J. Math. Study
doi: 10.4208/jms.v48n1.15.05

Vol. **48**, No. 1, pp. 66-78
March 2015

Numerical methods for the Maxnear criterion of multiple-sets canonical analysis

Xin-Guo Liu, Jian-Ping You[*]

School of Mathematical Sciences, Ocean University of China, Qingdao 266003, Shandong, P.R. China.

Abstract. This paper deals with numerical methods for the Maxnear criterion of multiple-sets canonical analysis. Optimality conditions are derived. Upper and lower bounds of the optimal objective function value are presented. Two iterative methods are proposed. One is an alternating variable method, and the other called Gauss-Seidel method is an inexact version of the alternating variable method. Convergence of these methods are analyzed. A starting point strategy is suggested for both methods. Numerical results are presented to demonstrate the efficiency of these methods and the starting point strategy.

AMS subject classifications: 62H25, 65KO5

Key words: Multiple-sets canonical analysis, Maxnear criterion, alternating variable method, starting point strategy.

1 Introduction

Since Hotelling [3,4] proposed canonical correlation analysis (CCA) as the method for describing the relation between the scores of a set of observation units on two groups of variables, CCA has become an important method in multivariate statistics. It has been widely applied in the econometrics, signal processing, biology, artificial intelligence, and other fields. Several generalizations of canonical correlation analysis for multiple-sets have been proposed by Kettenring [6], Van de Geer [7], Hanafi and Kiers [1] and other scholars. In this paper, we shall concern ourselves with the Maxnear criterion proposed by Van de Geer [7], which can be introduced briefly as follows.

Let $y_i=(y_{i,1},\dots,y_{i,n_i})^T, i=1,\dots,m$ be m-sets of random variables. Considering $z_i(t)=t_i^T y_i, t_i\in \boldsymbol{R}^{n_i}$, which is the linear combination of $y_{i,1},\dots,y_{i,n_i}$, the basic idea of canonical correlation analysis is finding $t_1,\dots,t_m$ so as to optimize some functions of correlations or

[*]Corresponding author. *Email addresses:* liuxinguo656@sina.com (X.-G. Liu), you_jian_ping@163.com (J.-P. You)

covariances of $z_1(t),\ldots,z_m(t)$. Therefore, given the covariance matrix A of $y=(y_1,\ldots,y_m)^T$, partitioned as

$$A=(A_{ij})_{m\times m}, A_{ii}\in \boldsymbol{R}^{n_i\times n_i},$$

where A_{ii} is the covariance matrix of y_i, and $A_{ij}(i\neq j)$ is the covariance matrix between y_i and y_j . Suppose A is symmetric and positive definite in the following, and let

$$n=n_1+\ldots+n_m,\;\; D=diag(A_{11},\ldots,A_{mm}).$$

The Maxnear criterion can be described as the following optimization problem:

$$\min\, x^T(mD-A)x,\;\; s.t.\; x\in\Sigma_m, \tag{1.1}$$

$$\Sigma_m=\left\{ x=\begin{pmatrix} x_1\\ \vdots \\ x_m\end{pmatrix}\in \boldsymbol{R}^n : x_i\in \boldsymbol{R}^{n_i}, \| x_i \|_2=1 \right\}.$$

Next, we briefly present a statistical property of Maxnear. Because the matrix A is symmetric and positive definite, it can be factorized as follows:

$$A=P^TP,\;\; P=[P_1,\ldots,P_m],\;\; P_j\in \boldsymbol{R}^{n\times n_j}.$$

Noting that

$$\begin{aligned} Var(y_i^Tx_i-y_j^Tx_j)&=Var(y_i^Tx_i)+Var(y_j^Tx_j)-2cov(y_i^Tx_i,y_j^Tx_j)\\ &=x_i^TA_{ii}x_i+x_j^TA_{jj}x_j-2x_i^TA_{ij}x_j,\end{aligned}$$

adding them up, we have

$$\begin{aligned}&\sum_{i,j=1}^{m} Var(y_i^Tx_i-y_j^Tx_j)\\ &=2mx^TDx-2x^TAx=2x^T(mD-A)x.\end{aligned}$$

Hence, the Maxnear is equivalent to the following optimization problem:

$$\min \sum_{i,j=1}^{m} Var(y_i^Tx_i-y_j^Tx_j),\;\; s.t.\; x\in\Sigma_m. \tag{1.2}$$

In this paper, we mainly concentrate on developing efficient algorithm for Maxnear. In fact, lacking of efficient methods is one obstacle of applying Maxnear in practice. All general-purpose optimization algorithms applying to (1.1) are mainly centered around satisfying the first-order necessary condition(see Theorem 2.1 below). Without the global minimizer, the canonical correlation would not be established, making the statistical prediction less reliable. For general $m\geq 2$, several remarks about (1.1) are in order.

