Journal of Computational Mathematics
Vol.xx, No.x, 2017, 1–17.

http://www.global-sci.org/jcm
doi:??

An Inexact Smoothing Newton Method for Euclidean Distance Matrix Optimization Under Ordinal Constraints*

Qingna Li
School of Mathematics and Statistics and Beijing Key Laboratory on MCAACI, Beijing Institute of Technology, Beijing, 100081, China.
Email: qnl@bit.edu.cn
Houduo Qi
School of Mathematics, The University of Southampton, Highfield, Southampton SO17 1BJ, UK.
Email: hdqi@soton.ac.uk

Abstract

When the coordinates of a set of points are known, the pairwise Euclidean distances among the points can be easily computed. Conversely, if the Euclidean distance matrix is given, a set of coordinates for those points can be computed through the well known classical Multi-Dimensional Scaling (MDS). In this paper, we consider the case where some of the distances are far from being accurate (containing large noises or even missing). In such a situation, the order of the known distances (i.e., some distances are larger than others) is valuable information that often yields far more accurate construction of the points than just using the magnitude of the known distances. The methods making use of the order information is collectively known as non-metric MDS. A challenging computational issue among all existing nonmetric MDS methods is that there are often a large number of ordinal constraints. In this paper, we cast this problem as a matrix optimization problem with ordinal constraints. We then adapt an existing smoothing Newton method to our matrix problem. Extensive numerical results demonstrate the efficiency of the algorithm, which can potentially handle a very large number of ordinal constraints.

Mathematics subject classification: 90C30, 90C26, 90C90.
Key words: Nonmetric multidimensional scaling, Euclidean distance embedding, Ordinal constraints, Smoothing Newton method.

1. Introduction

Suppose that we are given the coordinates of a set of points, namely $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ with $\mathbf{x}_i \in \mathbb{R}^r$. It is straightforward to compute the pairwise Euclidean distances: $d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$, $i, j = 1, \ldots, n$. The matrix $D = (d_{ij}^2)$ is known as the (squared) Euclidean Distance Matrix (EDM) of those points. However, the inverse problem is more interesting and important (and challenging). Suppose D is given. The method of the classical Multi-Dimensional Scaling (cMDS) generates a set of such coordinates that preserve the pairwise distances in D. We give a short description of it below. Let

$$J := I - \frac{1}{n}\mathbf{1}\mathbf{1}^T \qquad \text{and} \qquad B := -\frac{1}{2}JDJ,$$

where I is the $n \times n$ identity matrix and $\mathbf{1}$ is the (column) vector of all ones in $\mathbb{R}^n$. In literature, J is known as the centralization matrix and B is the double-centralized matrix of D

* Received September 21, 2016 / Revised version received February 14, 2017 / Accepted February 07, 2017 /

(also the Gram matrix of D because B is positive semidefinite). Suppose B admits the spectral decomposition:

$$B = [\mathbf{p}_1, \ldots, \mathbf{p}_r] \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_r \end{bmatrix} \begin{bmatrix} \mathbf{p}_1^T \\ \vdots \\ \mathbf{p}_r^T \end{bmatrix}, \tag{1.1}$$

where $\lambda_1, \ldots, \lambda_r$ are positive eigenvalues of B (the rest are zero) and $\mathbf{p}_1, \ldots, \mathbf{p}_r$ are the corresponding orthonormal eigenvectors. Then the following coordinates $\mathbf{x}_1, \ldots, \mathbf{x}_n$ obtained by

$$[\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n] := \begin{bmatrix} \sqrt{\lambda_1} & & \\ & \ddots & \\ & & \sqrt{\lambda_r} \end{bmatrix} \begin{bmatrix} \mathbf{p}_1^T \\ \vdots \\ \mathbf{p}_r^T \end{bmatrix} \tag{1.2}$$

preserve the known distances in the sense that $\|\mathbf{x}_i - \mathbf{x}_j\| = d_{ij}$ for all $i, j = 1, \ldots, n$. This is the well known cMDS. We refer to [16, 25, 30, 31] and [4, 7, 8] for detailed description of cMDS and its generalizations.

