_k = 0$, then x_k is a stationary point of (2.1), and vice versa.*

Proof. If the feasible set is nonempty, there exists a unique solution of the quadratic programming subproblem (2.4) due to the quadratic programming subproblem (2.4) is strictly convex [18], that is, it is well defined.

Let $y_k := Bx_k$, since the current iteration point $x_k \geq 0$ and $x_k \neq 0$, then there exists at least one component $(y_k)_i > 0$, otherwise it contradicts the fact that $x_k^Ty_k = x_k^TBx_k > 0$. Define $\widetilde{d} \in \mathbb{R}^n$, where the elements satisfy

$$(\widetilde{d})_j := \begin{cases} \frac{\frac{1}{2}x_k^Ty_k+1}{(y_k)_j} - (x_k)_j, & j = i, \\ -(x_k)_j, & j \neq i. \end{cases}$$

It follows that $\widetilde{d} + x_k \geq 0$ and

$$\begin{aligned} x_k^T B\widetilde{d} = y_k^T \widetilde{d} &= \sum_{j=1}^{n} (y_k)_j (\widetilde{d})_j \\ &= \frac{1}{2}(x_k^T y_k + 1) - \sum_{j=1}^{n} (y_k)_j (x_k)_j \\ &= -\frac{1}{2} x_k^T y_k + 1 = -\frac{1}{2} x_k^T B x_k + 1, \end{aligned}$$

which means that $\widetilde{d}$ belongs to the feasible set of (2.4), i.e., the quadratic programming subproblem (2.4) is well defined.

(i) Let $\varphi(\alpha) := P_\sigma(x_k + \alpha d_k)$. From the continuity of the penalty function $P_\sigma(x)$ defined in (2.5) and the convexity of the absolute value function, we have

$$\begin{array}{rcl} \varphi^{'}(\alpha)|_{\alpha=0} & = & \lim_{\alpha\to 0^+} \frac{P_\sigma(x_k+\alpha d_k)-P_\sigma(x_k)}{\alpha} \\ & = & \lim_{\alpha\to 0^+} \frac{f(x_k+\alpha d_k)-f(x_k)+\sigma|\frac{1}{2}(x_k+\alpha d_k)^T B(x_k+\alpha d_k)-1|-\sigma|\frac{1}{2}x_k^T Bx_k-1|}{\alpha} \\ & \leq & (Ax_k)^T d_k + \sigma \lim_{\alpha\to 0^+} \frac{|\frac{1}{2}x_k^T Bx_k-1+\alpha x_k^T Bd_k|-|\frac{1}{2}x_k^T Bx_k-1|}{\alpha} \\ & = & (Ax_k)^T d_k + \sigma \lim_{\alpha\to 0^+} \frac{|\alpha(\frac{1}{2}x_k^T Bx_k-1+x_k^T Bd_k)+(1-\alpha)(\frac{1}{2}x_k^T Bx_k-1)|-|\frac{1}{2}x_k^T Bx_k-1|}{\alpha} \\ & = & (Ax_k)^T d_k + \sigma \lim_{\alpha\to 0^+} \frac{|(1-\alpha)(\frac{1}{2}x_k^T Bx_k-1)|-|\frac{1}{2}x_k^T Bx_k-1|}{\alpha} \\ & = & (Ax_k)^T d_k - \sigma \mid \frac{1}{2}x_k^T Bx_k - 1 \mid . \end{array} \tag{2.6}$$

The solution d_k satisfies the Karush-Kuhn-Tucker conditions:

$$\begin{cases} Md_k + Ax_k = \lambda_k Bx_k + z_k, \\ z_k \geq 0, \ d_k + x_k \geq 0, \\ z_k^T(d_k + x_k) = 0, \\ x_k^T Bd_k + \frac{1}{2}x_k^T Bx_k - 1 = 0, \end{cases} \tag{2.7}$$

where the scalar $\lambda_k \in \mathbb{R}$ and the vector $z_k \in \mathbb{R}^n_+$ are respectively the Lagrange multipliers of the equality and inequality constraints in (2.4). It follows from (2.7) that

$$\begin{aligned} (Ax_k)^T d_k &= -d_k^T M d_k + \lambda_k x_k^T B d_k + {z_k}^T d_k \\ &= -d_k^T M d_k - {z_k}^T x_k + \lambda_k (1 - \frac{1}{2} x_k^T B x_k). \end{aligned} \tag{2.8}$$

Substituting (2.8) into (2.6), we obtain

$$\begin{aligned} \varphi^{'}(\alpha)|_{\alpha=0} &\leq -d_k^T M d_k - {z_k}^T x_k - \lambda_k (\frac{1}{2} x_k^T B x_k - 1) - \sigma \mid \frac{1}{2} x_k^T B x_k - 1 \mid \\ &\leq -d_k^T M d_k - {z_k}^T x_k - (\sigma - \mid \lambda_k \mid) \mid \frac{1}{2} x_k^T B x_k - 1 \mid . \end{aligned} \tag{2.9}$$

Since the matrix M is symmetric positive definite and ${z_k}^T x_k \geq 0$, we have $\varphi^{'}(\alpha)|_{\alpha=0} \leq 0$ as long as $\sigma \geq |\lambda_k|$. Hence, d_k is a descent direction of the penalty function $P_\sigma(x)$ at x_k for given penalty parameter $\sigma \geq |\lambda_k|$.

(ii) Since the solution d_k satisfies the Karush-Kuhn-Tucker conditions (2.7), then the assertion clearly holds by $x_k \geq 0$ and $x_k + d_k \geq 0$.

(iii) If the solution $d_k = 0$, then the Karush-Kuhn-Tucker conditions (2.7) reduces to (2.3), which implies x_k is a stationary point of (2.1) and the corresponding Lagrange multiplier λ_k is a Pareto eigenvalue of (1.1).

Conversely, let x_k be a stationary point of (2.1), then there exist $\lambda_k \in \mathbb{R}$ and nonnegative vector $z_k \in \mathbb{R}^n$ satisfy the Karush-Kuhn-Tucker conditions (2.3), i.e.,

$$\begin{cases} Ax_k = \lambda_k Bx_k + z_k, \\ z_k \geq 0, x_k \geq 0, \\ z_k^T x_k = 0, \\ \frac{1}{2} x_k^T Bx_k - 1 = 0, \end{cases} \tag{2.10}$$

which implies $d_k = 0 \in \mathbb{R}^n$ is a Karush-Kuhn-Tucker solution of (2.7). Moreover, $\frac{1}{2} x_k^T Bx_k = 1$ yields that $d_k = 0$ is a feasible solution of (2.4). Hence $d_k = 0$ is also the unique solution of (2.4) due to the fact that the quadratic programming subproblem (2.4) is strictly convex. □

Theorem 2.1 shows that the solution of the quadratic programming subproblem (2.4) can be considered as a search direction in each iteration, and the solution of (2.1) may be found along this direction. Consequently, we can establish the SQP algorithm framework for solving symmetric Pareto **EiCP**.

SQP: Sequential Quadratic Programming Algorithm

1. **Initialization**

   (a) Give the symmetric matrix $A \in \mathbb{S}^n$, and $B \in \mathbb{S}^n_+$ is symmetric positive definite;

   (b) Choose a symmetric positive definite matrix $M \in \mathbb{S}^n_+$ and a penalty parameter $\sigma > 0$;

   (c) Choose a positive tolerance ϵ, the initial iteration point $x_0 \geq 0$ and $x_0 \neq 0$, and set $k = 0$.

2. **Iteration**

Step 1. Find the unique solution d_k of (2.4);

Step 2. If $\|d_k\|_2 \leq \epsilon$, then stop; otherwise, go to step 3;

Step 3. Compute a step size $\alpha_k \in [0, 1]$ such that

$$P_\sigma(x_k + \alpha_k d_k) = \min_{0 \leq \alpha \leq 1} P_\sigma(x_k + \alpha d_k);$$

Step 4. Let $x_{k+1} = x_k + \alpha_k d_k$, $k = k + 1$, and return to step 1.

For the sequence $\{x_k\}$ generated by the SQP algorithm, the assertion (2) in Theorem 2.1 shows the iteration point $x_k \geq 0$ for all k as long as the initial point $x_0 \geq 0$ and $x_0 \neq 0$. Since the penalty function $P_\sigma(x)$ is nearly quadratic, we attempt to compute the exact step size α_k in step 3 of the SQP algorithm at the end of this section.

$\boxed{2.2}$ The exact step size

For the iteration point x_k and the solution d_k of (2.4), we denote $\tau_k := 1 - \frac{1}{2}x_k^T B x_k$ and $c_k := \frac{1}{2}d_k^T B d_k$ for simplicity, then we have

$$\begin{aligned} \varphi(\alpha) &= P_\sigma(x_k + \alpha d_k) \\ &= f(x_k + \alpha d_k) + \sigma \mid \frac{1}{2}(x_k + \alpha d_k)^T B (x_k + \alpha d_k) - 1 \mid \\ &= \frac{1}{2}x_k^T A x_k + \frac{1}{2}\alpha^2 d_k^T A d_k + \alpha x_k^T A d_k + \sigma \mid c_k \alpha^2 + \tau_k \alpha - \tau_k \mid . \end{aligned} \tag{2.11}$$

Obviously, the function $\varphi(\alpha)$ is continuous, that is, the optimization problem in step 3 of the SQP algorithm is well defined.

Theorem 2.2. *Let x_k and d_k be generated by the SQP algorithm. Suppose $d_k \neq 0$ and $\sigma > \rho$, where ρ is the spectral radius of matrix $B^{-1}A$, then the exact step size α_k is selected by the following rules:*

(i) *when $\tau_k > 0$, the exact step size α_k in step 3 can be determined by*

$$\alpha_k = \begin{cases} 1, & \text{if } \ 1 \leq \delta_k, \\ \delta_k, & \text{if } \ \delta_k^1 < \delta_k < 1, \\ \delta_k^1, & \text{if } \ \delta_k \leq \delta_k^1, \end{cases} \tag{2.12}$$

(ii) *when $\tau_k \leq 0$, the exact step size α_k in step 3 can be determined by*

$$\alpha_k = \begin{cases} 1, & \text{if } \ 1 \leq \delta_k, \\ \delta_k, & \text{if } \ \delta_k < 1, \end{cases} \tag{2.13}$$

where

$$\delta_k = -\frac{x_k^T A d_k + \sigma \tau_k}{d_k^T (A + \sigma B) d_k}, \quad \delta_k^1 = \frac{-\tau_k + \sqrt{\tau_k^2 + 4c_k\tau_k}}{2c_k}. \tag{2.14}$$

Proof. Note $\varphi(\alpha)$ is also a piecewise smooth function which is determined by the quadratic function

$$q(\alpha) := c_k\alpha^2 + \tau_k\alpha - \tau_k. \tag{2.15}$$

(i) when $\tau_k > 0$, the quadratic equation $q(\alpha) = 0$ has two distinct roots:

$$\delta_k^1 = \frac{-\tau_k + \sqrt{\tau_k^2 + 4c_k\tau_k}}{2c_k} \quad \text{and} \quad \delta_k^2 = \frac{-\tau_k - \sqrt{\tau_k^2 + 4c_k\tau_k}}{2c_k}. \tag{2.16}$$