- (i)If $m > [\frac{n}{2}]$, then the feasible set of (1.1) is not connected.
- (ii)When $m = n$, (1.1) is a discrete optimization problem, equivalent to max-cut, finding its global solution is a problem of $NP-hard$.
- (iii)The feasible set for (1.1) is not convex,and (1.1) is not convex. So, there may be multiple local minimum points.

On the other hand, (1.1) have two important characteristics as follows:

- (i)Its objective function is quadratic, and it is defined by the symmetric and positive semi-definite (typically, positive definite) matrix $mD-A$. This provides the possibility for us to use numerical linear algebra techniques.
- (ii)The feasible region is a special product Stiefel manifold.

The main purpose of this article is to develop efficient methods based on the above characteristics of (1.1).

This paper is organized as follows. Section 2 focuses on the optimality conditions for (1.1). We propose two iteration methods in Section 3. In Section 4, we present upper and lower bounds of the optimal objective function value, and based on these, we suggest an efficient starting point strategy for solving (1.1). Finally, in Section 5, we present numerical tests to demonstrate the efficiency of our methods.

2 The optimality conditions

Applying the general conclusions on constrained optimization problems [8, Theorems 12.1, 12.5 and 12.6], we can get the following three results.

Theorem 2.1. *Suppose that $x^* \in \Sigma_m$ is a solution of (1.1). Then there are scalars $\lambda_1, \ldots, \lambda_m$ such that the following equation holds*

$$(mD-A)x^* = \Lambda x^* \tag{2.1}$$

where $\Lambda = diag(\lambda_1 I_{n_1}, \ldots, \lambda_m I_{n_m})$.

Theorem 2.2. *Suppose that $x^* \in \Sigma_m$ is a solution of (1.1) and (x^*, Λ) satisfies (2.1). Then*

$$\omega^T[mD-A-\Lambda]\omega \geq 0, \textit{for all } \omega \in \mathcal{F}$$

where

$$\mathcal{F} = \left\{ \omega = \begin{pmatrix} \omega_1 \\ \vdots \\ \omega_m \end{pmatrix} \in \boldsymbol{R}^n : \omega_i \in \mathbf{R}^{n_i},\ \omega_i^T x_i^* = 0,\ i = 1, \ldots, m \right\}.$$

Theorem 2.3. *Suppose that $x^* \in \Sigma_m$ and (x^*,Λ) satisfies (2.1). Suppose also that*

$$\omega^T[mD-A-\Lambda]\omega>0, \textit{for all } \omega\in\mathcal{F},\omega\neq 0.$$

Then, x^* is a strict local minimizer of (1.1).

It is worth pointing out that the shortcoming of the above three results is not providing the information of the global minimizer. The following result provides a sufficient condition.

Theorem 2.4. *Suppose that (x^*,Λ) satisfies (2.1). If $mD-A-\Lambda$ is positive semi-definite, then x^* is a global solution for (1.1).*

Proof. Let $M=mD-A$, $\rho(x)=x^TMx$. It is not difficult to see that

$$(M-\Lambda)x^*=0,\ x^T\Lambda x=x^{*T}\Lambda x^*, \text{ for } x\in\Sigma_m.$$

Consequently,

$$\rho(x)-\rho(x^*)=(x-x^*)^T(M-\Lambda)(x-x^*), \text{ for all } x\in\Sigma_m.$$

Since $M-\Lambda$ is symmetric and positive semi-definite, then we have

$$\rho(x)\geq\rho(x^*), \text{ for all } x\in\Sigma_m.$$

□

Theorem 2.5. *Suppose $m=2$, $x^*\in\Sigma_2$, and $Mx^*=\Lambda x^*$. Then the necessary and sufficient condition of x^* being a global solution for (1.1) is that $M-\Lambda$ is a positive semi-definite matrix.*

Proof. The sufficiency is obtained by Theorem 2.4. The proof of the necessity can be proved similarly to [2, the Result 4]. □

