THE BUNDLE SCHEME FOR SOLVING ARBITRARY EIGENVALUE OPTIMIZATIONS

MING HUANG
Department of Mathematics, Dalian Maritime University
Dalian 116026, China
and
School of Control Science and Engineering, Dalian University of Technology
Dalian 116024, China

XI-JUN LIANG
College of Science, China University of Petroleum
Qingdao 266580, China

YUAN LU
School of Sciences, Shenyang University
Shenyang 110044, China

LI-PING PANG∗
School of Mathematical Sciences, Dalian University of Technology
Dalian 116024, China

(Communicated by Liqun Qi)

ABSTRACT. Optimization involving eigenvalues arises in a large spectrum of applications in various domains, such as physics, engineering, statistics and finance. In this paper, we consider the arbitrary eigenvalue minimization problems over an affine family of symmetric matrices, which is a special class of eigenvalue function–D.C. function $\lambda$. An explicit proximal bundle approach for solving this class of nonsmooth, nonconvex (D.C.) optimization problem is developed. We prove the global convergence of our method, in the sense that every accumulation point of the sequence of iterates generated by the proposed algorithm is stationary. As an application, we use the proposed bundle method to solve some particular eigenvalue problems. Some encouraging preliminary numerical results indicate that our method can solve the test problems efficiently.

2010 Mathematics Subject Classification. Primary: 49J52, 15A18; Secondary: 90C26, 65K10.

Key words and phrases. Eigenvalue optimization, nonsmooth optimization, D.C. function, nonconvex optimization, first-order proximal bundle method.

∗ Corresponding author. The first and forth authors’ work was supported in part by the National Natural Science Foundation of China under projects No.11171049. The first author was supported in part by National Natural Science Foundation of China under projects No.11626053, the Project funded by China Postdoctoral Science Foundation under No. 2016M601296, Fundamental Research Funds for the Central Universities under projects No.3132016108 and Scientific Research Foundation Funds of DLMU under projects No.02501102. The second author’s work was supported in part by the National Natural Science Foundation of China under projects No. 61503412 and Natural Science Foundation of Shandong Province under Grant No. ZR2014AP004. The third author’s work was supported in part by the National Natural Science Foundation of China under projects No.11301347, and Program for Liaoning Excellent Talents in University under projects No. LIQ2015075.
1. **Introduction.** During the past decade, eigenvalue optimization problems have received remarkable attention in many application fields, among the plethora of applications of eigenvalues in mathematics and engineering. They can be mentioned such as numerical analysis, structural design, quantum mechanics and system dynamics (physical, chemical and biological models). Singular values and condition number of matrices are also defined in terms of eigenvalues. Eigenvalue optimization is an important testing-ground for nonsmooth optimization theory. Optimization problems involving the eigenvalues of a real symmetric matrix arise in many applications, from engineering design to graph-partitioning: two extensive surveys are [19, 37].

The analysis for optimization of eigenvalues of a symmetric matrix is a classical subject in applied and numerical analysis. For problems involving eigenvalues, in general it is not just the case to calculate the eigenvalues of a given matrix $A$. To obtain fast convergence, here our interest is nonconvex arbitrary eigenvalue optimization problem. Very little research dealing with the general (arbitrary) eigenvalue case has been presented. The aim of this paper is to formulate a general executable algorithm for solving the following particular model problem:

$$\min_{x \in \mathbb{R}^m} \lambda_l(A(x)), \quad 1 \leq l \leq n,$$

and the constrained optimization problem

$$\min_{x \in \mathbb{R}^m} c^T x$$

subject to

$$\lambda_l(A(x)) \leq 0,$$

where $\lambda_l(\cdot)$ is the $l$-th maximum eigenvalue function of $A(x)$, the mapping $A : \mathbb{R}^m \rightarrow \mathbb{S}_n$, is a symmetric matrix depending smoothly on the variable $x$, which satisfies $A(x) = A_0 + A \cdot x$ is affine: $A$ is a linear operator, which maps the vector $x$ to the matrix, defined by $A \cdot x := \sum_{i=1}^{m} x_i A_i$; $A_i, i = 0, 1, \cdots, m$ are the given real $n \times n$ symmetric matrices. We suppose these eigenvalues have been sorted in the decreasing order $\lambda_1(A(x)) \geq \lambda_2(A(x)) \geq \cdots \geq \lambda_n(A(x))$. Note that programs of the form (1) may be transformed into (2) using exact penalization, so we only need to design the method to solve the unconstrained problem (1) for the sake of simplicity. Generally speaking, one of the main difficulties with the numerical analysis of such problems is that the behavior of the eigenvalues $\lambda_l(A(x))$, considered as functions of a symmetric matrix, can be nonsmooth, especially at those points where $\lambda_l(A(x))$ are multiple, which are precisely these points of utmost interest. This leads that we cannot directly apply smooth optimization techniques in solving it. A gradient-based numerical method (e.g., Newton's method) may get into trouble when hitting those points. In addition, theoretical analysis becomes difficult without differentiability.

Motivated by above statement, the purpose of this paper is to introduce the recent algorithm formulations for solving the arbitrary eigenvalue optimization. In this work we describe an approach to solve the arbitrary eigenvalue problems using the proximal bundle method. The idea of using the bundle method in nonsmooth optimization can be dated back to 1970's [18, 39] for studying some convex nondifferentiable functions. To our best knowledge, up to now there is no successful method for deriving manner of computing this arbitrary eigenvalue (D.C.) problem. Therefore, this paper can be regarded as a novel technique which allows to use the corresponding information to deal with the special class of eigenvalue functions.
Next, to derive the arbitrary eigenvalue function, we study a phenomenon of intrinsic interest in eigenvalue optimization. Let $f_k(\cdot)$ be the sum of $k$ largest eigenvalues. Then, the classical eigenvalue optimization problem is often formulated as follows

$$\min_{x \in \mathbb{R}^m} f_k(A(x)) := \sum_{j=1}^{k} \lambda_j(A(x)), \quad k = 1, \ldots, n. \tag{3}$$

It is not hard to verify that $\lambda_1(\cdot)$ is a convex function (the largest eigenvalue function) and $\lambda_n(\cdot)$ is a concave function; and, for $l = 2, \ldots, n-1$, $\lambda_l(\cdot)$ is the difference of two convex functions, which can be written as $\lambda_l(A(x)) = f_l(A(x)) - f_{l-1}(A(x))$. Besides, we can easily know the function $f_k(A(x))$ is a positively homogeneous and Lipschitz continuous convex function, according to Ky Fan's maximum principle [3, 7], so the arbitrary eigenvalue function is generally a nonconvex, positively homogeneous and Lipschitz continuous (In fact the Lipschitz constant is 1) function. The function $\lambda_l(\cdot)$ can be written as a difference of the two convex functions, and is called a D.C. function. Its form is the following

$$\min_{x \in \mathbb{R}^n} f(x) := h_1(x) - h_2(x),$$

where $h_1, h_2$ are both proper convex functions on $\mathbb{R}^n$. At optimal solutions of (3) the eigenvalues of the optimal matrix tend to coalesce at some minimum point $x^*$, and the multiplicity of $\lambda_k(A(x^*))$ can be greater than two. The clustering phenomenon plays a central role in eigenvalue optimization. So the function $f_k$ is nondifferentiable at $x^*$. The readers can refer to Overton and Womersley (1993, [26]) and Hiriart-Urruty and Ye (1995, [12]) for the characterization of the subdifferential of $f_k$. A more general treatment on computing the subdifferentials of functions of eigenvalues can be seen in [19].

We need to note that the arbitrary eigenvalue functions have special structure—the D.C. form, which will make the effective algorithms to solve such problems possible under mild conditions. When $l = 1$, the maximum eigenvalue function $\lambda_1(A(x))$ has been studied from the theoretical point in [24, 25, 32, 33]. To the sum of the $k$-largest eigenvalues, we can refer to [12, 26, 27]. Many practical problems are derived from the arbitrary eigenvalue problem. For instance, when $l = n - 1$, $\lambda_l$ becomes the second smallest eigenvalue, which has been studied to solve the fastest mixing Markov process on a graph by Boyd et al. in 2006 [34], and used for homogenous quasi-linear elliptic operators by Robinson [30]. Friedland et al. proposed the inverse eigenvalue problems [8]: Cullum et al. solved the graph partitioning problems [6]; low rank matrix optimization can also be considered as an arbitrary eigenvalue problem [9]; Polak and Wardi presented structural optimization problems [28, 29]. Torki had studied the arbitrary eigenvalue function by the way of epi-differentiability [35, 36]. In 1995 Hiriart-Urruty and Ye introduced the arbitrary eigenvalue [12], where they presented the first-order sensitivity analysis of all eigenvalues of a symmetric matrix and theory result.

Approaches to solve the convex maximum eigenvalue optimization (i.e., $l = 1$) can be divided into interior point methods and nonsmooth optimization methods. The interior-point method for solving the convex eigenvalue optimization or semidefinite programming had been gained great interest in [2, 13, 23]. Besides, most of the interior-point schemes proposed in the early 1990s were the path-following or the potential reduction methods [1, 11, 21, 22]. There has been a recent renewed interest in first order subgradient methods for semidefinite programming (SDP) and
eigenvalue optimization, see \[10, 17\]. But, when \(2 \leq l \leq n\), rare work has studied the implementable algorithm in detail. Based on this point, studying the executive algorithm towards (1) is becoming more meaningful. Here our primary contribution is to gain an efficient and fast algorithm with global convergence. To the best of our knowledge, it is the first time that such an optimal method of solving the arbitrary eigenvalue problem has been presented in the literature. The main purpose of this paper is to take full advantage of special structure of the objective function and design a specialized bundle method to solve the nonconvex arbitrary eigenvalue optimization (1). The main motivation of this work is to show that making use of the idea from nonsmooth analysis, we can design an proximal bundle method for solving the nonconvex arbitrary eigenvalues problem. In addition, the global convergence of the proximal bundle method can be proved, where the generated sequence tends to a stationary point of the arbitrary eigenvalue function.