Obviously,

$$-\tau_k - \sqrt{\tau_k^2 + 4c_k\tau_k} < 0 = -\tau_k + \sqrt{\tau_k^2} < -\tau_k + \sqrt{\tau_k^2 + 4c_k\tau_k} \tag{2.17}$$

and

$$\begin{aligned} -\tau_k + \sqrt{\tau_k^2 + 4c_k\tau_k} &< -\tau_k + \sqrt{\tau_k^2 + 4c_k\tau_k + (2c_k)^2} \\ &= -\tau_k + \sqrt{(\tau_k + 2c_k)^2} \\ &= -\tau_k + (\tau_k + 2c_k) \\ &= 2c_k. \end{aligned} \tag{2.18}$$

It follows from (2.16), (2.17), (2.18) that $\delta_k^2 < 0 < \delta_k^1 < 1$ and the function $\varphi(\alpha)$ within the interval $[0,1]$ can be reformulated as

$$\varphi(\alpha) = \begin{cases} \frac{1}{2}\alpha^2 d_k^T(A+\sigma B)d_k + \alpha(x_k^T A d_k + \sigma\tau_k) + \frac{1}{2}x_k^T A x_k - \sigma\tau_k, & \delta_k^1 < \alpha \leq 1, \\ \frac{1}{2}\alpha^2 d_k^T(A-\sigma B)d_k + \alpha(x_k^T A d_k - \sigma\tau_k) + \frac{1}{2}x_k^T A x_k + \sigma\tau_k, & 0 \leq \alpha \leq \delta_k^1. \end{cases} \tag{2.19}$$

If the penalty parameter satisfies $\sigma > \rho$, then the matrices $A+\sigma B$ and $A-\sigma B$ are respectively positive definite and negative definite, which means that the function $\varphi(\alpha)$ within the interval $[0,\delta_k^1]$ is concave and within the interval $[\delta_k^1, 1]$ is convex. Since d_k is a descent direction, the function $\varphi(\alpha)$ is monotone decreasing in the interval $[0,\delta_k^1]$. Since $\delta_k = -\frac{x_k^T A d_k + \sigma\tau_k}{d_k^T(A+\sigma B)d_k}$ is the unique minimizer of the quadratic function

$$\frac{1}{2}\alpha^2 d_k^T(A+\sigma B)d_k + \alpha(x_k^T A d_k + \sigma\tau_k) + \frac{1}{2}x_k^T A x_k - \sigma\tau_k,$$

then we provide the formula (2.12) for α_k by comparing three scalars $1, \delta_k$ and δ_1.

(ii) when $\tau_k \leq 0$, we first discuss the quadratic function $q(\alpha)$ defined in (2.15) in two cases:

(A1) If $-4c_k < \tau_k \leq 0$, then $q(\alpha) \geq 0$ for all $\alpha \in \mathbb{R}$;

(A2) If $\tau_k \leq -4c_k$, then δ_k^1 and δ_k^2 in (2.16) are the solutions of the equation $q(\alpha) = 0$, and

$$\begin{aligned} -\tau_k - \sqrt{\tau_k^2 + 4c_k\tau_k} &> -\tau_k - \sqrt{\tau_k^2 + 4c_k\tau_k + (2c_k)^2} \\ &= -\tau_k - \sqrt{(\tau_k + 2c_k)^2} \\ &= -\tau_k - |\tau_k + 2c_k| \\ &= 2c_k. \end{aligned} \tag{2.20}$$

From the above inequalities (2.20) and (2.16), we can get $1 < \delta_k^2 \leq \delta_k^1$, which implies that $q(\alpha) \geq 0$ for all $\alpha \in [0,1]$.

Hence, the function $\varphi(\alpha)$ within the interval $[0,1]$ for the case that $\tau_k \leq 0$ can be reformulated as

$$\varphi(\alpha) = \frac{1}{2}\alpha^2 d_k^T(A+\sigma B)d_k + \alpha(x_k^T A d_k + \sigma\tau_k) + \frac{1}{2}x_k^T A x_k - \sigma\tau_k, \quad 0 \leq \alpha \leq 1. \tag{2.21}$$

Since the function $\varphi(\alpha)$ is convex, we can provide the formula (2.13) for α_k by comparing the scalar 1 with the minimizer δ_k. □

3 Global Convergence of the SQP Algorithm

In this section, we focus on the global convergence of the SQP algorithm. The convergence of the classical sequential quadratic programming algorithm has been discussed in much literature. However, the proposed algorithm is different from the classical sequential quadratic programming algorithm in the construction of the subproblems and the selection of the step size. The following theoretical proof is inspired by [29, Theorem 12.2.3].

Lemma 3.1. *Let the sequence $\{d_k\}$ be generated by the SQP algorithm and the penalty parameter σ satisfies $\sigma > \max\{|\lambda_k|, \rho\}$ for all k, where λ_k is the Lagrange multiplier of the equation (2.7), ρ is the spectral radius of matrix $B^{-1}A$. If there is an infinite set K with $||d_k||_2 \geq \eta > 0$ for any $k \in K$, then there exists a positive constant $\bar{\eta}$ such that*

$$\Delta P_k := P_\sigma(x_{k+1}) - P_\sigma(x_k) \leq -\bar{\eta}, \quad \forall k \in K.$$

Proof. From the second part of the article, we know that the matrices M and $A+\sigma B$ are symmetric positive definite matrices, the matrix $A - \sigma B$ is the negative definite matrix. Obviously, there are constants $m_1 > 0$, $m_2 > 0$, $m_3 > 0$, and $m_4 > 0$, such that the following inequality

$$\begin{aligned} d_k^T M d_k \geq m_1 ||d_k||_2^2 \geq m_1\eta^2, \quad d_k^T(A+\sigma B)d_k \leq m_2||d_k||_2^2, \\ d_k^T(A-\sigma B)d_k \leq -m_3||d_k||_2^2 < 0, \quad d_k^T B d_k \leq m_4||d_k||_2^2 \end{aligned} \tag{3.1}$$

holds for all $k \in K$.

Theorem 2.2 shows that we can make $P_\sigma(x_{k+1}) = P_\sigma(x_k + \alpha_k d_k) = \min_{0\leq\alpha\leq 1} P_\sigma(x_k + \alpha d_k)$ by choosing the exact step size α_k, so we investigate the upper bound of ΔP_k according to Theorem 2.2.

(i) When $\tau_k > 0$, we investigate the following three cases according to (2.12):

(B1) If $\alpha_k = 1$, then it follows from (2.19), (2.8) and $1 \leq \delta_k$ that

$$\begin{aligned} \Delta P_k &= \frac{1}{2}d_k^T(A+\sigma B)d_k + x_k^T A d_k - \sigma\tau_k \\ &\leq \frac{1}{2}x_k^T A d_k - \frac{1}{2}\sigma\tau_k \\ &= -\frac{1}{2}d_k^T M d_k - \frac{1}{2}z_k^T x_k - \frac{1}{2}(\sigma - \lambda_k)\tau_k \\ &\leq -\frac{1}{2}m_1\eta^2. \end{aligned} \tag{3.2}$$

(B2) If $\alpha_k = \delta_k$, then it follows from (2.19), (2.14) and (2.8) that

$$\begin{aligned} \Delta P_k &= \frac{1}{2}\delta_k^2 d_k^T(A+\sigma B)d_k + \delta_k(x_k^T A d_k + \sigma\tau_k) - 2\sigma\tau_k \\ &= -\frac{1}{2}\frac{(x_k^T A d_k + \sigma\tau_k)^2}{d_k^T(A+\sigma B)d_k} - 2\sigma\tau_k \\ &= \frac{1}{2}\delta_k(x_k^T A d_k + \sigma\tau_k) - 2\sigma\tau_k \\ &= -\frac{1}{2}\delta_k d_k^T M d_k - \frac{1}{2}\delta_k z_k^T x_k + \frac{1}{2}\delta_k(\sigma+\lambda_k)\tau_k - 2\sigma\tau_k \\ &\leq -\frac{1}{2}\delta_k d_k^T M d_k + \frac{1}{2}(\sigma+\lambda_k)\tau_k - 2\sigma\tau_k \\ &= -\frac{1}{2}\delta_k d_k^T M d_k - \frac{1}{2}(\sigma-\lambda_k)\tau_k - \sigma\tau_k \\ &\leq -\frac{1}{2}\delta_k m_1\eta^2 - \sigma\tau_k. \end{aligned} \tag{3.3}$$

(B3) If $\alpha_k = \delta_k^1$, then it follows from (2.19), (2.14) and (2.8) that

$$\begin{aligned}\Delta P_k &= \frac{1}{2}(\delta_k^1)^2 d_k^T(A-\sigma B)d_k + \delta_k^1(x_k^T A d_k - \sigma\tau_k)\\ &= \frac{1}{2}(\delta_k^1)^2 d_k^T(A-\sigma B)d_k + \delta_k^1(-d_k^T M d_k - z_k{}^T x_k - (\sigma-\lambda_k)\tau_k)\\ &\le \frac{1}{2}(\delta_k^1)^2 d_k^T(A-\sigma B)d_k - \delta_k^1 d_k^T M d_k\\ &= \frac{(-\tau_k+\sqrt{\tau_k^2+4c_k\tau_k})^2 d_k^T(A-\sigma B)d_k}{4c_k d_k^T B d_k} - \delta_k^1 d_k^T M d_k\\ &\le -\frac{(-\tau_k+\sqrt{\tau_k^2+4c_k\tau_k})^2 m_3}{4c_k m_4} - \delta_k^1 m_1\eta^2.\end{aligned}\tag{3.4}$$

(ii) When $\tau_k \le 0$, we investigate the following two cases according to (2.13):

(C1) If $\alpha_k = 1$, then it follows from (2.21), (2.8) and $1 \le \delta_k$ that

$$\begin{aligned}\Delta P_k &= \frac{1}{2}d_k^T(A+\sigma B)d_k + x_k^T A d_k + \sigma\tau_k\\ &\le \frac{1}{2}x_k^T A d_k + \frac{1}{2}\sigma\tau_k\\ &= -\frac{1}{2}d_k^T M d_k - \frac{1}{2}z_k^T x_k + \frac{1}{2}(\sigma+\lambda_k)\tau_k\\ &\le -\frac{1}{2}m_1\eta^2.\end{aligned}\tag{3.5}$$

(C2) If $\alpha_k = \delta_k$, it is clear that

$$\begin{aligned}\delta_k &= -\frac{x_k^T A d_k + \sigma\tau_k}{d_k^T(A+\sigma B)d_k}\\ &= \frac{d_k^T M d_k + z_k^T x_k - (\sigma+\lambda_k)\tau_k}{d_k^T(A+\sigma B)d_k}\\ &\ge \frac{d_k^T M d_k}{d_k^T(A+\sigma B)d_k}\\ &\ge \frac{m_1}{m_2}.\end{aligned}\tag{3.6}$$