Next, we prove that $mD-A$ is a positive semi-definite(typically, positive definite) matrix. Let

$$R=D^{-\frac{1}{2}}AD^{-\frac{1}{2}}=(R_{ij})_{m\times m},\ R_{ij}=A_{ii}^{-\frac{1}{2}}A_{ij}A_{jj}^{-\frac{1}{2}}.$$

It is easy to see that $mD-A$ is a positive semi-definite matrix $\Leftrightarrow \lambda_{max}(R)\leq m$. Here, $\lambda_{max}(R)$ denotes the largest eigenvalue of R. Some estimates of $\lambda_{max}(R)$ will be given below. Since A is symmetric and positive definite, the matrix R can be decomposed as

$$R=L^TL,\ L=[L_1,\dots,L_m],\ L_j\in\boldsymbol{R}^{n_j\times n_j},\ L_j^TL_j=I_{n_j}.$$

Let

$$Rx=\lambda_{max}(R)x,\ x=\begin{pmatrix} x_1\\ \vdots\\ x_m\end{pmatrix},\ x_j\in\boldsymbol{R}^{n_j},\ \|x\|_2=1.$$

Theorem 2.6. *We have*

$$\lambda_{max}(R) \leq \left(\sum_{i=1}^{m} \| x_j \|_2 \right)^2. \tag{2.2}$$

Proof. Observe that

$$\begin{aligned} \lambda_{max}(R) &= x^T R x = \| Lx \|_2^2 \\ &\leq \left(\sum_{i=1}^{m} \| L_i x_i \|_2 \right)^2 = \left(\sum_{i=1}^{m} \| x_i \|_2 \right)^2. \end{aligned}$$

This gives the desired estimate. □

Notice that

$$\left(\sum_{i=1}^{m} \| x_i \|_2 \right)^2 \leq m,$$

which holds if and only if $\| x_1 \|_2 = \ldots = \| x_m \|_2 = \frac{1}{\sqrt{m}}$. Using this fact we can prove the following result.

Corollary 2.1.

$$\lambda_{max}(R) = m \Leftrightarrow \| x_1 \|_2 = \ldots = \| x_m \|_2 = \frac{1}{\sqrt{m}}, \text{ and } L_i x_i = L_1 x_1, \ (i = 2,3 \ldots, m).$$

Proof. From the Cauchy inequality and the proof of Theorem 2.6, we have

$$\lambda_{max}(R) = m \Leftrightarrow \| x_1 \|_2 = \ldots = \| x_m \|_2 = \frac{1}{\sqrt{m}}$$

and there exists $u_2, \ldots, u_m \geq 0$ so that

$$\begin{aligned} &L_1 x_1 + \cdots + L_{m-1} x_{m-1} = u_m L_m x_m, \\ &L_1 x_1 + \cdots + L_{m-2} x_{m-2} = u_{m-1} L_{m-1} x_{m-1}, \\ &\qquad\qquad\vdots \\ &L_1 x_1 = u_2 L_2 x_2. \end{aligned}$$

Consequently, from $u_2, \ldots, u_m \geq 0$ and $\| L_i x_i \|_2 = \| x_i \|_2 = \frac{1}{\sqrt{m}}$, we see that

$$L_i x_i = L_1 x_1, \ (i = 2, \ldots, m).$$

This completes the proof. □

Because A is positive definite, the following result is a consequence of Corollary 2.1.

Corollary 2.2. We have

$$\lambda_{max}(R) < m.$$

Let $\rho_{ij} = \| R_{ij} \|_2$. Since R is positive definite, its sub-matrix $\begin{pmatrix} I_{n_i} & R_{ij} \\ R_{ij}^T & I_{n_j} \end{pmatrix}$ is also positive definite, and therefore $\rho_{ij} < 1 \ \ (i \neq j)$. Let $\rho_{max} = \max\limits_{i \neq j} \rho_{ij}$.