The rest of the paper is organized as follows. In Section 2, we present the bundle method for the convex maximum eigenvalues. The corresponding proximal bundle algorithm for the arbitrary eigenvalue function (D.C.) is established in Section 3. The main results are given in Section 4: for the arbitrary eigenvalue function \(\lambda_l\), we design an implementable proximal bundle algorithm with aggregate subgradients. Then in Section 5 we present the convergence result. It can be proved that our method are both readily implementable and globally convergent in the sense that all their accumulation points are stationary. As an important application, we use the proposed method to solve a class of eigenvalue problems and report numerical results in Section 6. Finally, we present some concluding remarks by pointing out some possible research topics.

2. The proximal bundle algorithm for the convex maximum eigenvalue.

In this section, we present the basic ideas about the general proximal bundle method of Kiwiel \[17\] in convex case, i.e., \(f = \lambda_1\).

Our aim is to produce a sequence \(\{x^k\}_{k=1}^\infty \subset \mathbb{R}^m\) converging to some minimizer of the problem. Consider the minimization of a finite-valued, convex function \(\lambda_1: \mathbb{R}^m \to \mathbb{R}\). Although \(\lambda_1\) may be nonsmooth, it is well known that its Moreau-Yosida regularization

\[
f_\mu(x) = \min_{y \in \mathbb{R}^m} \{ \lambda_1(y) + \frac{\mu}{2} \|y - x\|^2 \} \quad (\mu > 0)
\]

is a Lipschitz continuously differentiable convex function with the same set of minimizers as \(\lambda_1\) \[11\]. The unique minimizer \(\hat{x}\) called the proximal point of \(x\) in \[4\], satisfies

\[
\nabla f_\mu(x) = \mu(x - \hat{x}),
\]

and hence represents a substitution of \(x\) along the negative gradient of \(f_\mu\), with \(\mu^{-1}\) being a step length. The notion of the proximal point algorithm for minimizing \(f\) is then to minimize \(f_\mu\) by updating the current point \(x\) to its proximal point \(\hat{x}\).

The idea behind the proximal bundle algorithm is to approximate \(\hat{x}\) by replacing the \(\lambda_1(y)\) term in the right hand side of (4) with a cutting plane model based on the subdifferential \[11, 21\] of \(\lambda_1\). Suppose that the starting point \(x^1\) is feasible and at the \(k\)-th iteration of the algorithm we have the current iteration point \(x^k\) and some trial points \(y^j \in \mathbb{R}^m\) and some subset of past iterations \(J^k \subset \{1, \cdots, k\}\), the so-called “bundle”, namely function values \(\lambda_1(y^j)(j \in J^k)\) and a set \(G_k = \{g_j \in \partial \lambda_1(y^j) : j \in J^k\}\) of subgradients, furnishes a piecewise linear function
the first order Taylor expansion, it equivalently can be written in the form
\[ \lambda_1^k(y) = \max_{j \in J^k} \{ \lambda_1(x^k) + g_j^T(y - x^k) - \alpha_j \}, \]  
which minorizes \( \lambda_1 \): in other words, we replace \( \lambda_1 \) by so-called polyhedral cutting plane model \( \lambda^k \). \( \lambda_1^k \) equivalently can be written in the form
\[ \lambda_1^k(y) = \max_{j \in J^k} \{ \lambda_1(x^k) + g_j^T(y - x^k) - \alpha_j \}, \]  
with the linearization errors
\[ \alpha_j^k = \alpha(x^k, y^j, g_j) := \lambda_1(x^k) - \lambda_1(y^j) - g_j^T(x^k - y^j), \quad \text{for all } j \in J^k. \]

Note that in convex case the subdifferential can be rewritten as
\[ \partial f(x) = \{ g \in \mathbb{R}^m | f(y) \geq f(x) + g^T(y - x) \quad \text{for all } y \in \mathbb{R}^m \}; \]
see [31]. Then it is not difficult to prove that
\[ \hat{\lambda}_1^k(y) \leq \lambda_1(y) \quad \text{for all } y \in \mathbb{R}^m \quad \text{and} \quad \alpha_j^k \geq 0 \quad \text{for all } j \in J^k. \]  
That is to say, when we study the maximum eigenvalue function, the cutting plane model \( \hat{\lambda}_1^k \) is an underestimate for \( f = \lambda_1 \) and the linearization error \( \alpha_j \) is nonnegative, which measures the approximation of the cutting plane model to the function \( \lambda_1 \). Using \( \hat{\lambda}_1^k \) from (5) in lieu of \( \lambda_1 \) and setting \( x = x^k \) in (4) yields
\[ \min_{y \in \mathbb{R}^m} \lambda_1^k(y) + \frac{\mu}{2} \| y - x^k \|^2 = \min_{y \in \mathbb{R}^m} \{ \max_{j \in J^k} \lambda_1(y^j) + g_j^T(y - y^j) \} + \frac{\mu}{2} \| y - x^k \|^2 \]  
as the subproblem, which is readily formulated as a quadratic programming. By solving this programming problem, one obtains the optimal subgradient in (5), call it \( \tilde{g} \), as well as the unique minimizer \( y^{k+1} \) which approximates the proximal point of \( x^k \).

If \( y^{k+1} \) meets the ‘sufficient descent’ criterion
\[ \lambda_1(y^{k+1}) \leq \lambda_1(x^k) + m_L (\hat{\lambda}_1^k(y^{k+1}) - \lambda_1(x^k)) \]  
for a given constant \( m_L \in (0, 1/2) \), it is accepted as the new iterate by setting \( x^{k+1} = y^{k+1} \); this is termed a serious step. Otherwise, \( x^{k+1} = x^k \), and we have a null step. In either case, the cutting plane model \( G_k \) is improved by adding \( \tilde{g} \) and a new subgradient \( g \in \partial \lambda_1(y^{k+1}) \) to yield \( G_{k+1} \), thus making a serious step more likely when (5) is solved in the next iteration. The stopping criterion
\[ \lambda_1(x^k) - \hat{\lambda}_1^k(y^{k+1}) \leq \delta (|\lambda_1(x^k)| + 1) \]  
tests if the upper bound on the maximum attainable descent in this step [left hand side of (10)] falls below a small fraction \( \delta \) of the magnitude of the objective value [right hand side of (10)], in which case \( x^k \) is returned as the computed solution.

Two additional points need to be addressed to complete the description of the algorithm. First, less significant subgradients in \( G_k \) may be united together into an aggregate subgradient, to keep storage bounded. Second, for good practical performance, the penalty parameter \( \mu \) has to be judiciously varied at every iteration, as was already recognized in the early computational works of Lemaréchal. Both issues have been addressed by Kiwiel [15][17]. The following framework form is the resulting algorithm.
Begin of the algorithm

Algorithm 2.1 (Proximal bundle method)

Step 0 (Initialization). Given a starting point $y^0$, an improvement parameter $m_L \in (0, 1/2)$, a termination parameter $\delta > 0$, a penalty parameter $\mu_0 \geq \mu_{\min} > 0$, where $\mu_{\min}$ is a lower bound for the $\mu_k$ sequence. Set the iteration counter $k = 0$.

Compute a function value $\lambda_1(y^0)$ and a subgradient $g_0 \in \partial \lambda_1(y^0)$.

Step 1. Solve the problem (8) to obtain $y^{k+1}$ and call the oracle to get $\lambda_1(y^{k+1})$ and $g_{k+1} \in \partial \lambda_1(y^{k+1})$.

If (9), then set $x^{k+1} = y^{k+1}$ (serious step).
Else set $x^{k+1} = x^k$ (null step).

Step 2 (Stopping criterion). If (10) is satisfied, stop. Return $y^* = x^k$ as the computed solution.

Step 3 (Update). Add $\bar{g}$ and a new subgradient $g_{k+1} \in \partial \lambda_1(y^{k+1})$ to $G_k$, and perform aggregation (if necessary) to obtain $G_{k+1}$. Update $\mu_k$ to its new value $\mu_{k+1}$.

Step 4. Replace $k$ by $k + 1$ and loop to Step 1.

End of the algorithm

In the following we give the solution of the programming (8). Notice that the problem (8) still is a nonsmooth optimization problem. However, it can be rewritten as a smooth quadratic programming subproblem due to its piecewise linear nature, i.e., the problem (8) is equivalent to the quadratic programming with affine constraints:

$$\begin{align*}
\min & \quad [r + \frac{1}{2} \mu_k \|y - x^k\|^2] \\
\text{s.t.} & \quad \lambda_1(y^i) + g^T_i (y^i - y^j) \leq r, \quad \text{for}\ i = 1, \ldots, k.
\end{align*}$$

The dual of this quadratic programming can be formulated explicitly, yielding very instructive interpretations, which is a critical point to understand the different variants of bundle methods. In the following, $\Delta_k$ is the unit simplex of $R^k$ defined by $\Delta_k := \{ \gamma \in R^k_+ : \sum_{i=1}^k \gamma_i = 1 \}$; the coefficients

$$\begin{align*}
\alpha_k &= \alpha(x^k, y^i, g_i) := \lambda_1(x^k) - \lambda_1(y^i) - g^T_i (x^k - y^i), \quad \text{for}\ i = 1, \ldots, k, \\
\alpha(k) &= [\alpha_1^k, \ldots, \alpha_k^k]^T
\end{align*}$$

are the linearization errors between $y^i$ and $x^k$.

**Proposition 1.** For $\mu_k > 0$, the unique solution of the augmented model problem (8) is

$$y^{k+1} = x^k - \frac{1}{\mu_k} \sum_{i=1}^k \bar{\theta}_i g_i, \quad (14)$$

where $\bar{\theta} \in R^k$ solves

$$\min_{\theta \in \Delta_k} \frac{1}{2} \sum_{i=1}^k \theta_i g_i^2 + \mu_k \sum_{i=1}^k \theta_i \alpha_i^k. \quad (15)$$

Furthermore, there holds

$$\lambda_1(x^{k+1}) = \lambda_1(x^k) - \sum_{i=1}^k \bar{\theta}_i \alpha_i^k - \frac{1}{\mu_k} \sum_{i=1}^k \bar{\theta}_i g_i^2 \quad (16)$$
and the primal-dual relation
\[ \epsilon_k := \bar{g}^T \alpha(k) = \delta_k - \frac{1}{2\mu_k} \| \bar{g}_k \|^2, \] (17)

where \( \bar{g}_k := \sum_{i=1}^k \tilde{\theta}_i g_i \) and nominal decrease \( \delta_k := \lambda_1(x^k) - \lambda_1^{\ast}(y_{k+1}) - \frac{\mu_k}{2} \| y_{k+1} - x^k \|^2 \).