It follows from (2.21), (2.8) and $0 < \delta_k \le 1$ that

$$\begin{aligned}\Delta P_k &= \frac{1}{2}\delta_k^2 d_k^T(A+\sigma B)d_k + \delta_k(x_k^T A d_k + \sigma\tau_k)\\ &= -\frac{1}{2}\frac{(x_k^T A d_k + \sigma\tau_k)^2}{d_k^T(A+\sigma B)d_k}\\ &= \frac{1}{2}\delta_k(x_k^T A d_k + \sigma\tau_k)\\ &\le -\frac{1}{2}\delta_k m_1\eta^2\\ &\le -\frac{m_1^2\eta^2}{2m_2}.\end{aligned}\tag{3.7}$$

In the following, we focus on the cases (3.3) and (3.4) with $\tau_k > 0$. First, we discuss the case $\alpha_k = \delta_k$. Define $K_1 := \{k \mid \alpha_k = \delta_k, k \in K\}$. If their exists a positive constant $\eta_1^{'} > 0$ such that $\tau_k \geq \eta_1^{'} > 0$ for any $k \in K_1$, it means the upper bound of ΔP_k in (3.3) can be updated as

$$\Delta P_k \leq -\sigma\eta_1^{'}. \tag{3.8}$$

If there exists an infinite subset $Q_1 \subset K_1$, such that $\lim_{\substack{k\in Q_1\\k\to\infty}} \tau_k = 0$. According to Theorem 2.1, we have

$$\begin{aligned}\delta_k &= \frac{d_k^T M d_k + z_k^T x_k - (\sigma+\lambda_k)\tau_k}{d_k^T(A+\sigma B)d_k}\\ &\geq \frac{d_k^T M d_k - 2\sigma\tau_k}{d_k^T(A+\sigma B)d_k}\\ &\geq \frac{m_1 - \frac{2}{\|d_k\|_2^2}\sigma\tau_k}{m_2}\\ &\geq \frac{m_1 - \frac{2}{\eta^2}\sigma\tau_k}{m_2}.\end{aligned} \tag{3.9}$$

The second inequality in (3.9) dues to the fact that $\sigma + \lambda_k \leq 2\sigma$ by $\sigma \geq |\lambda_k|$. Hence, we have

$$\begin{aligned}\lim_{\substack{k\in Q_1\\k\to\infty}} \delta_k &\geq \lim_{\substack{k\in Q_1\\k\to\infty}} \frac{m_1 - \frac{2}{\eta^2}\sigma\tau_k}{m_2}\\ &= \frac{m_1}{m_2}\\ &> 0,\end{aligned} \tag{3.10}$$

which means that there exists a $\eta_2^{'} > 0$ such that $\delta_k \geq \eta_2^{'} > 0$ for any $k \in Q_1$. Consequently, the upper bound of ΔP_k in (3.3) can be updated as

$$\Delta P_k \leq -\frac{1}{2}\eta_2^{'} m_1 \eta^2, \quad \forall k \in Q_1. \tag{3.11}$$

For the case $\alpha_k = \delta_k^1$, we define $K_2 := \{k \mid \alpha_k = \delta_k^1, k \in K\}$. If there exists an infinite subset $P_1 \subset K_2$, such that $\lim_{\substack{k\in P_1\\k\to\infty}} \tau_k = 0$. According to formula (3.4) and (2.12), we have

$$\begin{aligned}\Delta P_k &\leq -\frac{(-\tau_k + \sqrt{\tau_k^2 + 4c_k\tau_k})^2 m_3}{4c_k m_4} - \delta_k^1 m_1 \eta^2\\ &\leq -\delta_k m_1 \eta^2.\end{aligned} \tag{3.12}$$

By (3.10) and (3.12), we get

$$\Delta P_k \leq -\eta_2^{'} m_1 \eta^2, \quad \forall k \in P_1. \tag{3.13}$$

Otherwise, we may assume that there is a constant $\eta_3^{'} > 0$ such that

$$1 \geq \tau_k \geq \eta_3^{'} > 0, \quad \forall k \in K_2. \tag{3.14}$$

If c_k is bounded by a constant $\eta_4^{'}$, i.e.,

$$\eta_4^{'} \geq c_k > 0, \quad \forall k \in K_2, \tag{3.15}$$

we have

$$\begin{aligned}\delta_k^1 &= \frac{-\tau_k + \sqrt{\tau_k^2 + 4c_k\tau_k}}{2c_k}\\ &= \frac{2\tau_k}{\tau_k + \sqrt{\tau_k^2 + 4c_k\tau_k}}\\ &\geq \frac{2\tau_k}{\tau_k + \sqrt{\tau_k^2 + 4c_k\tau_k + (2c_k)^2}}\\ &= \frac{\tau_k}{\tau_k + c_k}\\ &\geq \frac{\eta_3^{'}}{1+\eta_4^{'}}.\end{aligned} \tag{3.16}$$

Therefore, we obtain

$$\begin{aligned}\Delta P_k &\leq -\frac{(-\tau_k + \sqrt{\tau_k^2 + 4c_k\tau_k})^2 m_3}{4c_k m_4} - \delta_k^1 m_1 \eta^2\\ &\leq -\delta_k^1 m_1\eta^2\\ &\leq -\frac{\eta_3^{'} m_1 \eta^2}{1+\eta_4^{'}}.\end{aligned} \tag{3.17}$$

If c_k is unbounded, i.e., there exists a constant

$$\eta_5^{'} = \frac{2+\sqrt{4+(\eta_3^{'})^2}}{(\eta_3^{'})^2} > 0,$$

such that $c_k \geq \min\{c_k, \eta_5^{'}\}$ for all $k \in K_2$. Consequently, we partition K_2 into two parts $K_2 = K_{11} \bigcup K_{12}$, where

$$K_{11} = \{k \in K_1 | 0 < c_k \leq \eta_5^{'}\}, \quad K_{12} = \{k \in K_1 | c_k > \eta_5^{'}\},$$

then the inequality (3.4) can be reformulated as

$$\Delta P_k \leq -\frac{\eta_3^{'} m_1 \eta^2}{1+\eta_5^{'}}, \quad k \in K_{11}, \tag{3.18}$$

and

$$\begin{aligned}\Delta P_k &\leq -\frac{(\tau_k^2 + 2c_k\tau_k - \tau_k\sqrt{\tau_k^2 + 4c_k\tau_k})m_3}{2c_k m_4}\\ &\leq -\frac{((\eta_3^{'})^2 + 2c_k\eta_3^{'} - \sqrt{1+4c_k})m_3}{2c_k m_4}\\ &\leq -\left(\eta_3^{'} - \frac{\sqrt{1+4c_k}}{2c_k}\right)\frac{m_3}{m_4}\\ &\leq -\frac{\eta_3^{'} m_3}{2m_4},\end{aligned} \tag{3.19}$$

for $k \in K_{12}$.

Combining the above discussion, we conclude that there exists a positive constant

$$\bar{\eta} = \min\{\frac{1}{2}m_1\eta^2, \sigma\eta_1^{'}, \frac{1}{2}\eta_2^{'} m_1\eta^2, \frac{\eta_3^{'} m_1\eta^2}{1+\eta_4^{'}}, \frac{\eta_3^{'} m_1\eta^2}{1+\eta_5^{'}}, \frac{\eta_3^{'} m_3}{2m_4}, \frac{m_1^2\eta^2}{2m_2}\} > 0 \tag{3.20}$$

such that

$$P_\sigma(x_{k+1}) \le P_\sigma(x_k) - \bar{\eta}, \quad \forall k \in K.$$

□

Lemma 3.2. *Let $\{x_k\}$ and $\{d_k\}$ be the sequences generated by the SQP algorithm, the penalty parameter σ satisfies $\sigma > \max\{|\lambda_k|, \rho\}$ for all k, where λ_k is the Lagrange multiplier of the equation (2.7), ρ is the spectral radius of matrix $B^{-1}A$. Then the sequence $\{x_k\}$ is bounded and for all $k > 1$*

$$\| x_k \|_2^2 \le \max\{\frac{2}{\lambda_{min}(B)}, \frac{x_0^T A x_0 + \sigma |x_0^T B x_0 - 2| + 2\sigma}{\lambda_{min}(A + \sigma B)}\},$$

where $\lambda_{min}(A)$ denotes the smallest eigenvalue of the matrix A.

Proof. From Theorem 2.1, d_k is a descent direction of the penalty function $P_\sigma(x)$. And since

$$P(x_{k+1}) = P_\sigma(x_k + \alpha_k d_k) = \min_{0 \le \alpha \le 1} P_\sigma(x_k + \alpha d_k),$$

we have $P(x_{k+1}) - P(x_k) \le 0$ for any k. Therefore, we can obtain

$$\begin{aligned} 0 &\le \sum_{i=0}^{k} [P_\sigma(x_i) - P_\sigma(x_{i+1})] \\ &= P_\sigma(x_0) - P_\sigma(x_{k+1}) \\ &= \frac{1}{2} x_0^T A x_0 + \sigma |\frac{1}{2} x_0^T B x_0 - 1| - \frac{1}{2} x_{k+1}^T A x_{k+1} - \sigma |\frac{1}{2} x_{k+1}^T B x_{k+1} - 1|. \end{aligned} \tag{3.21}$$

It is easy to get

$$\frac{1}{2} x_{k+1}^T A x_{k+1} + \sigma |\frac{1}{2} x_{k+1}^T B x_{k+1} - 1| \le \frac{1}{2} x_0^T A x_0 + \sigma |\frac{1}{2} x_0^T B x_0 - 1|.$$

The following discussion is divided into two situations:

(c1) If $\frac{1}{2} x_{k+1}^T B x_{k+1} - 1 < 0$, then $\| x_{k+1} \|_2^2 \le \frac{2}{\lambda_{min}(B)}$;

(c2) If $\frac{1}{2} x_{k+1}^T B x_{k+1} - 1 \ge 0$, then $\frac{1}{2} x_{k+1}^T A x_{k+1} + \sigma \frac{1}{2} x_{k+1}^T B x_{k+1} \le \frac{1}{2} x_0^T A x_0 + \sigma |\frac{1}{2} x_0^T B x_0 - 1| + \sigma$,

$$\| x_{k+1} \|_2^2 \le \frac{x_0^T A x_0 + \sigma |x_0^T B x_0 - 2| + 2\sigma}{\lambda_{min}(A + \sigma B)}. \tag{3.22}$$

In summary, the sequence $\{x_k\}$ is bounded. □

Now we proceed to establish convergence of the proposed algorithm.