Theorem 2.7. *We have*

$$\lambda_{max}(R) \leq 1 + (m-1)\rho_{max}.$$

Proof. Let

$$Rx = \lambda_{max}(R)x, \ \ x = \begin{pmatrix} x_1 \\ \vdots \\ x_m \end{pmatrix}, \ \ \| x_k \|_2 = \max_{1 \leq i \leq m} \| x_i \|_2 .$$

Noticing that

$$\lambda_{max}(R) x_k = x_k + \sum_{j=1, j \neq k}^{m} R_{kj} x_j,$$

we see that

$$\begin{aligned} \lambda_{max}(R) &= 1 + \frac{1}{\| x_k \|_2^2} \sum_{j \neq k} x_k^T R_{kj} x_j \\ &\leq 1 + \sum_{j=1, j \neq k}^{m} \rho_{kj} \left(\frac{\| x_j \|_2}{\| x_k \|_2} \right) \\ &\leq 1 + (m-1)\rho_{max}. \end{aligned}$$

This completes the proof of the theorem. □

Theorem 2.7 implies that, if ρ_{max} is not close to 1, then $\lambda_{max}(R)$ is not very close to m. Furthermore, except for the case where $\| x_1 \|_2 \approx \ldots \approx \| x_m \|_2 \approx \frac{1}{\sqrt{m}}$, $\lambda_{max}(R)$ is not very close to m.

3 Numerical methods

Notice that the feasible set for (1.1) is the product of unit spheres, naturally, we propose an alternating variable method (AVM) to solve (1.1).

Algorithm 3.1 (AVM)

(1) Take $x^{(0)} \in \Sigma_m$

(2) Iteration. Suppose that the current approximate point is $x^{(k)} = \begin{pmatrix} x_1^{(k)} \\ \vdots \\ x_m^{(k)} \end{pmatrix}$, then compute $x^{(k+1)}$ by

$$x_i^{(k+1)} = arg\min \rho(x_i^{(k+1)}, \ldots, x_{i-1}^{(k+1)}, x_i, x_{i+1}^{(k)}, \ldots, x_m^{(k)}), \; i=1,2,\ldots,m \tag{3.1}$$

If $\rho(x^{(k)}) - \rho(x^{(k+1)}) \leq \varepsilon$, then stop the algorithm, and $x^{(k+1)}$ is an approximate solution.

Obviously, Algorithm 3.1 has the following property:

$$\rho(x^{(k+1)}) \leq \rho(x^{(k)}).$$

Therefore, it is a descent algorithm. Pay attention to $\rho(x) \leq m \cdot \lambda_{min}(M)$, we see that $\{\rho(x^{(k)})\}$ is convergent.Let

$$b_i^{(k)} = \sum_{j=1}^{i-1} A_{ij} x_j^{(k+1)} + \sum_{j=i+1}^{m} A_{ij} x_j^{(k)}.$$

Then, (3.1) is equivalent to the following optimization problem:

$$x_i^{(k+1)} = arg \min_{\|x_i\|_2=1} \{x_i^T((m-1)A_{ii})x_i - 2b_i^{(k)T} x_i\}. \tag{3.2}$$

This is the least squares problem with the spherical constraint(LSS), several efficient methods are available, such as Newton method, see Nocedal and Wright [8].

On the other hand, if the Newton method is used to solve the sub-problem (3.2), it is relatively complicated contrast to the simplicity of (3.2), and there may meet hard-cases(see, [8]). Therefore, an approximate method is given below.

Notice that

$$\begin{aligned} g(x_i) &\equiv x_i^T M_{ii} x_i - 2b_i^{(k)T} x_i \\ &= g(x_i^{(k)}) + 2\left\langle M_{ii} x_i^{(k)} - b_i^{(k)}, x_i - x_i^{(k)} \right\rangle + (x_i - x_i^{(k)T}) M_{ii} (x_i - x_i^{(k)}) \\ &\leq h(x_i) \equiv g(x_i^{(k)}) + 2\left\langle M_{ii} x_i^{(k)} - b_i^{(k)}, x_i - x_i^{(k)} \right\rangle + \lambda_{max}(M_{ii}) \| x_i - x_i^{(k)} \|_2^2. \end{aligned}$$

Algorithm 3.2 (Gauss-Seidel method)

(1) Take $x^{(0)} \in \Sigma_m$

(2) Iteration. Suppose that the current approximate point is $x^{(k)} = \left(x_1^{(k)},\ldots,x_m^{(k)}\right)^T$, then compute $x^{(k+1)}$ as following:

$$x_i^{(k+1)} = arg \min_{\|x_i\|_2=1} h(x_i), \;\; i=1,2,\ldots,m \tag{3.3}$$

If $\rho(x^{(k)})-\rho(x^{(k+1)}) \le \varepsilon$, then stop the algorithm with the approximate solution $x^{(k+1)}$.