The convergence of the algorithm, which requires only that the updated bundle contain \( \bar{g} \) and one new subgradient from \( \partial \lambda_1(y_{k+1}) \), may be summarized as follows.

**Theorem 2.1.** (Kiwiel [15] [17]) Suppose that Algorithm 2.1 is implemented with \( \delta = 0 \) in the stopping test [10]. If the algorithm executes a finite number of serious steps, i.e., performs only null steps from some iteration \( k \) onwards, then \( x^k \in \arg \min f \). If the algorithm generates an infinite sequence of serious steps, then \( \{ x^k \} \) is a minimizing sequence for \( f \), and is convergent to a minimizer of \( f \), once one exists.

### 3. Proximal bundle method for D.C. arbitrary eigenvalue

In this section, we will describe the ideas of the proximal bundle method for the nonsmooth and D.C. eigenvalue function minimization.

Because the eigenvalue functions \( \lambda_i \) are locally Lipschitz, we next give the Clarke generalized gradient for local Lipschitz function.

**Definition 3.1.** [4] [5] Let \( f \) be local Lipschitz on \( \mathbb{R}^m \), the **generalized gradient** of \( f \) at \( x \), denoted by \( \partial f(x) \), is defined by
\[ \partial f(x) := \{ \xi \in \mathbb{R}^m : f^c(x; d) \geq \langle \xi, d \rangle \text{ for all } d \in \mathbb{R}^m \}, \] (18)

where \( f^c(x; d) = \limsup_{y \to x, t \to 0^+} \frac{f(y+t d) - f(y)}{t} \) is the **generalized directional derivative** of \( f \) at \( x \) in the direction \( d \).

Moreover, for each \( f \) the subdifferential of \( f \) at \( x \) can be written
\[ \partial f(x) = \text{conv}\{ g \in \mathbb{R}^m | g = \lim_{i \to \infty} \nabla f(x_i), x_i \to x, \nabla f(x_i) \text{ exists, } \nabla f(x_i) \text{ converges} \}, \]
which is a well-defined, nonempty, convex and compact subset of \( \mathbb{R}^m \).

We also assume that at each point \( x \) both the objective function value \( \lambda_i(x) \) and some subgradient \( g \in \partial \lambda_i(x) \) can be obtained.

The basic idea of the bundle methods is to approximate the whole subdifferential of the objective function by only one arbitrary subgradient at each point. In practice, this is done by gathering subgradients from the previous iterations into a bundle. Suppose that at the \( k \)-th iteration of the algorithm we have the current iteration point \( x^k \) and some trial points \( y^j \in \mathbb{R}^m \) (from past iterations) and subgradients \( g_j \in \partial \lambda_i(y^j) \) for \( j \in J^k \), where the index set \( J^k \neq \emptyset \) and \( J^k \subset \{1, \cdots, k\} \).

The idea is to approximate the objective function \( \lambda_i \) below by a piecewise linear function, that is, \( \lambda_i \) is replaced by the so-called cutting-plane model
\[ \lambda_i^{\ast}(x^k) = \max_{j \in J^k} \{ \lambda_i(x^k) + g_j^T(x - x^k) \} \]
\[ = \max_{j \in J^k} \{ \lambda_i^{\ast}(x^k) + g_j^T(x - x^k) - \alpha_j^k \}, \] (19)

where
\[ \alpha_j^k = \lambda_i(x^k) - \lambda_i(y^j) - g_j^T(x^k - y^j) \text{ for all } j \in J^k \]
is the linearization error. Nonconvexity brings many difficulties when compared with the convex case. If \( l = 1 \) (i.e., the maximum eigenvalue function), \( \lambda^k_i \) is an underestimate for \( \lambda_i \) and \( \alpha^k_j \geq 0 \) for all \( j \in J^k \), which measures how good an approximation the model is to the original problem. In the nonconvex (D.C.) case, these facts are not valid anymore, and it may yield negative linearization error value which has a great influence on the convergence of algorithm. To overcome this difficulty, the linearization error \( \alpha^k_j \) can be replaced by the so-called subgradient locality measure

\[
\beta^k_j = \max\{ |\alpha^k_j|, \gamma \|x^k - y^j\|^2 \},
\]

where \( \gamma \geq 0 \) is the distance measure parameter (\( \gamma = 0 \) if \( l = 1 \)). Then obviously \( \beta^k_j \geq 0 \) for all \( j \in J^k \) and \( \lambda^k_i(x) = \max\{ \lambda_i(x^k) + g^T_j(x - x^k) - \beta^k_j \} \leq \lambda_i(x) \).

For considerations on computing and storage, we adopt the following so-called subgradient locality measure strategy

\[
\beta^k_j = \max\{ |\lambda_i(x^k) - f^k_j|, \gamma (s^k_j)^2 \},
\]

where \( s^k_j = \|x^j - y^j\| + \sum_{i=j}^{k-1} \|x^{i+1} - x^i\| \) for \( j < k \), \( s^k_k = \|x^k - y^k\| \), and \( f^k_j = \lambda_i(y^j) + g^T_j(x^k - y^j) \), so the quality \( \beta^k_j \) indicates how far \( g^j \) is from \( \partial \lambda_i(x^k) \). Hence, the \( k \)-th search direction is obtained by solving the following problem:

\[
\min \lambda^k_i(x^k + d) + \frac{1}{2} \mu_k \|d\|^2 \quad \text{over all} \quad d \in \mathbb{R}^m,
\]

where \( \lambda^k_i(x) = \max\{ \lambda_i(x^k) - \beta^k_j + g^T_j(x - x^k) : j \in J^k \} \) is in terms of the \( k \)-th polyhedral approximation to \( \lambda_i \). In convex case, \( (22) \) reduces to the search direction finding subproblem of the method in \( [17] \). These observations give ground for deriving similar processes described therein.

**Proposition 2.** For \( \mu_k > 0 \), the unique solution of the augmented model problem \( (22) \) is

\[
d^{k+1} = -\frac{1}{\mu_k} \sum_{i \in J^k} \theta_i g_i,
\]

where \( \bar{\theta} \in \mathbb{R}^k \) solves

\[
\min_{\theta \in \Delta^k} \frac{1}{2} \| \sum_{i \in J^k} \theta_i g_i \|^2 + \mu_k \sum_{i \in J^k} \theta_i \beta^k_i.
\]

Furthermore, there holds

\[
\hat{\lambda}^k_i(y^{k+1}) = \lambda_i(x^k) - \sum_{i \in J^k} \bar{\theta}_i \beta^k_i - \frac{1}{\mu_k} \| \sum_{i \in J^k} \bar{\theta}_i g_i \|^2
\]

and the primal-dual relation

\[
\epsilon_k := \bar{\theta}^T \beta(k) = \delta_k - \frac{1}{2\mu_k} \| g_k \|^2,
\]

where

\[
\bar{g}_k := \sum_{i \in J^k} \bar{\theta}_i g_i, \quad \beta(k) := [\beta^k_1, \ldots, \beta^k_{|J^k|}],
\]

\[
\delta_k := \lambda_i(x^k) - \hat{\lambda}^k_i(y^{k+1}) - \frac{\mu_k}{2} \| y^{k+1} - x^k \|^2.
\]
Next proximal bundle algorithm for arbitrary eigenvalue function is organized as follows.

**Begin of the algorithm**

**Algorithm 3.1** (Proximal bundle method for D.C. eigenvalue function)

**Step 0.** (Initialization) Given a starting point \(x^1\), together with a stopping tolerance \(\delta \geq 0\). Choose a spring-strength \(\mu_1 > 0\), the tolerance \(\bar{\epsilon} \geq 0\), and a descent-coefficient \(m_L \in (0, 1)\). Initialize the iteration-counter \(k = 1\) and set \(y^1 = x^1\); compute \(\lambda_i(y^1)\) and \(g_j \in \partial \lambda_i(y^1)\).

**Step 1.** Compute \(\bar{\theta}\) by solving (24) and set \(\bar{g}_k := \sum_{i \in J^k} \bar{\theta}_i g_i\), \(y^{k+1} := x^k - \frac{1}{\mu_k} \bar{g}_k\), \(\delta_k := \lambda_i(x^k) - \hat{\lambda}_i^k (y^{k+1}) - \frac{1}{2} \mu_k \|y^{k+1} - x^k\|^2\).

**Step 2.** (Stopping criterion). If \(\delta_k \leq \delta\), stop.

**Step 3.** Call the oracle to obtain \(\lambda_i(y^{k+1})\) and \(g_{k+1} \in \partial \lambda_i(y^{k+1})\). If \(\lambda_i(y^{k+1}) \leq \lambda_i(x^k) - m_L \delta_k\), set \(x^{k+1} = y^{k+1}\) (serious step). Otherwise set \(x^{k+1} = x^k\) (null step). Update \(\mu_k\) to \(\mu_{k+1}\).

**Step 4.** Replace \(k\) by \(k + 1\) and loop to Step 1.

**End of the algorithm**

It is not hard to find that the number of \(J^k\) increases as the iterations goes along. In practice this strategy presents serious problems with storage and computation after a large number of iterations. To overcome this drawback, we shall present the subgradient aggregation strategy, which is to aggregate the constraints generated by the subgradients and thus to keep the number of constraints bounded.