Theorem 3.3. *Let $\{x_k\}$ and $\{d_k\}$ be the sequences generated by the SQP algorithm and the penalty parameter σ satisfies $\sigma > \max\{|\lambda_k|, \rho\}$ for all k, where λ_k is the Lagrange multiplier of the equation (2.7), ρ is the spectral radius of matrix $B^{-1}A$. If $\bar{x}$ is an accumulation point of the sequence $\{x_k\}$, then $\bar{x}$ is a stationary point of the minimization problem (2.1).*

Proof. From Lemma 3.2, the sequence $\{x_k\}$ is bounded. It can be assumed that there exists a subsequence that converges to $\bar{x}$. That means there is an infinite set of K_0 such that

$$\lim_{\substack{k\in K_0\\ k\to\infty}} x_k = \bar{x}. \tag{3.23}$$

Since $\{x_k\}$ and $\{d_k\}$ are sequences generated by Algorithm SQP, then x_k and d_k satisfy the Karush-Kuhn-Tucker conditions (2.7). Next, we analyze the sequence $\{d_k\}$.

If

$$\lim_{\substack{k\in K_0\\ k\to\infty}} ||d_k||_2 = 0, \tag{3.24}$$

without loss of generality, we can assume that

$$\lim_{\substack{k\in K_0\\ k\to\infty}} \lambda_k = \bar{\lambda}, \quad \lim_{\substack{k\in K_0\\ k\to\infty}} z_k = \bar{z}, \tag{3.25}$$

where λ_k, z_k are the Lagrange multipliers of the equality and inequality constraints in (2.7). By taking limits as $k\to\infty$ along the infinite subset K_0 in (2.7), then we have

$$\begin{cases} A\bar{x} = \bar{\lambda}B\bar{x} + \bar{z}, \\ \bar{z}, \bar{x} \geq 0, \\ \bar{z}^T\bar{x} = 0, \\ \frac{1}{2}\bar{x}^T B\bar{x} - 1 = 0. \end{cases} \tag{3.26}$$

Thus, $\bar{x}$ is the stationary point of the minimization problem (2.1).

If

$$||d_k||_2 \geq \eta > 0, \quad \forall k\in K_0, \tag{3.27}$$

where η is a constant. By Lemma 3.1, it is easy to get

$$P_\sigma(x_{k+1}) \leq P_\sigma(x_k) - \bar{\eta}, \quad \forall k\in K_0.$$

Therefore, we can be obtained

$$\begin{aligned} \sum_{k\in K_0} \bar{\eta} &\leq \sum_{k\in K_0} [P_\sigma(x_k) - P_\sigma(x_{k+1})] \\ &\leq \sum_{k=1}^{\infty} [P_\sigma(x_k) - P_\sigma(x_{k+1})]. \end{aligned} \tag{3.28}$$

Since

$$\lim_{k\to\infty} P_\sigma(x_k) = P_\sigma(\bar{x}), \tag{3.29}$$

it can be derived

$$\sum_{k\in K_0} \bar{\eta} \leq P_\sigma(x_1) - P_\sigma(\bar{x}) < \infty. \tag{3.30}$$

From the above formulas (3.30) and $\bar{\eta} > 0$, we know that K_0 is a finite set. It contradicts the assumption that K_0 is an infinite set. This means that hypothesis (3.27) is not true. Therefore, it can be seen that every accumulation point of $\{x_k\}$ is a stationary point of the minimization problem (2.1). □

If the sequences $\{x_k\}$ and $\{d_k\}$ generated by Algorithm SQP satisfy Theorem 3.3, Theorem 3.3 shows the accumulation point $\bar{x}$ is a stationary point of the minimization problem (2.1), i.e., $\bar{x}$ is a Pareto eigenvector of the symmetric Pareto **EiCP** (1.1) and $\bar{\lambda}$ is the corresponding Pareto eigenvalue.

$\boxed{4}$ A Simple SQP Algorithm for Symmetric Pareto EiCP

In the previous parts of the paper, we have explained in detail how to choose the precise step size and proved the global convergence of the SQP algorithm. Moreover, the search direction can be obtained through the convex quadratic programming subproblem (2.4), which can be solved efficiently by operating the MATLAB function "`quadprog`". However, it requires more computing time to solve the quadratic programming subproblem for the large-scale symmetric Pareto **EiCP**. Next, we will propose a better way to solve the subproblems. In addition, we will compare these two methods in the final numerical experiment.

Theorem 3.3 shows that the global convergence holds for any symmetric positive matrix M in (2.4), and we can choose the matrix M as an $n \times n$ diagonal matrix, i.e.,

$$\begin{aligned} \min_{d\in\mathbb{R}^n} \quad & g_k(d) = \tfrac{1}{2} d^T \Theta d + (Ax_k)^T d \\ \text{s.t.} \quad & x_k^T B d + \tfrac{1}{2} x_k^T B x_k - 1 = 0, \\ & d + x_k \geq 0, \end{aligned} \tag{4.1}$$

where Θ is a diagonal matrix with positive diagonal elements. In this paper, we refer to (4.1) as a simple quadratic programming subproblem, and in this section, we used a simple method to solve (4.1) to achieve the purpose of improving the iterative efficiency.

Theorem 4.1. *Let d_k be the solution of the simple quadratic programming subproblem (4.1) and $\theta_i > 0$ be the ith diagonal element of Θ. The ith component of d_k can be expressed as*

$$(d_k)_i = \begin{cases} \frac{1}{\theta_i}(\lambda_k Bx_k - Ax_k)_i, & \lambda_k (Bx_k)_i > (Ax_k)_i - \theta_i (x_k)_i, \\ -(x_k)_i, & \lambda_k (Bx_k)_i \leq (Ax_k)_i - \theta_i (x_k)_i, \end{cases} \tag{4.2}$$

where λ_k is the Lagrange multiplier of the equality constraint in (4.1).

Proof. The solution d_k and the Lagrange multiplier λ_k of the simple quadratic programming subproblem (4.1) satisfy the Karush-Kuhn-Tucker conditions (2.7) with $M = \Theta$, so for each component of the corresponding vectors we have

$$\begin{cases} \theta_i (d_k)_i = (\lambda_k Bx_k - Ax_k)_i + (z_k)_i, \\ (z_k)_i \geq 0, \ (d_k + x_k)_i \geq 0, \\ (z_k)_i (d_k + x_k)_i = 0. \end{cases} \tag{4.3}$$

Let $u_i := (\lambda_k Bx_k - Ax_k)_i$, then $(d_k)_i = \frac{1}{\theta_i}(u_i + (z_k)_i)$. Next, we discuss the following two cases:

(C1) If $u_i > -\theta_i (x_k)_i$, i.e., $\lambda_k (Bx_k)_i > (Ax_k)_i - \theta_i (x_k)_i$, then

$$\theta_i (d_k + x_k)_i = u_i + (z_k)_i + \theta_i (x_k)_i > (z_k)_i \geq 0,$$

which yields $(d_k + x_k)_i > 0$ since $\theta_i > 0$. Furthermore, we immediately have $(z_k)_i (d_k + x_k)_i = 0$ by taking $(z_k)_i = 0$.

(C2) If $u_i \leq -\theta_i (x_k)_i$, i.e., $\lambda_k (Bx_k)_i \leq (Ax_k)_i - \theta_i (x_k)_i$, then

$$(d_k)_i = -(x_k)_i = \frac{1}{\theta_i}(u_i + (z_k)_i),$$

which leads $(d_k + x_k)_i = 0$ and $(z_k)_i = -\theta_i (x_k)_i - u_i \geq 0$.

We provide the result by concluding the above two case. □

From Theorem 4.1, we know that the d_k in (4.2) can be regarded as a function $d_k(\lambda_k)$ about the variable λ_k and $d_k = d_k(\lambda_k)$ satisfies the equality $x_k^T B d_k + \frac{1}{2}x_k^T B x_k - 1 = 0$. To this end, we first give the bounds on the Lagrange multiplier λ_k in the following conclusion.

Theorem 4.2. *Let λ_k be the Lagrange multiplier of the equality constraint in (4.1), then*

$$\min\left\{\frac{(Ax_k)_i - \theta_i(x_k)_i}{(Bx_k)_i} \mid (Bx_k)_i > 0\right\} < \lambda_k \le \max\{\Lambda_1, \Lambda_2, \Lambda_3\}, \tag{4.4}$$

where

$$\Lambda_1 := \frac{x_k^T B\Theta^{-1}Ax_k - \frac{1}{2}x_k^T Bx_k + 1}{x_k^T B\Theta^{-1}Bx_k}, \quad \Lambda_2 := \max\{\tfrac{(Ax_k)_i - \theta_i(x_k)_i}{(Bx_k)_i} \mid (Bx_k)_i > 0\}, \tag{4.5}$$

$$\Lambda_3 := \frac{|Bx_k|\,\Theta^{-1}\,|Ax_k| + 1 - \frac{1}{2}x_k^T Bx_k}{\min\{\frac{(Bx_k)_i^2}{\theta_i} \mid (Bx_k)_i^2 \ne 0\}}. \tag{4.6}$$

Proof. Define the index set

$$N := \{i \mid \lambda_k(Bx_k)_i > (Ax_k)_i - \theta_i(x_k)_i\}, \quad R := \{i \mid \lambda_k(Bx_k)_i \le (Ax_k)_i - \theta_i(x_k)_i\},$$

then the set N is nonempty. Otherwise, we have $\lambda_k Bx_k \le Ax_k - x_k$, it follows from (4.2) that $d_k = -x_k$. Substituting it in the the equality $x_k^T Bd_k + \frac{1}{2}x_k^T Bx_k - 1 = 0$, we get $-\frac{1}{2}x_k^T Bx_k = 1$, which contradicts to the fact that the matrix B is symmetric positive definite.

If $(Bx_k)_i \le 0$ for all $i \in N$, then we have

$$\sum_{i\in N}(d_k)_i(Bx_k)_i \le -\sum_{i\in N}(x_k)_i(Bx_k)_i$$

due to $(d_k)_i > -(x_k)_i$ for all $i \in N$. Consequently, we have

$$\begin{aligned}
&x_k^T Bd_k + \frac{1}{2}x_k^T Bx_k - 1\\
&= \sum_{i\in N}(d_k)_i(Bx_k)_i - \sum_{i\in R}(x_k)_i(Bx_k)_i + \frac{1}{2}x_k^T Bx_k - 1\\
&\le -\sum_{i\in N}(x_k)_i(Bx_k)_i - \sum_{i\in R}(x_k)_i(Bx_k)_i + \frac{1}{2}x_k^T Bx_k - 1\\
&= -\frac{1}{2}x_k^T Bx_k - 1\\
&< 0,
\end{aligned} \tag{4.7}$$

which contradicts to $x_k^T Bd_k + \frac{1}{2}x_k^T Bx_k - 1 = 0$. Hence, there at least exists an index i such that $(Bx_k)_i > 0$ and $\lambda_k(Bx_k)_i > (Ax_k)_i - \theta_i(x_k)_i$, i.e.,

$$\lambda_k > \min\left\{\frac{(Ax_k)_i - \theta_i(x_k)_i}{(Bx_k)_i} \mid (Bx_k)_i > 0\right\}.$$

On the other hand, it follows from (4.2) that

$$\begin{aligned}
&x_k^T B d_k + \frac{1}{2} x_k^T B x_k - 1 \\
&= \sum_{i \in N} (d_k)_i (Bx_k)_i - \sum_{i \in R} (x_k)_i (Bx_k)_i + \frac{1}{2} x_k^T B x_k - 1 \\
&= \sum_{i \in N} \frac{1}{\theta_i} (\lambda_k B x_k - A x_k)_i (Bx_k)_i - \sum_{i \in R} (x_k)_i (Bx_k)_i + \frac{1}{2} x_k^T B x_k - 1 \\
&= 0.
\end{aligned}$$

So the λ_k can be formulated as

$$\lambda_k = \frac{\sum_{i \in N} \frac{1}{\theta_i} (Bx_k)_i (Ax_k)_i + \sum_{i \in R} (Bx_k)_i (x_k)_i + 1 - \frac{1}{2} x_k^T B x_k}{\sum_{i \in N} \frac{1}{\theta_i} (Bx_k)_i^2}. \tag{4.8}$$

Below we make assumptions about the index set R:

(i) The set R is the empty set, i.e., $\lambda_k B x_k > A x_k - \Theta x_k$. It follows from (4.2) that

$$d = \lambda_k \Theta^{-1} B x_k - \Theta^{-1} A x_k,$$

$$x_k^T B d_k + \tfrac{1}{2} x_k^T B x_k - 1 = \lambda_k x_k^T B \Theta^{-1} B x_k - x_k^T B \Theta^{-1} A x_k + \tfrac{1}{2} x_k^T B x_k - 1 = 0,$$

then we can get $\lambda_k = \frac{x_k^T B \Theta^{-1} A x_k - \frac{1}{2} x_k^T B x_k + 1}{x_k^T B \Theta^{-1} B x_k}$.