Note that $h(x_i)$ can be reformulated as

$$h(x_i) = g(x_i^{(k)}) + 2\left\langle M_{ii}x_i^{(k)} - b_i^{(k)} - \lambda_{max}(M_{ii})x_i^{(k)}, x_i - x_i^{(k)}\right\rangle, \tag{3.4}$$

by applying the constraints $x_i^T x_i = 1$ and $x_i^{(k)T} x_i^{(k)} = 1$. we see that (3.3) can be solved as follows:

$$y_i^{(k)} = M_{ii}x_i^{(k)} - b_i^{(k)} - \lambda_{max}x_i^{(k)}, \tag{3.5a}$$

$$u_i^{(k)} = \| y_i^{(k)} \|_2, \;\; x_i^{(k+1)} = -\frac{y_i^{(k)}}{u_i^{(k)}}. \tag{3.5b}$$

It is not difficult to see that $g(x_i^{(k+1)}) \le h_i(x_i^{(k+1)}) \le g(x_i^{(k)})$. So Algorithm 3.2 is a descent algorithm.

If $y_i^{(k)} = 0$, then $x_i^{(k)}$ is a stationary point of (3.2). Further, since $M_{ii} - \lambda_{max}(M_{ii}) I_{n_i}$ is negative semi-definite, $x_i^{(k)}$ is a maximum point of $g(x_i)$. For such a case, we make the following change for $x_i^{(k+1)}$:

$$x_i^{(k+1)} = b_i^{(k)} / \| b_i^{(k)} \|_2.$$

If $y_i^{(k)} \neq 0$, but
$$M_{ii}x_i^{(k)} - b_i^{(k)} - \lambda_{max}(M_{ii})x_i^{(k)} = \delta_i^{(k)} x_i^{(k)},$$

$$\delta_i^{(k)} = u_i^{(k)},$$

then $x_i^{(k)}$ is the maximum point of $g(x_i)$, and there must be

$$h(x_i^{(k+1)}) < g(x_i^{(k)}).$$

Obviously, Algorithm 3.2 is more simple and more easy to implement than Algorithm 3.1, but may be slower. To overcome this shortcoming, in the next section we introduce a starting point strategy.

4 Starting point strategy

First, it is easy to see that the following conclusion holds.

Proposition 4.1. Let $\rho_{maxnear} = \min_{x\in\Sigma_m} x^T(mD-A)x$. Then

$$(m-\lambda_{max}(R))\sum_{i=1}^{m}\lambda_{min}(A_{ii}) \le \rho_{maxnear} \le (m-\lambda_{min}(R))\sum_{i=1}^{m}\lambda_{min}(A_{ii}). \tag{4.1}$$

Here $\lambda_{min}(A_{ii})$ denotes the smallest eigenvalue of A_{ii}. In Section 2, we point out that $\lambda_{max}(R) < m$, and $\lambda_{max}(R)$ is usually not close to m.

Inspired by (4.1), we can give a method for choosing $x^{(0)}$ in Algorithm 3.2. To this end, let v_i be a unit eigenvector of A_{ii} associated $\lambda_{min}(A_{ii})$, and $G_k = (A_{1,k},\ldots,A_{k-1,k})^T$.

Strategy 4.1
Take $x_1^{(0)} = v_1$. If $y^{(k-1)} \equiv \left(x_1^{(0)},\ldots,x_{k-1}^{0}\right)^T$ has been constructed, then

$$x_k^{(0)} = \begin{cases} v_k, & \text{if } v_k^T G_k^T y^{(k-1)} \ge 0, \\ -v_k, & \text{otherwise.} \end{cases} \tag{4.2}$$

Let $M_{k-1} = (M_{ij})_{1\le i,j\le k-1}$. Then

$$y^{(k)T} M_k y^{(k)} = y^{(k-1)T} M_{k-1} y^{(k-1)} - 2x_k^{(0)T} G_k^T y^{(k-1)} + (m-1)\lambda_{min}(A_{kk}).$$

Consequently, the following result holds.