4. **Proximal bundle algorithm with subgradient aggregation.** In the last section, at each iteration of Algorithm 3.1, every previously computed subgradient generates one linear inequality in the current search direction finding subproblem; there are many inequalities at the \(k\)-th iteration. When the size of the bundle becomes too big, it is necessary to compress it and clean the model. Here we introduce rules for aggregating the past subgradient information, i.e., this leads to the concept of a first-order proximal bundle algorithm with subgradient aggregation.

First we describe search direction finding subproblems based on subgradient aggregation. For convenience, we suppose that \(f\) is convex and consider the \(k\)-th iteration of the method with subgradient aggregation. For search direction finding the scheme replaces the past subgradients \((g_j = g(y^j) \in \partial \lambda_i(y^j), f_j^k = \lambda_i(y^j) + g_j^T(x^k - y^j)), j = 1, \ldots, k-1\) by just one aggregate subgradient \((p^{k-1}, f_p^k)\), which is their convex combination

\[
(p^{k-1}, f_p^k) \in \text{conv}\{(g_j, f_j^k) : j = 1, \ldots, k-1\}
\] (27)

calculated at the \((k-1)\)-th iteration. The former linearizations of \(\lambda_i\)

\[
f_j(x) = \lambda_i(y^j) + (g_j, x - y^j)
= f_j^k + (g_j, x - x^k)
\]
and the \((k - 1)\)-th aggregate linearization
\[
\hat{\lambda}_l^{k-1}(x) = f_p^k + \langle p^{k-1}, x - x^k \rangle
\]
satisfies
\[
\lambda_l(x) \geq f_j(x) \quad j = 1, \cdots, k
\]
\[
\lambda_l(x) \geq \hat{\lambda}_l^{k-1}(x)
\]
for all \(x \in \mathbb{R}^m\) when \(l = 1\). Therefore the \(k\)-th aggregate polyhedral approximation \(\hat{\lambda}_{agg}^k\) to \(f\), defined by choosing a set \(J^k \subset \{1, \cdots, k\}, k \in J^k\),
\[
\hat{\lambda}_{agg}^k(x) = \max\{\lambda_l^{k-1}(x), f_j(x) : j \in J^k\}
\]
is a lower approximation to \(\lambda_l\): \(\hat{\lambda}_{agg}^k(x) \leq \lambda_l(x)\) for all \(x \in \mathbb{R}^m\).

\(\hat{\lambda}_{agg}^k\) can be expressed as
\[
\hat{\lambda}_{agg}^k(x) = \max\{\lambda_l(x^k) - (\lambda_l(x^k) - f_p^k) + \langle p^{k-1}, x - x^k \rangle, \lambda_l(x^k) - (\lambda_l(x^k) - f_j^k) + \langle g_j, x - x^k \rangle : j \in J^k\},
\]
where the linearization errors satisfy
\[
\lambda_l(x^k) - f_p^k \geq 0, \quad \lambda_l(x^k) - f_j^k \geq 0, \quad j \in J^k.
\]

We extend the above construction to the D.C. case. Suppose that at the \(k\)-th iteration we have the \((k - 1)\)-th aggregate subgradient \((p^{k-1}, f_p^k, s_p^k) \in \mathbb{R}^m \times \mathbb{R} \times \mathbb{R}\) that satisfies the following generalization of (27):
\[
(p^{k-1}, f_p, s_p^k) = \sum_{j=1}^{k-1} \hat{\theta}_j^{k-1}(g_j, f_j^k, s_j^k),
\]

where \(\hat{\theta}_j^{k-1} \geq 0, \text{ for } j = 1, \cdots, k - 1, \sum_{j=1}^{k-1} \hat{\theta}_j^{k-1} = 1\). We define the following aggregate subgradient locality measures
\[
\beta_p^k = \max\{|\lambda_l(x^k) - f_p^k|, \gamma(s_p^k)^2\}.
\]
The value of \(\beta_p^k\) shows how far \(p^{k-1}\) is from \(\partial \lambda_l(x^k)\). Moreover, only local subgradients \(g_j\) with small values of \(s_j^k\) contribute to \(p^{k-1}\). Hence, in this case \(p^{k-1}\) is close to \(\partial \lambda_l(x^k)\) by the local upper semicontinuity of \(\partial \lambda_l\).

So we may define the \(k\)-th aggregate polyhedral approximation to D.C. function \(\lambda_l\) as follows
\[
\hat{\lambda}_{agg}^k(x) = \max\{\lambda_l(x^k) - \beta_p^k + \langle p^{k-1}, x - x^k \rangle, \lambda_l(x^k) - \beta_j^k + \langle g_j, x - x^k \rangle : j \in J^k\}
\]
and use it for finding the \(k\)-th search direction \(d^k\) that solves the problem
\[
\min_{d \in \mathbb{R}^m} \hat{\lambda}_{agg}^k(x^k + d) + \frac{1}{2} \mu_k ||d||^2.
\]
The above problem \([33]\) can be recast as the following quadratic programming problem for \((d^k, \hat{\omega}^k) \in \mathbb{R}^m \times \mathbb{R}\):
We have completed the recursion, so these subgradients $(36)$ yield

$$
\begin{align*}
\min_{(d, \hat{w}) \in \mathbb{R}^{m+1}} & \quad \frac{1}{2} \mu_k ||d||^2 + \hat{w} \\
\text{s.t.} & \quad -\beta^k_j + \langle g^*_j, d \rangle \leq \hat{w}, \quad j \in J^k, \\
& \quad -\beta^k_p + \langle p^{k-1}, d \rangle \leq \hat{w},
\end{align*}
\tag{34}
$$

then the variable $\hat{w}^k = \lambda^*_l(k)(x^k + d^k) - \lambda_l(x^k)$ approximates the derivative of $\lambda_l$ at $x^k$ in the direction $d^k$, which can be used for line searching.

Next we show how to update the aggregate subgradient recursively. Set $\theta^k_j, j \in J^k$, and $\theta^k_p$ be the Lagrange multipliers of the $k$-th subproblem $(34)$. These multipliers form a convex combination

$$
\theta^k_j \geq 0, \quad j \in J^k, \quad \theta^k_p \geq 0, \quad \sum_{j \in J^k} \theta^k_j + \theta^k_p = 1,
$$

where the sets $\{J^k\}$ are selected recursively so that

$$
k + 1 \in J^{k+1} \subset J^k \cup \{k + 1\}, \quad |J^{k+1}| \leq M - 1,
$$

and $M \geq 2$ is a user-supplied, fixed upper bound on the number of subgradient. Then one can control the size of subproblem $(34)$.

Therefore, the current subgradient is

$$
(p^k, f^k_{p, s}^k) = \sum_{j \in J^k} \theta^k_j (g^*_j, f^k_j, s^k_j) + \theta^k_p (p^{k-1}, f^k_p, s^k_p),
\tag{35}
$$

and this leads to the following property according to $(30)$

$$
(p^k, f^k_{p, s}^k) \in \text{conv}\{(g^*_j, f^k_j, s^k_j) : j = 1, \cdots, k\}.
\tag{36}
$$

We can update the linearization values by letting

$$
\begin{align*}
{f}^k_{p}^{k+1} &= {f}^k_{p} + \langle p^k, x^{k+1} - x^k \rangle, \\
{f}^k_{j}^{k+1} &= {f}^k_{j} + \langle g^*_j, x^{k+1} - x^k \rangle, \quad j \in J^k.
\end{align*}
$$

Because $s^{k+1}_j = s^k_j + ||x^{k+1} - x^k||$ for $j \in J^k$, we will set $s^{k+1}_p = s^k_p + ||x^{k+1} - x^k||$.

So the above updating formulas and $(36)$ yield

$$
(p^k, f^k_{p, s}^{k+1}, s^{k+1}_p) \in \text{conv}\{(g^*_j, f^k_j, s^k_j) : j = 1, \cdots, k\} \setminus J^k.
$$

We have completed the recursion, so these subgradients $(g^*_j, f^k_j, s^k_j)$ for $j \in \{1, \cdots, k\} \setminus J^k$, need not be stored.

Now we state an algorithmic procedure for solving the arbitrary eigenvalue problem considered.

**Begin of the algorithm**

**Algorithm 4.1**

Step 0 (Initialization). Select the starting point $x^1 \in \mathbb{R}^n$ and a final accuracy parameter $\epsilon > 0$. Choose fixed positive line search parameters $m_L, m_R, m_\beta, \bar{a}$ and $\bar{b}, \bar{e} \leq 1$ and $m_L + m_\beta < m_R < 1$, and a distance measure parameter $\gamma > 0$, an initial weight $\mu_1 > 0$, a lower bound for weights $\mu_{\min} > 0$. Set

$$
y^1 = x^1, p^0 = g_1 = g(y^1), f^1_{p} = f^1_{1} = \lambda_1(y^1), s^1_p = s^1_1 = 0, J^1 = \{1\}.
\tag{37}
$$

Set $a^1 = 0$ and the reset indicator $r^1_a = 1$. Set the counter $k = 1$. 

Step 1 (Direction finding). Find the solution \((d, \hat{w})\) to the following \(k\)-th quadratic programming problem

\[
\begin{cases}
\min_{(d, \hat{w}) \in \mathbb{R}^{m+1}} \frac{1}{2} \mu_k ||d||^2 + \hat{w} \\
\text{s.t.} \quad -\beta_k^f + \langle g_j, d \rangle \leq \hat{w}, j \in J^k, \\
\quad -\beta_p^k + \langle p^{k-1}, d \rangle \leq \hat{w}, \quad r_a^k = 0,
\end{cases}
\]

where

\[
\beta_j^k = \max\{ |\lambda_i(x^k) - f_j^k|, \gamma(s_j^k)^2 \}, \quad \text{for} \quad j \in J^k;
\]
\[
\beta_p^k = \max\{ |\lambda_i(x^k) - f_p^k|, \gamma(s_p^k)^2 \}.
\]

Compute the Lagrange multipliers \(\theta_j^k, j \in J^k\), and \(\theta_p^k\) of \((38)\), setting \(\theta_p^k = 0\) if \(r_a^k = 1\). Set

\[
(p^k, \tilde{p}^k, \tilde{s}^k) = \sum_{j \in J^k} \theta_j^k (g_j, f_j^k, s_j^k) + \theta_p^k (p^{k-1}, f_p^k, s_p^k),
\]

If \(\theta_p^k = 0\) set

\[
a^k = \max\{ s_j : j \in J^k \}.
\]

Set

\[
\delta_k = \frac{1}{2 \mu_k} ||p||^2 + \tilde{\beta}_p^k.
\]

Step 2 (Stopping criterion). If \(\delta_k \leq \epsilon\), stop. Otherwise, go to Step 3.