(ii) The set R is nonempty, there exists i such that $\lambda_k (Bx_k)_i \leq (Ax_k)_i - \theta_i (x_k)_i$.

(a) $\exists\ (Bx_k)_i > 0$, $i \in R$, we have

$$\lambda_k \leq \frac{(Ax_k)_i - \theta_i (x_k)_i}{(Bx_k)_i} \leq \max\{\frac{(Ax_k)_i - \theta_i (x_k)_i}{(Bx_k)_i} \mid (Bx_k)_i > 0\}.$$

(b) $\forall i \in R$, $(Bx_k)_i \leq 0$, we have

$$\begin{aligned}
\lambda_k &\leq \frac{\sum_{i \in N} \frac{1}{\theta_i} (Bx_k)_i (Ax_k)_i + 1 - \frac{1}{2} x_k^T B x_k}{\sum_{i \in N} \frac{1}{\theta_i} (Bx_k)_i^2} \\
&\leq \frac{\sum_{i \in N} \frac{1}{\theta_i} (Bx_k)_i (Ax_k)_i + 1 - \frac{1}{2} x_k^T B x_k}{\min\{\frac{(Bx_k)_i^2}{\theta_i} \mid (Bx_k)_i^2 \neq 0\}} \\
&\leq \frac{\sum_{i \in N} \frac{1}{\theta_i} \left|(Bx_k)_i\right| \left|(Ax_k)_i\right| + \sum_{i \in R} \frac{1}{\theta_i} \left|(Bx_k)_i\right| \left|(Ax_k)_i\right| + 1 - \frac{1}{2} x_k^T B x_k}{\min\{\frac{(Bx_k)_i^2}{\theta_i} \mid (Bx_k)_i^2 \neq 0\}} \\
&= \frac{\left|Bx_k\right| \Theta^{-1} \left|Ax_k\right| + 1 - \frac{1}{2} x_k^T B x_k}{\min\{\frac{(Bx_k)_i^2}{\theta_i} \mid (Bx_k)_i^2 \neq 0\}}.
\end{aligned}$$

Thus, the proof is completed. □

From the range of λ_k in (4.4), we can easily get the desired λ_k which satisfies the equality $x_k^T B d_k(\lambda_k) + \frac{1}{2} x_k^T B x_k - 1 = 0$ by the bisection method. Consequently, we can establish an algorithm for the symmetric Pareto **EiCP** by solving the simple quadratic programming subproblem (4.1).

SSQP: Simple Sequential Quadratic Programming Algorithm

1. **Initialization**

   a) Give the symmetric matrix $A \in \mathbb{S}^n$, and $B \in \mathbb{S}^n_+$ is symmetric positive definite;

   b) Choose a diagonal matrix $\Theta \in \mathbb{R}^{n\times n}$ with positive diagonal elements and a penalty parameter $\sigma > 0$;

   c) Choose a positive tolerance ϵ, the initial iteration point $x_0 \geq 0$ and $x_0 \neq 0$, and set $k = 0$.

2. **Iteration**

Step 1. Find the solution d_k of the subproblem (4.1) by the formula (4.2) and the bisection method;

Step 2. If $\|d_k\|_2 \leq \epsilon$, then stop; otherwise, go to step 3;

Step 3. Compute a step size $\alpha_k \in [0,1]$ by (2.12) and (2.13);

Step 4. Set $x_{k+1} = x_k + \alpha_k d_k$, $k = k+1$, and return to step 1.

5 Numerical Experiments

In this section, we will provide important details for the implementation of the proposed algorithm and describe how to set up the numerical experiments. Finally, we discuss their numerical performance for computing complementary eigenvalues. All computations have been performed on a personal computer with Intel(R) Core(TM)i7-8700 CPU. The algorithms have been implemented in MATLAB environment.

5.1 Implementation details

The most indispensable part of the SQP algorithm is the choice about the symmetric positive definite matrix M. Different choices of M may result in different numerical performance. As mentioned in the second part of the article, the matrix M we selected in this paper is a fixed approximate positive definite matrix, which is different from the classical sequential quadratic programming algorithm. In our numerical experiments, we test the proposed SQP algorithm with three kinds of choices about the matrix M and the corresponding algorithms are respectively named SQP(G), SSQP(D), and SSQP(I) for convenience, i.e.,

1. SQP(G) : $M = \begin{cases} A + \rho_1 I, & A \in \mathbb{S}^n, \\ A, & A \in \mathbb{S}^n_+, \end{cases}$

2. SSQP(D) : $M = \begin{cases} \operatorname{diag}(A + \rho_1 I), & A \in \mathbb{S}^n, \\ \operatorname{diag}(A), & A \in \mathbb{S}^n_+, \end{cases}$

3. SSQP(I) : $M = I$,

where $\rho_1 = \rho_A + 0.01$, ρ_A is the spectral radius of matrix A, I is the identity matrix and $\operatorname{diag}(A)$ is the diagonal part of matrix A.

As mentioned in the previous parts of this paper, the critical step of the SQP and SSQP algorithms is to determine the search direction d_k by solving the convex quadratic programming subproblems (2.4) and (4.1) in each iteration, respectively. In numerical experiments, the MATLAB function "`quadprog`" is used to solve the subproblem (2.4) of the SQP(G) algorithm. In the SSQP(D) and SSQP(I) algorithms, the solution d_k of each simple subproblem (4.1) is obtained from the explicit expression (4.2), where the bisection method is required. We set the stopping tolerance of the bisection method with $\epsilon = 10^{-9}$.

As shown in Theorem 3.3, we should choose the penalty parameter σ such that $\sigma > \max\{|\lambda_k|, \rho\}$ for all k to ensure the global convergence of the SQP algorithm. However, it is not easy to give a proper penalty parameter σ in practical computations due to the obstacles in the way of estimating an upper bound about the Lagrange multiplier λ_k for all k. Moreover, the SQP algorithm would tend to get worse with the increase of the penalty parameter σ, so it is inappropriate to choose a large penalty parameter in computational practice. Indeed, the Lagrange multiplier λ_k is an approximation of a Pareto eigenvalue of the symmetric Pareto **EiCP**, and as well known that the Pareto eigenvalues are less than ρ, which means that the Lagrange multiplier λ_k will be no larger than ρ for the sufficient large k in iteration. In the numerical experiments, we set $\sigma_0 = \rho + 0.01$ and then update $\sigma_k = \max\{\sigma_0, |\lambda_k|+0.1\}$ as the algorithm iterates. We find that $\sigma = \max\{\rho+0.01, |\lambda_1|+0.1\}$ and remains almost constant in the later iterations.

To analyze the effectiveness of the SQP and SSQP algorithms, we also solved the test problems by using the BAS algorithm in [4], the SPL algorithm in [7] and the sequential FLQP algorithm in [12]. Here we briefly explain that the BBP algorithm is applied in the sequential FLQP algorithm (FLQP). The initial point is the canonical vector e^s, where $s = \arg\max\{r_i | i = 1, ..., n\}$, $r_i = \min\{a_{ji}b_{ii} - a_{ii}b_{ji} | j = 1, ..., n\}$ [4]. For the test problems, the tolerance value for terminating these algorithms is $\|d_k\|_2 \leq 10^{-6}$ unless otherwise noted, and the maximum number of iterations is 100000. Here is a brief explanation, when the matrix A is an indefinite matrix, we obtain the Pareto eigenvalue of matrix pair (A, B) by solving the matrix pair $(A+\rho_1 B, B)$ in the SPL and FLQP algorithms, where $\rho_1 = \rho_A + 0.01$ and ρ_A is the spectral radius of matrix A. This ensures that the matrix $A + \rho_1 B$ is positive definite.

In the tables below, we use the following notations:

- n: the order of the matrix.
- λ: computed Pareto eigenvalue.
- IT: number of iterations required by the corresponding algorithm.
- CPU: computational time in seconds.
- Dualfeas $= \min\{w_i : i = 1, ..., n\}$, where w_i are the components of the vector $w = Ax - \lambda Bx$ at the computed solution of the Pareto **EiCP**.
- $*$: the maximum number of iterations has been reached.
- $-$: the algorithm has not converged after running for more than 300 seconds.

In addition, the boldface numbers are the best performance in time.

$\boxed{5.2}$ Convergence performance

In this subsection, we report the convergence performance of Algorithms SQP(G), SSQP(D), SSQP(I), FLQP, SPL, and BAS for solving the test problems. In order to test the feasibility of our algorithms, we first test a small problem. In this example, we set

$$A = \begin{bmatrix} 4 & -7 & 0 & 0 \\ -7 & -2 & 6 & 0 \\ 0 & 6 & 2 & -1 \\ 0 & 0 & -1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

We arbitrarily choose a non-zero and non-negative initial vector, such as $x_0 = [0, 0, 1, 0]^T$. We obtain

$$\lambda(x^*) = -0.4142 \quad \text{and} \quad x^* = \begin{bmatrix} 0 \\ 0 \\ 0.5412 \\ 1.3066 \end{bmatrix}$$

by using Algorithms SQP(G), SSQP(D) and SSQP(I) respectively. By further concrete computations, we have

$$Ax^* - \lambda(x^*)Bx^* = \begin{bmatrix} 0 \\ 3.2472 \\ 0 \\ 0 \end{bmatrix},$$

which means that $\lambda(x^*)$ and x^* are the Pareto eigenvalue and eigenvector of this problem. This also proves that the algorithm we proposed is effective. The matrix pair (A, B) has three non-zero Pareto eigenvalues with $\lambda_1 = -6.6158$, $\lambda_2 = -0.4142$, and $\lambda_3 = -0.2048$, respectively. In addition, we set the initial vector as $x = rand(4, 1)$ and test it randomly one hundred times. The Pareto eigenvalue we found in 83 experiments was $\lambda_1 = -6.6158$, while the Pareto eigenvalue found in the other 17 experiments was $\lambda_2 = -0.4142$. We can find out different eigenvalues by different initial vectors. But most cases will get the same eigenvalue. The number of occurrences of eigenvalue $\lambda_3 = -0.2048$ is zero. This is because our algorithm is a descent algorithm that prefers to solve for the minimum of the objective function. If we want to solve for other eigenvalues, the initial vectors need to be carefully designed.