Proposition 4.2. The $x^{(0)}$ constructed from Strategy 4.1 satisfies that

$$x^{(0)T}(mD-A)x^{(0)} \le (m-1)\sum_{i=1}^{m}\lambda_{min}(A_{ii}). \tag{4.3}$$

Combining (4.3) with (4.1), it shows that $x^{(0)}$ is usually a good starting point for Algorithm 3.2. Since Algorithm 3.2 is a descent algorithm, if x^* is a limit point of $\{x^{(k)}\}$, then $\rho(x^*) \le \rho(x^{(0)})$. Therefore, this choice of $x^{(0)}$ may also enhance the possibility of getting a global solution of (1.1) by Algorithm 3.2.

5 Numerical Examples

In this section, we present our numerical experiment of Algorithm 3.2 and Strategy 4.1 to show their efficiency, and most importantly, the effectiveness of Strategy 4.1 both in convergence rate and in seeking for the global minima of the Maxnear. All of our tests were conducted in MATLAB on a PC with Intel(R) Pentium(R)4 processor of 3.20 GHZ. The defaults value of ε is 10^{-5}.

For our comparison, we first create a small example, Example 5.1. Then, in Examples 5.2 and 5.3, the matrices are of size 200×200, which were randomly generated,and we did

Table 1: Numerical results for Example 5.1

	Strategy4.1	Random Starting Point
$\bar{\rho}_{maxnear}$	36.4939	42.5103
$Avg.Iter.\sharp$	7	18.8700
$Avg.Time$	0.0115	0.0267

the test for each matrix over 100 random starting points. The test results are documented in Tables 1, 2 and 3, respectively. Under the first column are the average numbers of iterations and CPU time with Strategy 4.1. Under the second column are the average numbers of iterations and CPU time for all matrices over 100 random starting points.

Example 5.1 The matrix A is given by,

$$A=\begin{pmatrix}
7.4470 & -5.3387 & -4.8906 & 0.0903 & 2.2446 & 3.6421 & -0.1752 & -0.0246 & 1.7437 & 5.8028 \\
-5.3387 & 13.8905 & 4.5704 & -1.4263 & 0.5230 & 1.8020 & -3.7392 & 3.5696 & -1.9936 & -8.1650 \\
-4.8906 & 4.4704 & 8.8486 & -0.1523 & -0.3143 & -1.6288 & -1.3250 & 0.4012 & -2.7730 & -2.6152 \\
-0.0903 & -1.4263 & -0.1523 & 4.4653 & -4.2442 & -5.9271 & -3.8049 & -3.9631 & 5.1445 & -0.4010 \\
2.2446 & 0.5230 & -0.3143 & -4.2442 & 25.9532 & 6.8747 & -4.8952 & -0.8508 & 3.9045 & -4.8526 \\
3.6421 & 1.8020 & -1.6228 & -5.9271 & 6.8747 & 12.3494 & 2.8848 & 3.0664 & -3.8247 & 0.7834 \\
-0.1752 & -3.7392 & -1.3250 & -3.8049 & -4.8952 & 2.8848 & 11.7609 & 5.7035 & -7.7411 & 6.0963 \\
-0.0246 & 3.5696 & 0.4012 & -3.9631 & -0.8508 & 3.0664 & 5.7035 & 12.4512 & -11.9284 & 4.7289 \\
1.7437 & -1.9936 & -2.7730 & 5.1445 & 3.9045 & -3.8247 & -7.7411 & -11.9284 & 16.8287 & -5.8446 \\
5.8028 & -8.1650 & -2.6152 & -0.4010 & -4.8526 & 0.7834 & 6.0963 & 4.7289 & -5.8446 & 11.6403
\end{pmatrix}$$