Step 3 (Line search). By a line search procedure as given below, find two step sizes \(t_L^k\) and \(t_R^k\) such that \(0 \leq t_L^k \leq t_R^k\), and the two corresponding points defined by

\[
x^{k+1} = x^k + t_L^k d^k, \quad y^{k+1} = x^k + t_R^k d^k
\]
satisfy \(t_L^k \leq 1\) and

\[
\lambda_i(x^{k+1}) \leq \lambda_i(x^k) - m_L t_L^k \delta_k,
\]

\[
t_R^k = r_L^k \quad \text{if} \quad t_L^k \geq \bar{t} \quad \text{or} \quad \beta(x^k, x^{k+1}) > m_R \delta_k,
\]

\[
- \beta(x^{k+1}, y^{k+1}) + \langle g(y^{k+1}), d^k \rangle \geq m_R \delta_k \quad \text{if} \quad t_L^k < \bar{t},
\]

\[
||y^{k+1} - x^{k+1}|| \leq \bar{a}/2,
\]

where \(\beta(x, y) = \max\{ |\lambda_i(y) + \langle g(y), x - y \rangle|, \gamma ||x - y||^2 \} \).

Step 4 (Subgradient updating). Select a set \(J^k \subset J^k\) and set

\[
J^{k+1} = J^k \cup \{k + 1\}.
\]

Set \(g_{k+1} = g(y^{k+1}) \in \partial \lambda_i(y^{k+1})\) and

\[
f_j^{k+1} = f_j^k + \langle g_j, x^{k+1} - x^k \rangle \quad \text{for} \quad j \in J^k,
\]

\[
f_p^{k+1} = f_p^k + p^{k+1} - x^k,
\]

\[
s_j^{k+1} = s_j^k + \|x^{k+1} - x^k\| \quad \text{for} \quad j \in J^k,
\]

\[
s_p^{k+1} = s_p^k + \|x^{k+1} - x^k\|.
\]

Step 5 (Distance resetting test). Set

\[
a^{k+1} = \max\{ a^k + \|x^{k+1} - x^k\|, s_p^{k+1} \}. \tag{56}
\]
If \( a^{k+1} \leq \bar{a} \) then set \( r_a^{k+1} = 0 \) and go to Step 7. Otherwise, set \( r_a^{k+1} = 1 \) and go to Step 6.

Step 6 (Distance resetting). Keep deleting from \( J^{k+1} \) the smallest indices until the reset value of \( a^{k+1} \) satisfies
\[
a^{k+1} = \max \{ \delta_j^{k+1} : j \in J^{k+1} \} \leq \bar{a}/2. \tag{57}
\]

Step 7 (Updating). Select \( \mu_{k+1} > 0 \). Increase \( k \) by 1 and go to Step 1.

**End of the algorithm**

Next, let us see some remarks about the algorithm.

**Remark.** (i). By Proposition 2, the \( k \)-th subproblem dual of (38) is to find values of multipliers \( \theta_j^k, j \in J^k \), and \( \theta_p^k \) to
\[
\begin{align*}
\min_{\theta} & \quad \frac{1}{2\mu_k} \sum_{j \in J^k} \theta_j g_j + \theta_p p^{-1}_k \|\| + \sum_{j \in J^k} \theta_j \beta_j^k + \theta_p \beta_p^k, \\
\text{s.t.} & \quad \theta_j \geq 0, j \in J^k, \quad \theta_p \geq 0, \quad \sum_{j \in J^k} \theta_j + \theta_p = 1, \\
& \quad \theta_p = 0 \quad \text{if} \quad r_a^k = 1.
\end{align*} \tag{58}
\]

Any solution of (58) is a Lagrange multiplier vector for (38) and it yields the unique solution \((d^k, \hat{w}^k)\) of (38) as follows
\[
d^k = -\frac{1}{\mu_k} p^k, \quad \hat{w}^k = -\{\|p^k\|^2 + 1 \sum_{j \in J^k} \theta_j^k \beta_j^k + \theta_p^k \beta_p^k\}, \tag{59}
\]
where \( p^k \) is given by (41). Moreover, any Lagrange multipliers of (38) also solve (58). In particular, they form a convex combination:
\[
\theta_j^k \geq 0, j \in J^k, \theta_p^k \geq 0, \quad \sum_{j \in J^k} \theta_j^k + \theta_p^k = 1; \quad \theta_p^k = 0 \quad \text{if} \quad r_a^k = 1. \tag{60}
\]

Thus one may equivalently solve the dual search direction finding subproblem (58) in Step 1 of Algorithm 4.1.

(ii). About the stopping criterion in Step 2, we have the following explanation. The value of the locality measure \( \beta_p^k \) given by (42) shows the distance of \( p^k \) and \( \partial \lambda_l(x^k) \). Since the value of the locality measure \( \beta_p^k \) is small, a small value of \( \delta_k \) indicates both that \( \|p^k\| \) is small and \( p^k \) is close to \( \partial \lambda_l(x^k) \). Thus 0 is close to \( \partial \lambda_l(x^k) \), i.e., \( x^k \) is an approximately stationary point. Generally, \( \delta_k \) may be thought of as a measure of stationary of \( x^k \). In the D.C.(nonconvex) case the stopping criterion is a generalization of the standard criterion of a small value of the gradient of \( \lambda_l \).

(iii). As for the line search in Step 3, it is easy to obtain that the line search is always entered with
\[
\eta^k := \hat{\lambda}^k_{\text{agg}}(x^k + d^k) - \lambda_l(x^k) \leq -\delta_k < 0. \tag{61}
\]

Hence the criterion guarantees that the objective value at \( x^{k+1} \) is significantly smaller than the one at \( x^k \) if \( x^{k+1} \neq x^k \). This ensures that the algorithm doesn’t take infinitely many serious steps \((t^k > 0)\) with no significant improvement in the objective function value, which could impair convergence.

(iv). The parameter \( \ell > 0 \) is introduced to decrease the number of function and subgradient evaluations at line searches. Here the parameter \( \ell \) distinguishes “long” serious steps with \( t^k \ell \geq \ell \), and “short” serious steps with \( 0 < t^k \ell < \ell \), for which (47) is satisfied. It will be seen that as far as convergence analysis is concerned,
short serious steps are essentially equivalent to null steps ($t^k_L = 0$). If $t^k_L \geq \bar{\ell}$, i.e., a significant decrease of the objective value occurs, there is no need for detecting discontinuities in the gradient of $\lambda_i$, so the algorithm sets $g_{k+1} = g(y^{k+1})$. On the other hand, if $t^k_L < \bar{\ell}$, which indicates that the algorithm is blocked at $x^k$ due to the nondifferentiability of $\lambda_i$, then the criterion \cite{17} ensures that the new subgradient $g_{k+1}$, with $g^{k+1}$ and $x^k$ lying on the opposite sides of a discontinuity of the gradient of $\lambda_i$, will force a significant modification of the next search direction finding subproblem. The criterion \cite{48}, which is related to the distance resetting test, prevents the algorithm from collecting the irrelevant subgradient information.

Clearly, the line search rules \cite{15,48} are so general that one can devise many procedures for implementing Step 3. For completeness, we give below a procedure for finding step sizes $t_L = t^k_L$ and $t_R = t^k_R$, which is based on the ideas of Mifflin \cite{20}.

Now we shall present a line search algorithm which finds step sizes $t^k_L$ and $t^k_R$ such that the requirements in Step 3 hold. We assume that the line search parameters are fixed: $m_L \in (0, 1/2), m_R \in (m_L, 1), \bar{\ell} \in (0, 1], \eta := 1 - \frac{1}{2} \frac{1}{1 - m_L}$.

**Begin of the algorithm**

**Algorithm 4.2 Line Search**

Step 1. Set $t^0_L := 0$ and $t := t_U := 1$.

Step 2. If

$$
\lambda_i(x^k + td^k) \leq \lambda_i(x^k) - m_L t \delta_k,
$$

set $t^k_L := t$, otherwise set $t_U := t$.

Step 3. If $t^k_L \geq t$ or $\beta(x^k + td^k) > m_R \delta_k$, set $t^k_L := t^k_L$ and stop. Otherwise calculate $\xi \in \partial \lambda_i(x^k + td^k)$ and

$$
\beta := \max\{|\lambda_i(x^k + t^k_L d^k) - \lambda_i(x^k + td^k) + (t - t^k_L) \xi^T d^k|, \gamma (t - t^k_L)^2 \|d^k\|^2\}. \quad (62)
$$

Then if

$$
-\beta + \xi^T d^k \geq - m_R \delta_k,
$$

set $t^k_R := t$ and stop.

Step 4. If $t^k_L := 0$, then set

$$
t := \max\{\eta \cdot t_U, \frac{\frac{1}{2} t^k_L \delta_k}{t_U \delta_k - \lambda_i(x^k) + \lambda_i(x^k + t_U d^k)}\} \quad (64)
$$

and if $t^k_L > 0$, then set $t := \frac{1}{2} (t^k_L + t_U)$.

Step 5. Go to Step 2.

**End of the algorithm**

The last but not least important question concerning the proximal bundle method is the choice of the weight $\mu_k$. We can adopt the simplest strategy: keep it constant $\mu_k \equiv \mu_{fix}$. However, this will lead to several difficulties. Due to the relationship \cite{59} in Remark (i), we observe that: (i) If $\mu_{fix}$ is very large, the values of $\delta_k$ and $\|d_k\|$ will be small, almost all steps are serious and the descent becomes slow; (ii) If $\mu_{fix}$ is very small, the values of $\delta_k$ and $\|d_k\|$ will be large, each serious step will be followed by many null steps. Therefore, we consider $\mu_k$ as a variable and update it when necessary for these reasons. In what follows we shall present a safeguarded quadratic interpolation technique by Kiwiel \cite{17} for updating $\mu_k$.