Test problem 1, 2, 3 and 4 are generated by setting A as follows:

$$A = Q^T DQ,$$

where $Q \in \mathbb{R}^{n\times n}$ is an orthogonal matrix obtained from the QR decomposition of a random matrix, and $D \in \mathbb{R}^{n\times n}$ is a diagonal matrix, where diagonal entries d_i is a random generated number. We set matrix $B = I$ or $B = C^TC + I$, where $C \in \mathbb{R}^{n\times n}$ is a random matrix with full rank. We choose diagonal entries $d_i \in (1, 1000)$ in Test Problems 1. Test Problems 2, A is a random produced positive definite matrix with $d_i \in (1, 1000)$ and $B = I$. Test Problems 3, A is a random produced positive definite matrix with $d_i \in (1, 1000)$ and $B = C^TC+I$. Test Problems 4, A is a random generated symmetric indefinite matrix with $d_i \in (-100, 1000)$ and $B = I$. For Test Problems 5, B is the identity matrix and A is a symmetric positive definite matrix from the Harwell-Boeing collection.

It is well known that the program "`fmincon`" in MATLAB can also solve the problem (2.1). To better appreciate the efficiency of our proposed algorithms, we use "`fmincon`" as a benchmark to solve Test problems 1. We considered $n = 20, 50, 70, 100, 150$. It can be seen from Table 1 that the program "`fmincon`" in MATLAB can solve the problem (2.1) well in the low-order case. When the order of the problem exceeds 100, it is difficult to get an accurate solution, as shown by the values given in the column "Dualfeas". From

Table 1: Performance of algorithms for solving Test Problems 1.

			$B = I$			
n		20	50	70	100	150
`fmincon`	λ	3.0210e+01	1.0643e+02	1.1159e+02	1.5952e+02	1.2151e+02
	Dualfeas	-7.0392e-05	-2.6635e-05	-4.3204e-05	-7.0511e-05	-3.4524e+00
	CPU	3.0000e-02	1.4600e-01	1.8600e-01	3.7500e-01	1.3470e+00
SQP(G)	λ	3.0210e+01	1.0643e+02	1.1159e+02	1.5952e+02	1.2091e+02
	Dualfeas	-1.6251e-05	-5.4845e-05	-8.1087e-05	-5.9668e-05	-3.3822e-05
	CPU	2.5000e-02	5.0000e-02	4.0000e-02	2.4900e-01	4.2000e-01
SSQP(D)	λ	3.0210e+01	1.0643e+02	1.1159e+02	1.5952e+02	1.2091e+02
	Dualfeas	-1.4942e-04	-2.0205e-04	-2.2208e-04	-1.6912e-04	-1.3276e-04
	CPU	**3.0000e-03**	**1.0000e-02**	**1.0000e-02**	**1.0000e-02**	**2.0000e-02**
SSQP(I)	λ	3.0210e+01	1.0643e+02	1.1159e+02	1.5952e+02	1.2091e+02
	Dualfeas	-2.6364e-07	-3.6758e-07	-3.0330e-07	-2.0569e-07	-1.0911e-07
	CPU	1.4300e-01	2.1000e-01	2.6000e-01	5.5100e-01	7.1700e-01
			$B = C^T C + I$			
`fmincon`	λ	7.1620e-02	2.0291e-02	7.0364e-03	3.3512e-03	2.3627e-03
	Dualfeas	-6.3280e-05	-4.1203e-02	-4.6145e-01	-2.4257e-01	-2.5792e-01
	CPU	5.3000e-02	7.3000e-02	3.5700e-01	5.3100e-01	1.4750e+00
SQP(G)	λ	7.1620e-02	9.7273e-03	5.1241e-03	2.7958e-03	1.4851e-03
	Dualfeas	-9.7893e-05	-3.0917e-05	-2.0608e-06	-4.3515e-07	-1.0827e-06
	CPU	**1.0000e-02**	**2.0000e-02**	**1.0000e-02**	**2.0000e-02**	**3.0000e-02**
SSQP(D)	λ	7.1620e-02	9.7273e-03	5.1241e-03	2.7958e-03	1.4851e-03
	Dualfeas	-2.3715e-04	-1.5511e-04	-1.7208e-04	-1.4218e-04	-1.1456e-04
	CPU	1.2700e-01	2.4700e-01	5.0000e-02	3.0000e-02	7.0100e-01
SSQP(I)	λ	7.1620e-02	9.7273e-03	5.1241e-03	2.7964e-03	1.4861e-03
	Dualfeas	-9.2069e-05	-8.9468e-05	-4.8748e-05	-2.3514e-03	-6.5100e-04
	CPU	7.0200e+00	7.6920e+00	7.9870e+00	1.3836e+01	1.5209e+01

the benchmark results in Table 1, we can know that our proposed algorithms have good performance.

Next, we compare the convergence performance of Algorithms SQP(G), SSQP(D), SSQP(I), BAS, SPL and FLQP by four Test problems. We set the matrix $A = Q^T DQ$, where $Q \in \mathbb{R}^{100\times 100}$ is an orthogonal matrices obtained from the QR decomposition of a random matrix, and $D \in \mathbb{R}^{100\times 100}$ is a diagonal matrix with diagonal entries $\lambda_1 > \lambda_2 > \cdots > \lambda_{100}$, and the ith is

$$\lambda_{n-i+1} = \lambda_{100} + \frac{i-1}{n-1}(\lambda_1 - \lambda_{100}), \quad i = 1, 2, \ldots, n.$$

Obviously, the matrix A is symmetric positive definite if $\lambda_{100} > 0$, and A is symmetric indefinite if $\lambda_1 > 0$ and $\lambda_{100} < 0$. In addition, we set the matrix $B = I$ or $B = C^T C + I$, where $C \in \mathbb{R}^{100\times 100}$ is a random matrix.

A is a symmetric positive definite matrix with the maximum eigenvalue $\lambda_1 = 1000$ and the minimum eigenvalue $\lambda_{100} = 1$ in Figure 1. In Figure 2, A is a symmetric and indefinite matrix with the maximum eigenvalue $\lambda_1 = 100$ and the minimum eigenvalue $\lambda_{100} = -50$. In each figure, we plot the curves of the norm d_k of the testing methods versus the number of iteration steps. These figures clearly show that Algorithm SPL, Algorithm FLQP, Algorithm SQP(G), and Algorithm SSQP(D) require fewer iterations in the computation than the other algorithms. Algorithms SPL, FLQP, SQP(G), and SSQP (D) show relatively smooth norm curves, whereas Algorithms SSQP(I) and BAS are oscillatory, and the convergence behaviors are erratic. Algorithms SQP(G) and SSQP(D) show better convergence performance than the SSQP(I) algorithm because the corresponding convex quadratic programming subproblems (2.4) and (4.1) contain more information about the original constrained quadratic programming subproblem (2.1).

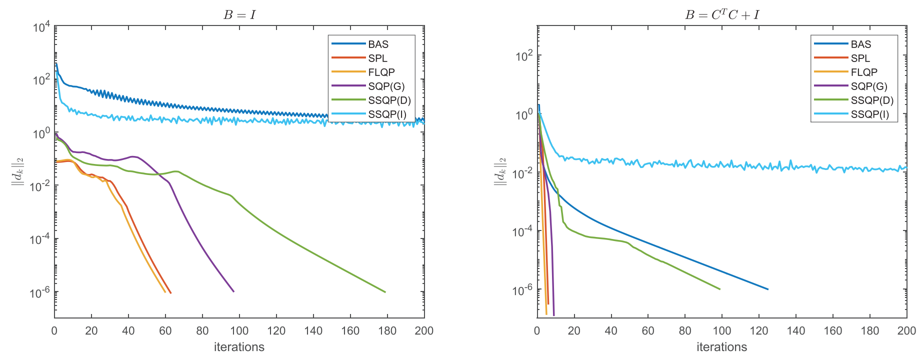

Figure 1: A is a symmetric positive definite matrix with $\lambda_1 = 1000$ and $\lambda_{100} = 1$.

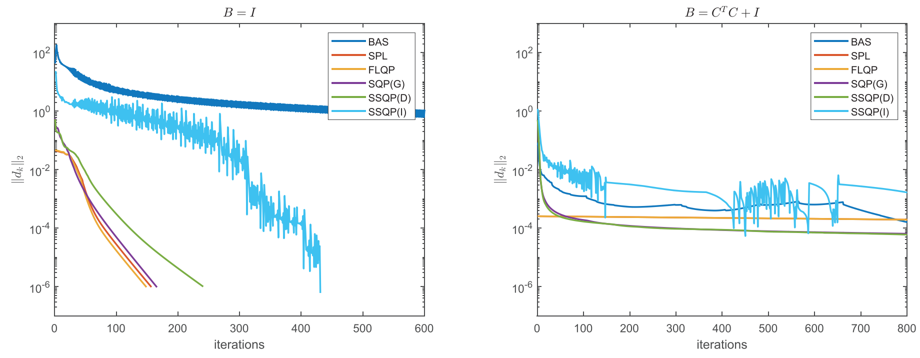

Figure 2: A is a symmetric indefinite matrix with $\lambda_1 = 100$ and $\lambda_{100} = -50$.

The numerical results of various test problems with different sizes are displayed in Tables 2, 3, and 4 to compare Algorithms BAS, SPL, FLQP, SQP(G), SSQP(D), and SSQP(I). In most cases, there is more than one solution to the Pareto **EiCP**, and these algorithms search from different directions, so sometimes the algorithms may find distinct Pareto eigenvalues. As discussed in [4, 7, 12], the BAS, SPL, and FQLP algorithms can effectively solve the symmetric Pareto **EiCP** problem. From these numerical experiments, we can find that the SSQP(D) algorithm is also effective in solving the symmetric Pareto **EiCP** problem and has some advantages over other algorithms in terms of solution speed. In Table 3, all the algorithms perform well except for the SSQP(I) algorithm. The BAS algorithm performs well when the order of the matrix is low. We can observe that the FLQP algorithm and the SQP(G) algorithm perform poorly when matrix A is indefinite. The FLQP and SQP(G) algorithms fail to converge after running for more than 300 seconds when the matrix order is 2000. In Table 5, the SPL and SSQP(D) algorithms effectively solve the problem, although the SQP(G) and SSQP(I) algorithms do not perform well in some cases.