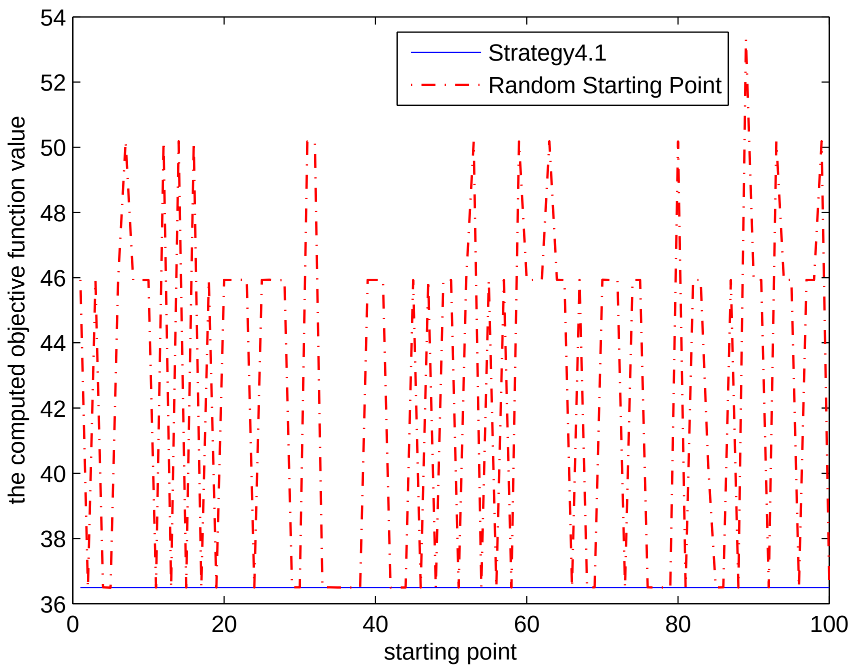

Figure 1: The objective function values $\rho_{maxnear}$

Table 2: Numerical results for Example 5.2

	Strategy4.1	Random Starting Point
$\bar{\rho}_{maxnear}$	4.6555	5.2392
$Avg.Iter.\sharp$	88.52	148.0696
$Avg.Time$	2.1837	3.6219

with $m=4, n_1=2, n_2=2, n_3=3, n_4=3$.

The numerical results are documented in Table1 and the computed objective function values are recorded in Fig. 1.

Example 5.2 In this example, we randomly generate 100 matrices with size 200×200, with $m=5, n_i=40, i=1,\ldots,5$. These matrices were constructed as follows.

$$C=rand(200,200);[Q,R]=qr(C);$$
$$\Lambda=diag(rand(1,200));A=Q*\Lambda*Q'.$$

The numerical results are documented in Table 2 and the computed objective function values for each matrix over 100 random tests are recorded in Fig. 2.

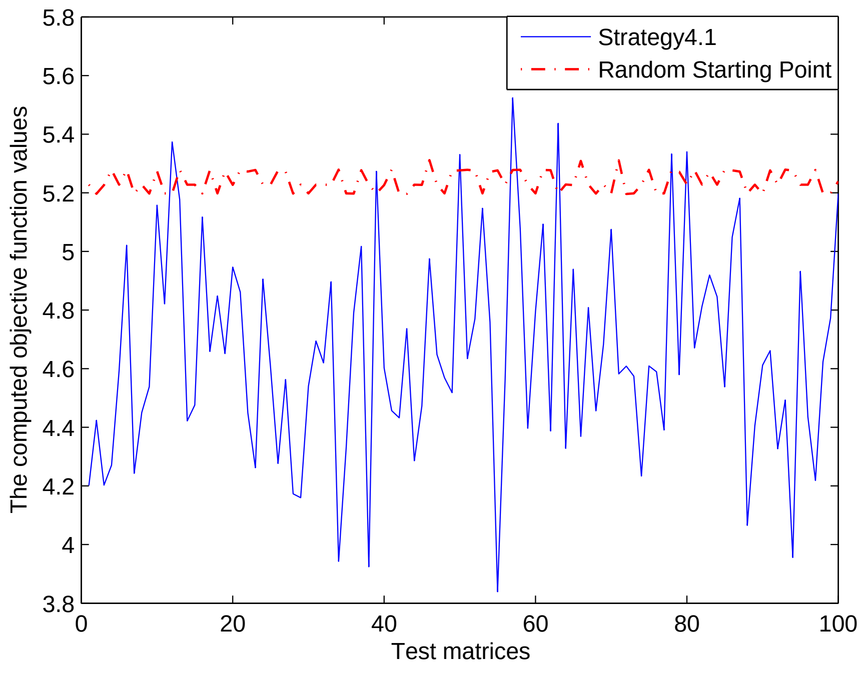

Figure 2: The computed objective function values $\rho_{maxnear}$

Table 3: Numerical results for Example 5.3

	Strategy4.1	Random Starting Point
$\bar{\rho}_{maxnear}$	7.6534	8.4600
$Avg.Iter.\sharp$	22.43	69.2027
$Avg.Time$	0.7411	2.2009

The results in Table 2 show that Strategy 4.1 enhances Algorithm 3.2 in terms of iterative steps and CPU time. On the other hand, Fig. 2 shows that about 7% of the tests Algorithm 3.2 with Strategy 4.1 did not converge to the global minima of the Maxnear.