In almost all applications, D is not fully given or contains noise and in this case D is often denoted by Δ. When Δ is not far from a true EDM, cMDS works fairly well. This has been justified in various situations including [28] on manifold learning and [9] where the noises can be bounded. Instead of directly applying cMDS on Δ, one may also modify Δ so as for it to be a true EDM. Research on this line were mainly contributed from numerical linear algebra and optimization and include e.g., [1,3,15,21,23,29]. Those methods, in addition to those discussed in [4], belong to the category of metric MDS, which means that the magnitudes of the known distances are far more important than the rest information. Metric MDS works well when the noise in D is not at a very high level.

When D contains inaccurate distances whose errors cannot be regarded to be from random noises, the magnitudes of the erroneous distances may have a disastrous effect on the embedding constructions. Let us take a look at a simple example, which clearly demonstrates the undesired effect. Network (a) in Fig. 1.1 is a true square network with the first point being in the center. The true distance matrix is

$$D = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 2 & 4 & 2 \\ 1 & 2 & 0 & 2 & 4 \\ 1 & 4 & 2 & 0 & 2 \\ 1 & 2 & 4 & 2 & 0 \end{bmatrix}.$$

Now suppose the distances from the center to the rest points are messed up with other distances. For example, $D_{12} = D_{14} = 2$ and $D_{13} = D_{15} = 4$. Comparing with the true distance, which is 1, the errors are large. We apply several well-known methods to the corresponding Δ matrix and the reconstructed networks are shown from (b) to (f) in Fig. 1.1. It can be seen that the smoothing Newton method proposed in this paper is the only one which correctly recovers the true network topology. In our experiments, we enforced the ordinal constraints:

$$D_{13} \le D_{12},\ D_{13} \le D_{14},\ D_{15} \le D_{12},\ D_{15} \le D_{14}.$$

We note that those ordinal constraints are obeyed by the true network, but are violated by the erroneous D. It is those explicitly enforced ordinal constraints that distinguish our method from the rest.

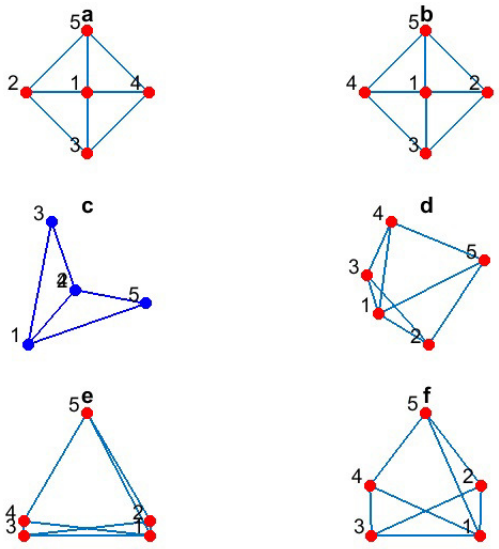

Fig. 1.1. A square network with large erroneous distances. Figure (a) is the true network; (b) is reconstructed by the smoothing Newton method in this paper; (c) is by the nearest EDM method in [21]; (d) is by cMDS (matlab built-in function `cmdscale`); (e) is by nonmetric MDS (matlab built-in function `mdscale`); (f) is by the matlab built-in function `cmdscale` with the criterion of `metricsstress`.

In general, MDS methods that deal with ordinal constraints (in various forms) are referred as nonmetric MDS. The default option of `mdscale` in Fig. 1.1 is such a method known to be the Shepard-Kruskal method [17, 18, 26, 27]. The purpose of this paper is to take advantage of the recent progress on matrix optimization [12–14, 21, 22] to investigate efficient numerical methods belonging to the category of nonmetric MDS. So why shall we consider a matrix optimization model for this widely studied problem? A short answer is that the matrix model proposed in this paper has already outperformed several leading models in Fig. 1.1, though on a small network problem.