From Table 1 to Table 5, it is easy to see that the SSQP(D) algorithm and the SPL algorithm perform better among all the algorithms. To better understand the proposed algorithm, we test the performance of the SSQP(D) algorithm and the SPL algorithm at smaller stopping tolerance values. In Table 6, we test the performance of the SSQP(D) and SPL algorithms under Problem 1, where the termination tolerance is $\|d_k\|_2 \leq 10^{-8}$. As can be seen from Table 6, the SSQP(D) algorithm also finds high-quality solutions. Since the SPL algorithm solves the subproblems by the block principal pivoting algorithm,

Table 2: Performance of algorithms for solving Test Problems 2.

n		50	100	200	500	700	1000	2000
BAS	λ	8.8601e+01	1.2850e+02	1.2572e+02	1.1043e+02	1.1788e+02	1.2895e+02	1.1539e+02
	IT	21591	*	11536	*	*	*	*
	Dualfeas	-7.3983e-08	-1.8990e-06	-1.0591e-07	-4.3928e-05	-4.2302e-05	-3.0565e-05	-2.4376e-05
	CPU	3.2300e-01	3.2860e+00	3.9400e-01	7.7180e+00	1.0273e+01	1.6513e+01	1.5606e+02
SPL	λ	8.8601e+01	1.2850e+02	1.2572e+02	1.1042e+02	1.1787e+02	1.2895e+02	1.1538e+02
	IT	32	75	84	148	126	273	340
	Dualfeas	-3.1165e-05	-3.5395e-05	-2.3875e-05	-2.0791e-05	-1.6995e-05	-1.8233e-05	-1.2027e-05
	CPU	4.0000e-02	2.0000e-02	6.7000e-02	8.0300e-01	1.2210e+00	4.6640e+00	2.2764e+01
FLQP	λ	8.8601e+01	1.2850e+02	1.2572e+02	1.1042e+02	1.1787e+02	1.2895e+02	1.1538e+02
	IT	30	69	79	134	121	268	327
	Dualfeas	-3.3252e-05	-3.6368e-05	-2.5940e-05	-2.0870e-05	-1.8083e-05	-1.8177e-05	-1.1975e-05
	CPU	5.9000e-02	2.4300e-01	5.9000e-01	3.1120e+00	6.3160e+00	2.4954e+01	1.3897e+02
SQP(G)	λ	8.8601e+01	1.2850e+02	1.2572e+02	1.1042e+02	1.1787e+02	1.2895e+02	–
	IT	34	73	173	674	382	1243	–
	Dualfeas	-3.7160e-05	-3.5969e-05	-3.5028e-05	-2.2816e-05	-1.8878e-05	-1.9240e-05	–
	CPU	3.8000e-02	1.4700e-01	7.7700e-01	2.1382e+01	2.4922e+01	1.7126e+02	–
SSQP(D)	λ	8.8601e+01	1.2850e+02	1.2572e+02	1.1042e+02	1.1787e+02	1.2895e+02	1.1538e+02
	IT	113	187	257	716	426	1136	1546
	Dualfeas	-1.5975e-04	-1.4199e-04	-1.2460e-04	-9.4285e-05	-8.7055e-05	-7.2075e-05	-5.2841e-05
	CPU	**8.0000e-03**	**1.0000e-02**	**2.0000e-02**	**1.3600e-01**	**1.7600e-01**	**3.5300e-01**	**3.2590e+00**
SSQP(I)	λ	8.8601e+01	1.2850e+02	1.2572e+02	1.1042e+02	1.1787e+02	1.2895e+02	1.1538e+02
	IT	3039	3629	3564	3874	4053	3876	4509
	Dualfeas	-2.7264e-07	-6.6573e-08	-2.4043e-07	-3.0016e-08	-1.0093e-07	-1.2692e-07	-8.1915e-08
	CPU	2.9800e-01	6.5500e-01	8.0900e-01	1.4530e+00	2.1840e+00	2.9770e+00	1.1139e+01

which is a direct method, it has an advantage over the SSQP(D) algorithm in terms of solution accuracy. In addition, the SSQP(D) algorithm loses some information when solving the subproblem with a diagonal matrix instead of the Hessian matrix of the Lagrangian function, which also causes a loss in the quality of the solution. The SSQP(D) algorithm still has advantages in the speed of computation when extremely high-quality solutions are not required.

In summary, the SSQP algorithm can solve the symmetric Pareto **EiCP** efficiently. In some instances, the SQP(G) algorithm requires fewer iterations but takes more time than the SSQP(D) algorithm, mainly because the SQP(G) algorithm pays too much on the solution of the subproblem. The SSQP algorithm uses (4.2) to obtain the solution of the simple subproblem (4.1), thus reducing the computation time. When matrix B is the identity matrix, although the SSQP(I) algorithm obtains a higher quality solution, it costs more time. The SSQP(D) algorithm combines the advantages of Algorithm SQP(G) and Algorithm SSQP(I) to solve the problem quickly and efficiently. When extremely high-quality solutions are not required, the SSQP(D) algorithm has advantages over existing algorithms.

6 Conclusions

In this paper, a sequential quadratic programming (SQP) algorithm framework is proposed for symmetric Pareto **EiCP**. The algorithm framework is to determine the search direction by a sequence of quadratic programming subproblems and a new approximate solution

Table 3: Performance of algorithms for solving Test Problems 3.

n		50	100	200	500	700	1000	2000
BAS	λ	1.3871e-02	2.6722e-03	7.9018e-04	1.0975e-04	5.8905e-05	3.0448e-05	7.3462e-06
	IT	84	73	70	197	93	254	197
	Dualfeas	-1.8737e-04	-9.6869e-05	-8.8549e-05	-3.1552e-04	-8.1174e-05	-2.7185e-04	-1.6257e-04
	CPU	1.0000e-02	**4.0000e-03**	**2.2000e-02**	2.8000e-02	**2.8000e-02**	1.5600e-01	7.2000e-01
SPL	λ	1.3871e-02	2.6722e-03	7.9018e-04	1.0975e-04	5.8905e-05	3.0448e-05	7.3462e-06
	IT	17	13	12	10	10	9	9
	Dualfeas	-1.8494e-05	-2.2241e-06	-1.2181e-06	-9.5447e-08	-4.4485e-08	-2.0345e-07	-3.0967e-08
	CPU	**1.0000e-02**	1.1000e-02	6.7000e-02	1.3700e-01	3.1200e-01	5.1100e-01	2.3060e+00
FLQP	λ	1.3871e-02	2.6722e-03	7.9018e-04	1.0975e-04	5.8905e-05	3.0448e-05	7.3462e-06
	IT	15	9	9	6	6	6	6
	Dualfeas	-9.5200e-06	-1.5879e-06	-3.2868e-07	-4.6977e-07	-1.4867e-07	-3.0152e-08	-1.6739e-09
	CPU	2.0000e-02	3.4000e-02	8.2000e-02	2.4200e-01	4.3100e-01	9.0000e-01	3.7150e+00
SQP(G)	λ	1.3871e-02	2.6722e-03	7.9020e-04	1.0988e-04	5.8978e-05	3.0547e-05	7.4065e-06
	IT	10	11	11	12	12	12	12
	Dualfeas	-3.6021e-05	-2.7197e-07	-2.7036e-06	-1.8458e-05	-1.6095e-05	-2.3437e-05	-2.1149e-05
	CPU	1.3000e-02	2.5000e-02	9.9000e-02	2.7500e-01	6.3500e-01	1.3070e+00	6.3920e+00
SSQP(D)	λ	1.3871e-02	2.6722e-03	7.9019e-04	1.0975e-04	5.8908e-05	3.0451e-05	7.3478e-06
	IT	2890	4889	2146	63	60	55	39
	Dualfeas	-1.9180e-04	-1.0813e-04	-9.4621e-05	-6.0933e-05	-6.0775e-05	-4.8960e-05	-4.0279e-05
	CPU	2.7200e-01	7.2400e-01	5.1500e-01	**1.8000e-02**	4.1000e-02	**6.0000e-02**	**1.7700e-01**
SSQP(I)	λ	1.3871e-02	2.6724e-03	7.9085e-04	1.0982e-04	5.8909e-05	3.3961e-05	–
	IT	*	*	*	*	*	*	–
	Dualfeas	-6.9910e-05	-7.4194e-04	-1.8153e-03	-3.5793e-04	-1.7450e-04	-2.4797e-03	–
	CPU	8.0280e+00	1.4048e+01	1.7484e+01	3.1096e+01	4.1766e+01	9.4249e+01	–

Table 4: Performance of algorithms for solving Test Problems 4.

n		50	100	200	500	700	1000	2000
BAS	λ	7.3288e+01	4.0515e+01	-1.6021e+00	5.2221e+00	2.0878e+01	1.1873e+01	1.0936e+01
	IT	*	*	*	*	*	*	*
	Dualfeas	-1.8012e-07	-5.7355e-05	-6.3618e-05	-1.3631e-05	-3.2646e-05	-5.0957e-05	-2.6534e-05
	CPU	1.4590e+00	3.2640e+00	3.8680e+00	7.8790e+00	1.0375e+01	1.6538e+01	1.5602e+02
SPL	λ	4.8601e+01	4.0515e+01	-1.6034e+00	5.2221e+00	2.0875e+01	1.1869e+01	1.0932e+01
	IT	299	633	236	543	1099	717	824
	Dualfeas	-3.3402e-04	-2.1729e-04	-1.9630e-04	-1.4881e-04	-1.4104e-04	-1.2635e-04	-8.1778e-05
	CPU	4.7000e-02	1.6600e-01	1.7400e-01	2.4800e+00	7.3120e+00	9.8240e+00	5.2964e+01
FLQP	λ	4.8601e+01	4.0515e+01	-1.6034e+00	5.2221e+00	2.0875e+01	1.1869e+01	–
	IT	285	628	226	543	1095	713	–
	Dualfeas	-3.3766e-04	-2.1786e-04	-2.1473e-04	-1.4676e-04	-1.4125e-04	-1.2668e-04	–
	CPU	4.4400e-01	2.0090e+00	1.4190e+00	1.1881e+01	4.7843e+01	6.4447e+01	–
SQP(G)	λ	7.3288e+01	4.0515e+01	-1.6034e+00	5.2221e+00	2.0875e+01	1.1869e+01	–
	IT	290	727	263	997	1498	871	–
	Dualfeas	-2.8018e-04	-2.2346e-04	-1.7544e-04	-1.7642e-04	-1.4517e-04	-1.2889e-04	–
	CPU	3.9000e-01	1.4890e+00	1.0470e+00	2.5866e+01	8.3413e+01	1.0989e+02	–
SSQP(D)	λ	7.3288e+01	4.0515e+01	-1.6034e+00	5.2221e+00	2.0875e+01	1.1869e+01	1.0932e+01
	IT	384	971	366	1329	2036	1173	1746
	Dualfeas	-3.7961e-04	-2.9850e-04	-2.7256e-04	-2.5001e-04	-2.0455e-04	-1.5590e-04	-1.4793e-04
	CPU	**1.3000e-02**	**5.5000e-02**	**2.1000e-02**	**2.0500e-01**	**3.8600e-01**	**4.4100e-01**	**3.3310e+00**
SSQP(I)	λ	7.3288e+01	4.0515e+01	-1.6034e+00	5.2221e+00	2.0875e+01	1.1869e+01	1.0932e+01
	IT	2988	2957	3058	3563	3747	4816	4151
	Dualfeas	-3.5169e-07	-6.4331e-08	-2.1290e-07	-3.9688e-08	-9.0270e-08	-2.7804e-08	-3.6852e-08
	CPU	2.6200e-01	5.2400e-01	6.7800e-01	1.3540e+00	2.0150e+00	3.2540e+00	1.0255e+01

Table 5: Performance of algorithms for solving Test Problems 5.