We denote by $\xi^k_p$ the variation estimate which corresponds to the size of $\|p^k\| + \bar{\beta}^k_p$ and by $t^k_L$ the step counter which counts the number of long serious steps with $t^k_L = 1$
and null steps since the latest change of $\mu_k$. These variables are initialized by

$$\varepsilon^k_v := +\infty, \quad i^k_v := 0.$$  

In addition, we define $\mu^{int}_{k+1} := 2\mu_k (1 + [\lambda_i(y^{k+1}) - \lambda_i(x^k)]/\delta_k)$.

**Begin of the algorithm**

**Algorithm 4.3 Weight Updating**

1. Set $\mu := \mu_k$.
2. If $t^k_L \in (0,1)$, stop.
3. If $t^k_L = 0$ go to Step 5.
4. Set $\varepsilon^{k+1}_v := \max\{\varepsilon^k_v, 2\delta_k\}$.
   If $i^k_\mu > 0$ and $\lambda_i(y^{k+1}) \leq \lambda_i(x^k) - m_R \delta_k$, set $\mu := \mu^{int}_{k+1}$; otherwise, if $i^k_\mu > 3$ set $\mu := \mu/2$. Set
   $$\mu_{k+1} := \max\{\mu, \mu_k/10, \mu_{\min}\},$$
   $$i^{k+1}_\mu := \max\{i^k_\mu + 1, 1\}.$$  

   If $\mu_{k+1} \neq \mu_k$ set $i^{k+1}_v := 1$. Stop.
5. Set $\varepsilon^{k+1}_v := \min\{\varepsilon^k_v, \|p_k\| + \beta_l^k\}$.
   If $\beta_l^k < -3$ and $\beta_{k+1}^l > \max\{\varepsilon^{k+1}_v, 10\delta_k\}$, set $\mu := \mu^{int}_{k+1}$. Set
   $$\mu_{k+1} := \min\{\mu, 10\mu_k\},$$
   $$i^{k+1}_\mu := \min\{i^k_\mu - 1, -1\}.$$  

   If $\mu_{k+1} \neq \mu_k$ set $i^{k+1}_v := -1$. Stop.

**End of the algorithm**

5. **Convergence analysis.** In this section, we will study the convergence of Algorithm 4.1. First the convergent result is given. Before presenting our results, let us see a proposition, which is stated in Ref. [4].

**Proposition 3.** If the locally Lipschitz continuous function $f : R^m \to R$ attains a local minimum at $x^*$, then

$$0 \in \partial f(x^*).$$  

(65)

When $l > 1$, the eigenvalue function $\lambda_i$ is not convex any more, and the optimality condition of Proposition [3] is not sufficient, so the methods cannot guarantee even local optimality of the solution. Hence we can only look for some candidates, called stationary points satisfying the above condition [65]. We suppose that each execution of Line Search Procedure 4.2 is finite. Naturally, the convergence results assume that the final accuracy tolerance $\epsilon$ is set to zero. In the absence of convexity, we will content ourselves with finding the stationary points for $\lambda_i$. Our principal result states that Algorithm 4.1 either terminates at a stationary point or generates an infinite sequence $\{x^k\}$ whose accumulation points are stationary for $\lambda_i$.

For finite termination case, we have the following conclusion.

**Lemma 5.1.** Suppose that $k \geq 1$ is such that Algorithm 4.1 did not stop before the $k$-th iteration. Then there exist the numbers $\mu^k_i$ and the vectors $(y^{k,i}, f^{k,i}, s^{k,i}) \in R^m \times R \times R$, $i = 1, \ldots, M$, satisfying

$$j^k_p = \sum_{i=1}^M \tilde{\mu}^k_i (g_{f}(y^{k,i}), f^{k,i}, s^{k,i}), \quad \mu^k_i \geq 0, i = 1, \ldots, M, \quad \sum_{i=1}^M \tilde{\mu}^k_i = 1;$$

(66)  

$$(g_{f}(y^{k,i}), f^{k,i}, s^{k,i}) \in \{(g_{f}(y^j), f^j, s^j) : j \in J^k_p\}, i = 1, \ldots, M;$$

(67)
Proof. If the algorithm stops at Step 2, due to 
\(\epsilon\), then, for \(\delta_k = \frac{1}{2}||p^k||^2 + \beta_k^p \geq 0\) and 
\(\beta_k^p \geq 0\), we have \(p^k = 0\) and \(\beta_k^p = \max\{|\lambda_i(x^k) - f_p^k|, \gamma(s_k^p)^2\} = 0\), hence 
\(\lambda_i(x^k) = f_p^k\) and \(s_k^p = 0\). Combining this with \(\delta_k \leq \epsilon\), we find that the assumption of Lemma 5.2 is satisfied by \(x^k\), \(p^k\), \(f_p^k\), \(s_k^p\), etc. Therefore 
0 = \(p^k \in \partial \lambda_1(x^k)\).

About the case of generating the infinite sequence \(\{x^k\}\), i.e., suppose that the algorithm does not terminate, the succedent result is useful:

\(\|y^{k,i} - x^k\| \leq s^{k,i}, i = 1, \cdots, M; \) (68)

where \(\mathcal{J}^k_p = \mathcal{J}^{p,(k)} \cup \{j : k_p(j) < j \leq k\}, M = m + 3, \) and \(k_p(j) = \max\{j : j \leq k \) and \(\theta^j_p = 0\} \).

Proof. It follows from Lemma 3.2 of [38], Caratheodory’s theorem, and the fact that 
g_{j} = g_{f}(p^j) \) for \(1 \leq j \leq k\).

Lemma 5.2. Suppose \(x \in R^m\) is given, and the following hypothesis is satisfied: there exist the vectors \(\bar{p}, \bar{g}, \bar{f}, \bar{s}\) for \(i = 1, \cdots, M\), and the numbers \(\bar{f}_p, \bar{f}_i, \bar{s}_i\), satisfying

\[
\langle \bar{p}, \bar{f}_p, \bar{s}_p \rangle = \sum_{i=1}^{M} \bar{\mu}_i(\bar{g}_i, \bar{f}_i, \bar{s}_i), \; \bar{\mu}_i \geq 0, i = 1, \cdots, M, \; \sum_{i=1}^{M} \bar{\mu}_i = 1, \] (69)

\[
\bar{g}_i \in \partial \lambda_i(\bar{g}_i), \; \bar{f}_i = \lambda_1(\bar{g}_i) + (\bar{g}_i, \bar{x} - \bar{g}_i), \; i = 1, \cdots, M, \] (70)

\[
\|\bar{g}_i - \bar{x}\| \leq \bar{s}_i, \; i = 1, \cdots, M, \; \lambda_1(\bar{x}) = \bar{f}_p, \; \gamma \bar{s}_p = 0. \] (71)

Then \(\bar{p} \in \partial \lambda_1(\bar{x})\).

Proof. We discuss in two cases.

(1) Suppose that \(\gamma > 0\). Set \(I = \{i : \bar{\mu}_i \neq 0\}\). By \(\bar{s}_p = 0\), \(\bar{s}_p = 0\), and \(\bar{s}_p = 0\), imply that \(\bar{g}_i = \bar{x}\) for all \(i \in I\), hence the first formula of \(\bar{s}_p = 0\) yields \(\bar{g}_i \in \partial \lambda_i(\bar{x})\) for all \(i \in I\). Thus we have

\[
\bar{p} = \sum_{i \in I} \bar{\mu}_i \bar{g}_i, \; \bar{\mu}_i > 0 \; \text{for} \; i \in I, \; \sum_{i \in I} \bar{\mu}_i = 1 \; \text{and} \; \bar{g}_i \in \partial \lambda_i(\bar{x}), \; i \in I,
\]

then \(\bar{p} \in \partial \lambda_i(\bar{x})\) by the convexity of \(\partial \lambda_i(\bar{x})\) in Definition 3.1.

(2) Suppose that \(\gamma = 0\). Then the function is convex (i.e., \(l = 1\) and \(\bar{s}_p \) gives

\[
\lambda_1(z) \geq \lambda_1(\bar{g}_i) + (\bar{g}_i, z - \bar{g}_i) = \lambda_1(\bar{x}) + (\bar{g}_i, z - \bar{x}) - (\lambda_1(\bar{x}) - \bar{f}_i), \; \text{for all} \; z \in R^m \] and \(i = 1, \cdots, M\)

according to the definition of the subdifferential. Multiplying the above inequality by \(\bar{\mu}_i\) and summing, we gain that for each \(z\)

\[
\lambda_1(z) \geq \lambda_1(\bar{x}) + \langle \bar{p}, z - \bar{x} \rangle - (\lambda_1(\bar{x}) - \bar{f}_p) = \lambda_1(\bar{x}) + \langle \bar{p}, z - \bar{x} \rangle
\]

by \(\bar{s}_p \) and \(\bar{s}_p \). So by the definition of subdifferential in convex case, \(\bar{p} \in \partial \lambda_1(\bar{x})\).

Note that the results mentioned shows that \(\gamma = 0\) only if \(l = 1\) (i.e., the maximum eigenvalue function); otherwise \(\gamma > 0\).

Theorem 5.3. If Algorithm 4.1 terminates at the \(k\)-th iteration, \(k \geq 1\), then the point \(\bar{x} = x^k\) is stationary for \(\lambda_1\).

Proof. If the algorithm stops at Step 2, due to \(\delta_k \leq \epsilon\), then, for \(\delta_k = \frac{1}{2}||p^k||^2 + \beta^k_p \geq 0\) and \(\delta^k_p \geq 0\), we have \(p^k = 0\) and \(\beta^k_p = \max\{|\lambda_i(x^k) - f_p^k|, \gamma(s^k_p)^2\} = 0\), hence \(\lambda_i(x^k) = f_p^k\) and \(s^k_p = 0\). Combining this with \(\delta_k \leq \epsilon\), we find that the assumption of Lemma 5.2 is satisfied by \(x = x^k, p = p^k, f^p = f_p^k, s^p = s^k_p, \) etc. Therefore 
0 = \(p^k \in \partial \lambda_1(x^k)\).