A deep reason is that the matrix model has an obvious convex relaxation, whereas the traditional vector models do not enjoy such a property. For example, the famous squared stress criterion used in Fig. 1.1 (f) has the objective:

$$\min_{\mathbf{x}_1,\cdots,\mathbf{x}_n\in\mathbb{R}^r} \sum_{i\neq j}\left(\delta_{ij}^2 - \|\mathbf{x}_i-\mathbf{x}_j\|^2\right)^2,$$

where δ_{ij} are known inaccurate distances in Δ matrix. Obviously, the objective is not convex and it is convex if and only if the embedding dimension is as high as $(n-1)$ (usually $r \ll n$). That is, its convex relaxation would render very high-dimensional embedding. It is of course to be avoided in practice. Other vector models well described in [4] also suffer a similar drawback. We omit details to save space.

The organization of the paper is as follows. In Section 2, we will derive the Euclidean distance matrix optimization model with ordinal constraints. In Section 3, we describe the inexact smoothing Newton method to solve our model and the general H-weighted model. In Section

4, we illustrate some key implementation issues (to reduce the computational complexity) and report some numerical results, which demonstrate the efficiency of our algorithm. We conclude in Section 5.

Notations. Let $\mathcal{S}^n$ be the set of symmetric matrices of $n \times n$, $\mathcal{S}^n_+$ be the set of positive semidefinite matrices. We use $X \preceq 0$ to mean that $X \in \mathcal{S}^n$ is negative semidefinite. $\Pi_\Omega(\cdot)$ denotes the projection onto set Ω. $\mathbf{p}_i$ denotes the i-th column of matrix P, and $\mathbf{e}_i$ is the i-th column of identity matrix I. '$\circ$' denotes the Hardmard product. $\mathbf{y}_{i:j}$ denotes the components of vector $\mathbf{y}$ from i to j, and $\{\mathbf{y}\}^\perp$ denotes the orthogonal space of $\mathbf{y}$. $\mathrm{diag}(X)$ is the vector formed by the diagonal entries of $X \in \mathcal{S}^n$, and $\mathrm{Diag}(\mathbf{y})$ is the diagonal matrix with $\mathbf{y}$ as the diagonal entries. $\|\cdot\|$ denotes the Frobenius norm for matrices.

2. An EDM Optimization Model with Ordinal Constraints

Suppose we are given an inaccurate distance matrix, denoted by Δ, and an index set $\mathcal{C}$ consisting of quadruple indices (i, j, k, l) such that

$$D_{ij} \le D_{kl}, \qquad (i, j, k, l) \in \mathcal{C} \qquad \text{(ordinal constraints)}. \tag{2.1}$$

Our purpose is to compute a true EDM D such that it is as close to Δ as possible, but under the constraints (2.1). This gives rise to the following problem:

$$\min_{D \in \mathcal{S}^n} \frac{1}{2}\|D - \Delta\|^2 \qquad \text{s.t. } D \text{ is EDM} \quad \text{and} \quad (2.1).$$

Our matrix optimization problem is a full characterization of the above.

It is well-known [25, 31] that D is an EDM if and only if

$$\mathrm{diag}(D) = 0 \qquad \text{and} \qquad D \in \mathcal{K}^n_- := \{Y \in \mathcal{S}^n \mid JYJ \preceq 0\},$$

where $\mathcal{K}^n_-$ is known as the almost negative semidefinite cone. Moreover, a set of points $\mathbf{x}_i \in \mathbb{R}^r$ $i = 1, \ldots, n$ can be obtained through cMDS (1.1) and (1.2). The embedding dimension r is given by

$$r = \mathrm{rank}(JDJ).$$

Therefore, our full matrix optimization model takes the following form:

$$\begin{array}{ll} \min_{D \in \mathcal{S}^n} & \frac{1}{2}\|D - \Delta\|^2 \\ \text{s.t.} & \mathrm{diag}(D) = 0, \quad D_{ij} \le D_{kl} \qquad (i, j, k, l) \in \mathcal{C}, \\ & D \in \mathcal{K}^n_-, \\ & \mathrm{rank}(JDJ) \le r. \end{array} \tag{2.2}$$