Problem		BCSSTK02	BCSSTK19	BCSSTK26	GR_30_30	PLAT362
n		66	817	1922	900	362
SPL	λ	6.5918e+00	1.1979e+14	1.0355e+09	6.1463e-02	2.6173e-06
	IT	19	2	1	8	36
	Dualfeas	-2.4058e-06	-1.2614e+01	-3.4983e-04	-2.4725e-09	-6.8982e-10
	CPU	**6.0000e-02**	3.5000e-02	9.2000e-02	4.8000e-02	4.1400e-01
SQP(G)	λ	6.5918e+00	1.1979e+14	–	6.1463e-02	–
	IT	19	28	–	13	–
	Dualfeas	-1.3010e-05	-7.2316e-01	–	-1.1955e-06	–
	CPU	1.4100e-01	2.0720e+00	–	4.1500e-01	–
SSQP(D)	λ	6.5918e+00	1.1979e+14	1.0355e+09	6.1463e-02	2.6173e-06
	IT	1296	1	2	318	4980
	Dualfeas	-1.6292e-03	-1.9531e-03	-3.6320e-06	-7.1320e-07	-5.2154e-09
	CPU	8.3000e-02	**5.0000e-03**	**1.8000e-02**	6.3000e-02	**3.9300e-01**
SSQP(I)	λ	6.1532e+00	1.1979e+14	–	6.1463e-02	2.6180e-06
	IT	71758	*	–	164	15972
	Dualfeas	-9.7936e-08	-2.5025e+01	–	-7.6827e-08	-2.1574e-08
	CPU	7.8550e+00	5.3890e+00	–	**3.7000e-02**	6.9100e-01

Table 6: Performance of algorithms for solving Test Problems 1.

				$B = I$				
n		100	200	500	700	1000	1500	2000
SPL	λ	9.4374e+01	1.2376e+02	1.0922e+02	1.1268e+02	1.1606e+02	1.2580e+02	1.1834e+02
	IT	77	98	133	252	284	407	615
	Dualfeas	-2.6474e-07	-1.8047e-07	-1.7129e-07	-2.0534e-07	-1.5573e-07	-1.3007e-07	-1.2227e-07
	CPU	2.3000e-02	9.2000e-02	7.5200e-01	1.7660e+00	4.5160e+00	1.4022e+01	3.6240e+01
SSQP(D)	λ	9.4374e+01	1.2376e+02	1.0922e+02	1.1268e+02	1.1606e+02	1.2580e+02	1.1834e+02
	IT	285	342	549	1131	1265	1989	2390
	Dualfeas	-1.2425e-06	-8.9861e-07	-8.4537e-07	-8.9346e-07	-6.7249e-07	-5.3137e-07	-5.0757e-07
	CPU	**1.5000e-02**	**3.0000e-02**	**9.6000e-02**	**2.3800e-01**	**4.0500e-01**	**1.8030e+00**	**4.3690e+00**
				$B = C^TC + I$				
SPL	λ	2.8766e-03	7.0960e-04	1.2524e-04	6.9109e-05	3.0008e-05	1.1868e-05	7.7352e-06
	IT	15	14	12	12	11	10	10
	Dualfeas	-3.5973e-08	-1.1712e-08	-1.4812e-09	-2.1534e-10	-3.3410e-10	-6.0960e-10	-4.3131e-10
	CPU	**1.0000e-02**	**2.0000e-02**	1.4000e-01	2.7100e-01	6.0400e-01	1.2030e+00	2.1850e+00
SSQP(D)	λ	2.8766e-03	7.0960e-04	1.2524e-04	6.9109e-05	3.0008e-05	1.1868e-05	7.7352e-06
	IT	1671	1228	113	195	97	91	89
	Dualfeas	-9.7200e-07	-8.8125e-07	-6.8473e-07	-6.1439e-07	-4.7135e-07	-4.2485e-07	-3.6926e-07
	CPU	2.7800e-01	2.0000e-01	**4.7000e-02**	**7.9000e-02**	**1.0900e-01**	**2.5500e-01**	**4.0300e-01**

can be obtained along this search direction with the exact step size. The global convergence analysis for the SQP algorithm framework is presented. The SQP algorithm seems to be competitive to the existing algorithms from numerical results. In order to further improve the computation efficiency, a kind of simple choices about the quadratic programming subproblems is introduced in practice, and various numerical results illustrate that the corresponding simple SQP algorithm (SSQP) is quite efficient for the solution of the large symmetric Pareto **EiCP**.

The key of the SQP algorithm framework is to construct a series of suitable quadratic programming subproblems. The computational efficiency of the SQP algorithm is closely related to the selection of the symmetric positive definite matrix M, which has been confirmed by a large number of numerical experiments. This article mainly focuses on the fixed matrix M, so in our future research, we will consider using a series of suitable symmetric positive definite matrices to promote the SQP algorithm proposed in this article.

References

[1] S. Adly and H. Rammal, A new method for solving Pareto eigenvalue complementarity problems, *Comput. Optim. Appl.* 55 (2013) 703–731.

[2] S. Adly and A. Seeger, A nonsmooth algorithm for cone-constrained eigenvalue problems, *Comput. Optim. Appl.* 49 (2011) 299–318.

[3] J.-B Baillon and A. Seeger, New results on Pareto spectra, *Linear Algebra Appl.* 588 (2020) 338–363.

[4] C.P. Brás, A. Fischer, J.J. Júdice, K. Schönefeld and S. Seifert, A block active set algorithm with spectral choice line search for the symmetric eigenvalue complementarity problem, *Appl. Math. Comput.* 294 (2017) 36–48.

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

[6] L. Fernandes, A. Fischer, J.J. Júdice, C. Requejo and J. Soares, A block active set algorithm for large-scale quadratic programming with box constraints, *Ann. Oper. Res.* 81 (1998) 75–95.

[7] M. Fukushima, J.J. Júdice, W. de Oliveira and V. Sessa, A sequential partial linearization algorithm for the symmetric eigenvalue complementarity problem, *Comput. Optim. Appl.* 77 (2020) 711–728.

[8] P.E. Gill, W. Murray and M.A. Saunders, SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization, *SIAM Rev.* 47 (2005) 99–131.

[9] N.I-M. Gould and P.L. Toint, SQP Methods for Large-Scale Nonlinear Programming, in: *IFIP Conference on System Modeling and Optimization.* 1999.

[10] J.J. Júdice, M. Fukushima, A. Iusem, J.M. Martinez, and V.Sessa, An alternating direction method of multipliers for the eigenvalue complementarity problem, *Optim. Methods Softw.* 36 (2021) 337–370.

[11] J.J. Júdice, M. Raydan, S.S. Rosa and S.A. Santos, On the solution of the symmetric eigenvalue complementarity problem by the spectral projected gradient algorithm, *Numer. Algorithms* 47 (2008) 391–407.

[12] J.J. Júdice, V. Sessa, and M. Fukushima, Solution of fractional quadratic programs on the simplex and application to the eigenvalue complementarity problem, *J. Optim. Theory Appl.* (2022) 545–573.

[13] J.J. Júdice, H.D. Sherali, I.M. Ribeiro and S.S. Rosa, On the asymmetric eigenvalue complementarity problem, *Optim. Methods Softw.* 24 (2009) 549–568.

[14] H.A. Le Thi, M. Moeini, T. Pham Dinh and J.J. Júdice, A DC programming approach for solving the symmetric eigenvalue complementarity problem, *Comput Optim Appl.* 51 (2012) 1097–1117.

[15] J.A.C. Martins, S. Barbarin, M. Raous and A. Pinto da Costa, Dynamic stability of finite dimensional linearly elastic systems with unilateral contact and Coulomb friction, *Comput. Methods Appl. Mech. Engrg.* 177 (1999) 289–328.

[16] J.A.C. Martins and A. Pinto da Costa, Stability of finite-dimensional nonlinear elastic systems with unilateral contact and friction, *International Journal of Solids and Structures.* 37 (2000) 2519–2564.

[17] W. Murray, Sequential quadratic programming methods for large-scale problems, *Comput. Optim. Appl.* 7 (1997) 127–142.

[18] J. Nocedal and S.J. Wright, *Numerical Optimization*, 2nd ed, Springer Ser. Oper. Res. Financ. Eng. 2006.

[19] A. Pinto da Costa and J.A.C. Martins, Computation of bifurcations and instabilities in some frictional contact problems, in: *European Conference on Computational Mechanics.* 2001.

[20] A. Pinto da Costa, J.A.C. Martins, I.N. Figueiredo and J.J. Júdice, The directional instability problem in systems with frictional contacts, *Comput. Methods Appl. Mech. Engrg.* 193 (2004) 357–384.

[21] A. Pinto da Costa and A. Seeger, Cone-constrained eigenvalue problems: theory and algorithms, *Comput. Optim. Appl.* 45 (2010) 25–57.

[22] M. Queiroz, J.J. Júdice and C.H. Jr, The symmetric eigenvalue complementarity problem, *Math. Comp.* 73 (2004) 1849–1863.

[23] P. Quittner, Spectral analysis of variational inequalities, *Comment. Math. Univ. Carolin.* 27 (1986) 605–629.

[24] R.C. Riddell, Eigenvalue problems for nonlinear elliptic variational inequalities on a cone, *J. Funct. Anal.* 26 (1977) 333–355.

[25] A. Seeger, Eigenvalue analysis of equilibrium processes defined by linear complementarity conditions, *Linear Algebra Appl.* 292 (1999) 1–14.

[26] A. Seeger, Cone-constrained eigenvalue problems: structure of cone spectra, *Set-Valued Var. Anal.* 29 (2021) 605–619.

[27] A. Seeger and D. Sossa, Measuring similarity between connected graphs: the role of induced subgraphs and complementarity eigenvalues, *Graphs Combin.* 37 (2021) 493–525.

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

[29] Y.X. Yuan and W.Y. Sun, *Optimization Theory and Method*, Science Press, 2007.

Manuscript received 15 February 2022
revised 23 July 2022, 5 November 2022
accepted for publication 7 December 2022

Lin Zhu
School of Mathematics
Hunan University, Changsha 410082, P.R. China
E-mail address:zhulin@hnu.edu.cn

Yuan Lei
School of Mathematics
Hunan University, Changsha 410082, P.R. China
E-mail address: yleimath@hnu.edu.cn

Jiaxin Xie
LMIB of the Ministry of Education
School of Mathematical Sciences
Beihang University, Beijing 100191, P.R. China
E-mail address: xiejx@buaa.edu.cn