About the case of generating the infinite sequence \(\{x^k\}\), i.e., suppose that the algorithm does not terminate, the succedent result is useful:
Lemma 5.4. Suppose there exists some point \( \bar{x} \in \mathbb{R}^m \) such that
\[
\lim_{k \to +\infty} \inf \max \{ \delta_k, \| \bar{x} - x^k \| \} = 0, \tag{72}
\]
or equivalently, there exists an infinite set \( K \subset \{1, 2, \cdots, \} \) such that \( x^k \to \bar{x} \) and \( \delta_k \to 0 \).

Then \( 0 \in \partial \lambda_1(\bar{x}) \).

Proof. The equivalence follows from the fact that, since we always have \( \delta_k = \frac{1}{2} \| p^k \|^2 + \beta^k_p \geq 0 \), \( \delta_k \) is nonnegative for all \( k \). So it implies \( p^k \to 0 \) and \( \beta^k_p \to 0 \). For the rest proof, we can see Ref. [38]. \( \square \)

Lemma 5.5. Suppose there exists an infinite set \( K \subset \{1, 2, \cdots, \} \) such that \( x^k \to \bar{x} \).

Then (72) is satisfied.

Proof. According to Lemma 5.4, we only need to prove that \( \delta_k \to 0 \). For contradiction purposes, assume that \( \delta_k \geq \delta > 0 \) for some \( \delta \) and all large \( k \) in \( K \). For \( x^k \to \bar{x} \) and the continuity of \( \lambda_1 \), we have \( f(x^k) \to f(\bar{x}) \). \( \{ \lambda_1(x^k) \} \) is nonincreasing, so \( \lambda_1(x^k) \downarrow \lambda_1(\bar{x}) \), then (45) gives \( t^k L \delta_k \to 0 \). But \( \delta_k \geq \delta \) for all large \( k \) in \( K \), hence \( t^k L \to 0 \), and we obtain \( \| x^{k+1} - x_k \| \to 0 \) from Lemma 3.3 in [16]. Thus both \( \{ x^k \}_{k \in K} \) and \( \{ x^{k+1} \}_{k \in K} \) converge to \( \bar{x} \), and the properties of \( \beta(\cdot, \cdot) \) imply that \( \beta(x^k, x^{k+1}) \to 0 \). So we have \( t^k L < \bar{t} \) and \( \beta(x^k, x^{k+1}) < m_3 \delta_k \) for all large \( k \) in \( K \). Since \( t^k L \to 0 \) and \( \delta_k \geq \delta > 0 \) for large \( k \) in \( K \), and we obtain a contradiction with (46) and the definition of \( K \). Therefore, we have \( \delta_k \to 0 \). We complete the proof. \( \square \)

Next we show that the generated sequence in Algorithm 4.1 converges to a stationary point of of D.C. function \( \lambda_1 \).

Theorem 5.6. Each accumulation point of the sequence \( \{ x^k \} \) generated by Algorithm 4.1 is a stationary point for \( \lambda_1 \). Moreover, if the level set \( S = \{ x \in \mathbb{R}^m : \lambda_1(x) \leq \lambda_1(x^0) \} \) is bounded and the final accuracy tolerance \( \epsilon \) is positive, then Algorithm 4.1 terminates in a finite number of iterates.

Proof. Combining Lemma 5.4 with Lemma 5.5, the first assertion is holding.

For the second assertion, we prove by contradiction, then the infinite sequence \( \{ x^k \} \subset S \) would have an accumulation point, say \( \bar{x} \). Then Lemma 5.5 would yield (72) and denote
\[
\hat{\delta}^k = \frac{1}{2} \| p^k \|^2 + \sum_{j \in J^k} \theta^k_j \beta^k_j + \theta^k_p \beta^k_p \tag{73}
\]
be the optimal value of the \( k \)-th dual search direction finding subproblem [58], and \( 0 \leq \delta_k \leq \hat{\delta}^k \), and the algorithm should stop owing to \( \delta_k \leq \epsilon \) for sufficiently large \( k \), which contradicts with this assumption. \( \square \)

6. Computational experiments. In this section, for preliminary validation of our approach, we wrote a Matlab implementation of Algorithm 4.1, and analyzed its performance on some test problems. We report the preliminary numerical results on a numerical implementation of the first-order proximal bundle method (Algorithm 4.1) for constrained eigenvalue problems. The goal is to provide a proof-of-concept implementation, not a complete benchmarking of the algorithm. All numerical experiments have been implemented on a computer in Matlab 7.8.0(2012a) with
Windows 7 system Intel Core™ i5-2400 Duo CPU 3.10 GHz processor and 8.00 GB memory RAM.

We consider the following constrained eigenvalue optimization
\[
\min_{\lambda_l(A(x))}, \quad l = 1, \cdots, n
\]
\[
\text{s.t.} \quad x \geq 0 \tag{74}
\]
where \(A(x) = \sum_{i=1}^{m} x_i A_i\), and \(A_i, i = 1, \cdots, m\) are given \(n \times n\) symmetric matrices.

According to the exact penalization, problems of the form (74) may be transformed into (1), even though it may be preferable to use the structure of (74) explicitly. So problems (74) may be equivalent to (1) via the exact penalization. We denote the exact penalty function
\[
P(x, \nu) := \lambda_l(A(x)) + \nu \left[ x \right]^+ = \lambda_l(A(x)) + \nu \max(0, x),
\]
where the penalty coefficient \(\nu \geq \nu^*\) for some threshold value \(\nu^* > 0\), \([x]^+\) stands for the maximum of 0 and \(x\). Hence, here we only need to solve the following problem
\[
\min_{x \in \mathbb{R}^m} P(x, \nu). \tag{75}
\]

Our tests are based on the following examples and the matrices \(A_i\) of the objective functions are generated randomly by computer. All numerical examples are of the form like (75).

The parameters have the following standard values: \(m_L = 0.01\), \(m_{\beta} = 0.1\), \(m_R = 0.5\), \(\bar{a} = 1.0 \times 10^3\), \(t = 0.01\), \(\mu_{\min} = 1.0 \times 10^{-6}\) and the stopping tolerance is taken as \(\epsilon = 10^{-6}\), i.e., optimality is declared when
\[
\delta_k \leq 10^{-6}.
\]
The first weight \(\mu_1\) is chosen by \(\mu_1 := \|g^1\|\). The distance measure parameter \(\gamma = 1.0\). In the implementation, optimality is declared when the stopping criterion is satisfied.

Table 1 reveals some relevant numerical results for the problem. As Boyd et al. proposed in Ref. [34], the second eigenvalue has special significance in some practical problems. So the second largest eigenvalue will be computed here. For each run, and for this algorithm, we give the total number of function and subgradient evaluation (i.e., calls to the oracle), denoted by \(\#f/g\). All the coefficients of symmetric matrices \(A_i\) are randomly generated by computer. We show some relevant data for the problems described above including the dimensionality of the problem \((n)\), \(n_n\) and \(n_d\) respectively stand for the numbers of iterations of null steps and descent steps, the computed solutions \(x_k\) and the computed function values \(f(x_k)\), and Time represents cpu running time, \(x^0\) is the starting point, \(k\) is the total number of iterations. The optimal solutions and optimal function values are calculated, Res stands for the relative error, defined by
\[
\text{Res} = \frac{\lambda_l(x_k) - \lambda_l(x^*)}{\lambda_l(x^0) - \lambda_l(x^*)}.
\]

Since Algorithm 4.1 presented in this paper focuses on minimizing the arbitrary eigenvalue function in which exists some structure, there exist limited testing methods suitable for comparison. Moreover, numerical experiments were carried out with different computers, different codings and precisions, hence it can be considered as an indication of performance for evaluating the merits of the compared algorithms.

We compare the numerical implementation of our algorithm with the generalized cutting plane method [16] and the corresponding results are listed in Table 2. The
Table 1. Numerical results for the second largest eigenvalue

| n  | n_a | n_d | # f/g | x^k            | Time  | f x^k          | Res  |
|----|-----|-----|-------|----------------|-------|----------------|------|
| 2  | 11  | 22  | 1e−06 | (0.4508, 0.0685) | 0.6093 | 2.8348e-07    | 4.3320e-07 |
| 3  | 15  | 25  | 1e−06 | (0.2321, 0.3796) | 0.8073 | 1.9516e-07    | 2.9823e-07 |
| 4  | 11  | 24  | 1e−06 | (0.3036, 0.4796) | 0.7225 | 2.4712e-07    | 3.7764e-07 |
| 5  | 17  | 21  | 1e−06 | (0.4569, 0.7587) | 0.8452 | 3.8994e-07    | 5.9589e-07 |
| 6  | 12  | 23  | 1e−06 | (0.3141, 0.2245) | 0.6242 | 2.0441e-07    | 3.1237e-07 |
| 7  | 27  | 24  | 1e−06 | (0.3944, 0.0098) | 0.8933 | 2.2090e-07    | 3.3757e-07 |
| 8  | 15  | 26  | 1e−06 | (0.3177, 0.5355) | 0.6810 | 2.7525e-07    | 4.2062e-07 |
| 9  | 24  | 284 | 1e−06 | (0.4139, 0.0045) | 0.7899 | 2.2825e-07    | 3.4228e-07 |
| 10 | 22  | 302 | 1e−06 | (0.0770, 0.2213) | 0.7927 | 1.2711e-07    | 1.9425e-07 |
| 11 | 16  | 26  | 1e−06 | (0.1415, 0.1247) | 0.9039 | 9.1817e-08    | 1.4031e-07 |
| 12 | 15  | 24  | 1e−06 | (0.3637, 0.0449) | 0.6943 | 2.2398e-07    | 3.7402e-07 |

Table 2. Comparison results of Algorithm 4.1 and Generalized cutting plane method (GCPM)

| n  | # f/g | Time | Res     | # f/g | Time | Res     |
|----|-------|------|---------|-------|------|---------|
| 5  | 214   | 0.6762 | 4.7140e-06 | 338   | 1.4614 | 4.7709e-06 |
| 10 | 246   | 0.8603 | 4.4696e-06 | 308   | 1.2572 | 4.8573e-06 |
| 15 | 234   | 0.7433 | 3.8621e-06 | 420   | 1.5076 | 4.8546e-06 |
| 20 | 266   | 0.8157 | 4.2591e-06 | 418   | 1.4537 | 4.6413e-06 |
| 25 | 235   | 0.7141 | 3.8623e-06 | 398   | 1.4743 | 4.7922e-06 |
| 30 | 290   | 0.7934 | 4.2803e-06 | 406   | 1.3006 | 5.4616e-06 |
| 35 | 258   | 0.7645 | 4.1979e-06 | 414   | 1.2542 | 4.8415e-06 |
| 40 | 290   | 0.8635 | 4.4795e-06 | 398   | 1.3216 | 5.8507e-06 |
| 45 | 238   | 0.7330 | 3.8985e-06 | 566   | 1.9189 | 4.5721e-06 |
| 50 | 282   | 0.8268 | 4.5536e-06 | 532   | 1.5479 | 6.6442e-06 |

The numerical results in Table 2 show that Algorithm 4.1 outperforms better on almost all of the test problems.