Example 5.3 In this example, we randomly generate 100 matrices, with $m=5,n_1=20,n_i=n_1+(i-1)10,i=2,\dots,5$. These matrices were constructed as follows.

$$\begin{aligned}&C=rand(200,200);[Q,R]=qr(C);\\&\Lambda=diag(10*rand(1,50),rand(1,150));\\&A=Q*\Lambda*Q'.\end{aligned}$$

The numerical results are documented in Table 3 and the computed objective functions values for each matrix averaged over 100 random tests are recorded in Fig. 3.

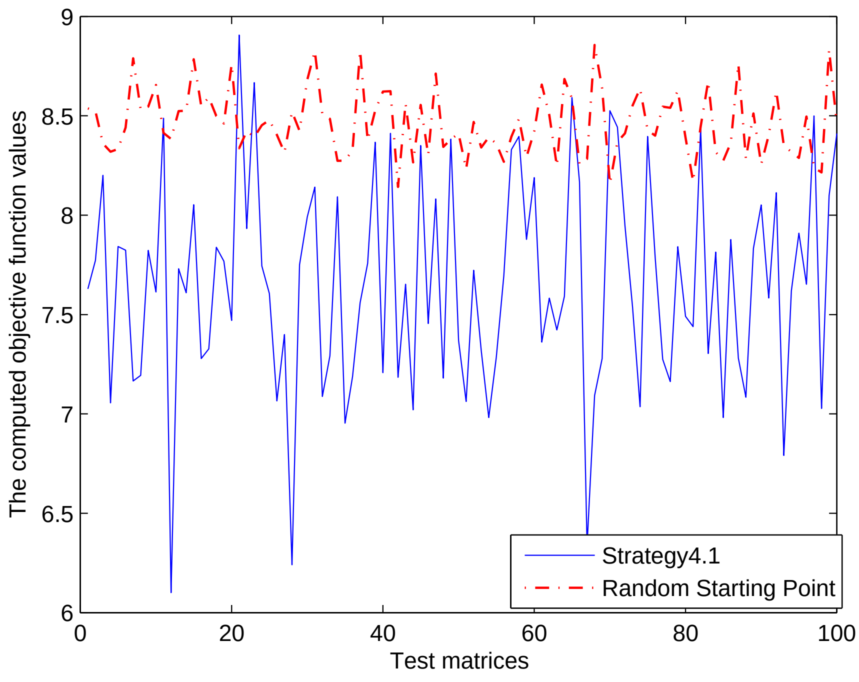

Figure 3: The computed objective function values $\rho_{maxnear}$

The results in Table 3 show that Strategy 4.1 enhanced Algorithm 3.2 in terms of iterative steps and CPU time. On the other hand, Fig. 3 shows that about 8% of the tests Algorithm 3.2 with Strategy 4.1 did not converge to the global minima of the Maxnear.

From above test results we see that Strategy 4.1 can enhance Algorithm 3.2 in terms of iteration steps and CPU time. However, it is worth pointing out that Algorithm 3.2 with Strategy 4.1 does not guarantee to converge to the global minima of the Maxnear, and further work is needed.

Acknowledgments

This work was partially supported by NSFC (No.11371333). The authors are grateful to the two referees for their valuable suggestions.

References

[1] M.Hanafi and H. A. L. Kiers. Analysis of K sets of data,with differential emphasis on agreement between and within sets. Computational Statistics and Data Analysis, 2006, 51: 1491-1508.

[2] M. Hanafi and J. M. F. Ten Berge. Global optimality of the successive Maxbet algorithm.Psychometrika, 2003, 68: 97-103.

[3] H. Hotelling. The most predictable criterion. J. Educ. Pyschol., 1935, 26: 139-142.

[4] H. Hotelling. Relations between two sets of variables. Biometrika, 1936, 28: 321-377.

[5] P. Horst. Relations among m sets of measures. Psychometrika, 1961, 26: 129-149.

[6] J. R. Kettenring. Canonical analysis of several sets of variables. Biometrika, 1971, 58: 433-451.

[7] J. P. Van De Geer. Linear relations among k sets of variables. Psychometrika, 1984, 4(9): 70-94.

[8] J. Nocedal and S. J.Wright. Numerical Optimization, Springer, New York, 1999.