In our opinion, the obtained results show a reasonable performance of our method. In the test cases, the method succeeds in obtaining a reasonably high accuracy. As a conclusion from our limited numerical experiments we can state that our method works very efficiently, and requires less function computational time, executes more descent-step and emphasize that it is worthwhile to apply our algorithm to this class of eigenvalue optimization.
7. **Conclusions.** In this paper, we mainly propose an explicit proximal bundle algorithm based on the nonsmooth analysis theory for solving a special class of eigenvalue function: arbitrary eigenvalue function \( \lambda_l \) which is a class of D.C. functions. Because the objective function is a nonsmooth, nonconvex (D.C.) function, the bundle techniques are applied to alleviate these difficulties. We design the new fast algorithm with subgradient aggregation idea applied to the class of the functions here. Moreover, we can prove its global convergence, in the sense that any accumulation point of the sequence generated by our method is stationary. In addition, some computational results show that the proposed method is effective and has good performance in practice.

Although the method described here has already been proved useful, there are some issues worth pursuing further work before the idea is complete: we only study the implementable algorithm to solve this special class of eigenvalue optimization (D.C. programming), utilizing second-order information of eigenvalue function, we will continue to try to obtain faster algorithms. Meanwhile, some improvement of our algorithm will be strived to extend to other large scale optimization problems. This, however, seems to be a nontrivial task, further improvement is expected from more sophisticated implementations. Detailed work will be researched in later papers.

**Acknowledgments.** I would like to acknowledge my advisor Zun-Quan Xia, who gives many useful suggestions on this subject. The authors would thank the editor and three anonymous referees for their valuable and helpful comments for improving the presentation of this paper.

**REFERENCES**

[1] F. Alizadeh, J.-P. A. Haeberly and M. L. Overton, *Primal-dual interior-point methods for semidefinite programming: Convergence rates, stability and numerical results*, SIAM Journal on Optimization, 8 (1998), 746–768.

[2] F. Alizadeh, *Interior point methods in semidefinite programming with applications to combinatorial optimization*, SIAM Journal on Optimization, 5 (1995), 13–51.

[3] R. Bhatia, *Matrix Analysis*, Graduate Texts in Mathematics, Vol.169, Springer-Verlag, New York, 1997.

[4] F. H. Clarke, *Optimization and Nonsmooth Analysis*, Classics in Applied Mathematics, Vol.5, Society for Industry and Applied Mathematics (SIAM), Philadelphia, 1990.

[5] F. H. Clarke, Y. S. Ledyaev, R. J. Stern and P. R. Wolenski, *Nonsmooth Analysis and Control Theory*, Graduate Texts in Mathematics, Vol. 178, Springer-Verlag, New York, 1998.

[6] J. Cullum, W. E. Donath and P. Wolfe, *The minimization of certain nondifferentiable sums of eigenvalues of symmetric matrices*, Nondifferentiable Optimization, Mathematical Programming Study, 3 (1975), 35–55.

[7] K. Fan, *On a theorem of Weyl concerning eigenvalues of linear transformations. I.*, Proc. Nat. Acad. Sci. U.S.A., 35 (1949), 652–655.

[8] S. Friedland, J. Nocedal and M. L. Overton, *The formulation and analysis of numerical methods for inverse eigenvalue problems*, SIAM Journal on Numerical Analysis, 24 (1987), 634–667.

[9] M. Huang, L. P. Pang and Z. Q. Xia, *The space decomposition theory for a class of eigenvalue optimizations*, Computational Optimization and Applications, 58 (2014), 423–454.

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

[11] J.-B. Hiriart-Urruty and C. Lemaréchal, *Convex Analysis and Minimization Algorithms I*, Grundlehren der mathematischen Wissenschaften, Vols 305-306, Springer-Verlag, Berlin, 1993.
[12] J.-B. Hiriart-Urruty and D. Ye, Sensitivity analysis of all eigenvalues of a symmetric matrix. Numerische Mathematik, 70 (1995), 45–72.
[13] F. Jarre, An interior-point method for minimizing the maximum eigenvalue of a linear combination of matrices. SIAM Journal on Control and Optimization, 31 (1993), 1360–1377.
[14] M. Kojima, M. Shida and S. Shindoh, Local convergence of predictor-corrector infeasible-interior-point algorithms for SDPs and SDLCPs. Mathematical Programming, 80 (1998), 129–160.
[15] K. C. Kiwiel, An aggregate subgradient method for nonsmooth convex minimization. Mathematical Programming, 27 (1983), 320–341.
[16] K. C. Kiwiel, A linearization algorithm for nonsmooth minimization. Mathematics of Operations Research, 10 (1985), 185–194.
[17] K. C. Kiwiel, Proximity control in bundle methods for convex nondifferentiable minimization. Mathematical Programming, 46 (1990), 105–122.
[18] C. Lemaréchal, An extension of davidon methods to nondifferentiable problems. Nondifferentiable Optimization, Mathematical Programming Study, 3 (2009), 95–109.
[19] A. S. Lewis and M. L. Overton, Eigenvalue optimization. Acta Numerica, 5 (1996), 149–190.
[20] R. Mifflin, An algorithm for constrained optimization with semismooth functions. Mathematics of Operations Research, 2 (1977), 191–207.
[21] Y. Nesterov, Interior-point methods: An old and new approach to nonlinear programming. Mathematical Programming, 79 (1997), 285–297.
[22] Y. Nesterov and A. Nemirovskii, A General Approach to Polynomial-Time Algorithms Design for Convex Programming, Technical report, Centr. Econ. & Math. Inst., USSR Academy of Sciences, Moscow, USSR, 1988.
[23] Y. Nesterov and A. Nemirovskii, Interior-Point Polynomial Algorithms in Convex Programming. SIAM Studies in Applied and Numerical Mathematics, Vol. 13, Philadelphia, 1994.
[24] F. Oustry, The $U$-Lagrangian of the maximum eigenvalue function. SIAM Journal on Optimization, 9 (1999), 526–549.
[25] M. L. Overton, On minimizing the maximum eigenvalue of a symmetric matrix. SIAM Journal on Matrix Analysis and Applications, 9 (1988), 256–268.
[26] M. L. Overton and R. S. Womersley, Optimality conditions and duality theory for minimizing sums of the largest eigenvalues of symmetric matrices. Mathematical Programming, 62 (1993), 321–357.
[27] G. Pataki, On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues. Mathematics of Operations Research, 23 (1998), 339–358.
[28] E. Polak, On the mathematical foundations of nondifferentiable optimization in engineering design. SIAM Review, 29 (1987), 21–89.
[29] E. Polak and Y. Wardi, Nondifferentiable optimization algorithm for designing control systems having singular value inequalities. Automatica, 18 (1982), 267–283.
[30] S. B. Robinson, On the second eigenvalue for nonhomogeneous quasi-linear operators. SIAM Journal on Mathematical Analysis, 35 (2004), 1241–1249.
[31] R. T. Rockafellar, Convex Analysis, Princeton University Press, NJ, 1997.
[32] A. Shapiro and M. K. H. Fan, On eigenvalue optimization. SIAM Journal on Optimization, 5 (1995), 552–569.
[33] A. Shapiro, First and second order analysis of nonlinear semidefinite programs. Mathematical Programming, 77 (1997), 301–320.
[34] J. Sun, S. Boyd, L. Xiao and P. Diaconis, The fastest mixing markov process on a graph and a connection to a maximum variance unfolding problem. SIAM Review, 48 (2006), 681–699.
[35] M. Torki, First- and second-order epi-differentiability in eigenvalue optimization. Journal of Mathematical Analysis and Applications, 234 (1999), 391–416.
[36] M. Torki, Second-order directional derivatives of all eigenvalues of a symmetric matrix. Nonlinear Analysis: Theory, Methods & Applications, 46 (2001), 1133–1150.
[37] L. Vandenberghe and S. Boyd, Semidefinite programming. SIAM Review, 38 (1996), 49–95.
[38] J. Vlcek and L. Lukšan, Globally convergent variable metric method for nonconvex nondifferentiable unconstrained minimization, *Journal of Optimization Theory and Applications*, 111 (2001), 407–430.

[39] P. Wolfe, A method of conjugate subgradients for minimizing nondifferentiable functions, *Nondifferentiable Optimization, Mathematical Programming Study*, 3 (1975), 145–173.

Received April 2015; 1st revision August 2015; final revision January 2016.

E-mail address: huangming0224@163.com
E-mail address: liangxijune@163.com
E-mail address: luyuan626@aliyun.com
E-mail address: lppang@dlut.edu.cn