We highlight two computational difficulties in this model. One is that the rank constraint makes the problem nonconvex. Fortunately, dropping the constraint straightforwardly yields the convex relaxation:

$$\begin{array}{ll} \min_{D \in \mathcal{S}^n} & \frac{1}{2}\|D - \Delta\|^2 \\ \text{s.t.} & \mathrm{diag}(D) = 0, \quad D_{ij} \le D_{kl} \qquad (i, j, k, l) \in \mathcal{C}, \\ & D \in \mathcal{K}^n_-. \end{array} \tag{2.3}$$

It is this model that we are going to solve. The other difficulty is that there might be too many ordinal constraints in (2.1). The order can be as high as $O(n^4)$. It still remains a challenging

question as to how to choose $\mathcal{C}$ given Δ. In a simpler case where Δ contains the true ordinal information, the number of ordinal constraints can be reduced to $O(n^2)$. Another issue that is worth to emphasize is that the problem (2.2) is always feasible. A trivial solution is the zero solution, i.e., D is the zero matrix. That is, all embedding points crash to just one point. This could happen if too many ordinal constraints (even constricting constraints) are enforced in the model (2.3). This degenerate solution also explains the often observed "crowding phenomenon" when there are too many ordinal constraints. The remaining part is to solve (2.3).

3. Inexact Smoothing Newton Method

In this section, we will adapt the inexact smoothing Newton method of Gao and Sun [13] to solve (2.3). Smoothing Newton methods have been the major methods to solve nonlinear complementarity problem (NCP) (see the celebrated paper by Qi et al. [24]). It was extended to solving a correlation matrix optimization problem by Gao and Sun [13]. Below we first reformulate (2.3) to nonlinear equations where a smoothing Newton method applies, and we then propose a variant version of the method in order to deal with a large number of ordinal constraints. Finally, we discuss the general weighted model for missing data case.

3.1. Smoothing nonlinear equations

In this part, we mainly explain how the problem can be solved by a smoothing method. We first derive a nonsmooth-equation reformulation of our problem, then choose a smoothing function to get a smooth approximation of the nonlinear equation. Finally, we describe the smoothing algorithm.

(a) Nonsmooth equation reformulation. To ease our description, we define the linear operator $\mathcal{A} : \mathcal{S}^n \longrightarrow \mathbb{R}^{p+q}$ by

$$\mathcal{A}(Y) := \begin{bmatrix} \langle A_i, Y\rangle \\ \vdots \\ \langle A_{p+q}, Y\rangle \end{bmatrix} = \begin{bmatrix} \langle \mathcal{A}_1, Y\rangle \\ \langle \mathcal{A}_2, Y\rangle \end{bmatrix} = \begin{bmatrix} \mathrm{diag}(Y) \\ (Y_{ij} - Y_{kl})_{(i,j,k,l)\in\mathcal{C}} \end{bmatrix}, \ \mathbf{b} = 0 \in \mathbb{R}^{p+q},$$

where $p = n$ and q are the numbers of equality and inequality constraints, respectively. The adjoint operator $\mathcal{A}^* : \mathbb{R}^{p+q} \longrightarrow \mathcal{S}^n$ is defined by $\mathcal{A}^*\mathbf{y} = \sum_{i=1}^{p+q} y_i A_i$. Then (2.3) can be equivalently written as

$$\begin{array}{ll} \min_{D\in\mathcal{S}^n} & \frac{1}{2}\|D - \Delta\|^2 \\ \text{s.t.} & \mathcal{A}(D) - \mathbf{b} \in \mathcal{Q} := \{0\}^p \times \mathbb{R}^q_+, \\ & D \in \mathcal{K}^n_-. \end{array} \tag{3.1}$$

The (Lagrangian) dual problem of (3.1) (we omit the details of the derivation) is

$$\begin{array}{ll} \min_{\mathbf{y}\in\mathbb{R}^{p+q}} & \theta(\mathbf{y}) = \|\Pi_{\mathcal{K}^n_-}(\Delta + \mathcal{A}^*\mathbf{y})\|^2 - \mathbf{y}^T\mathbf{b} \\ \text{s.t.} & \mathbf{y} \in \mathcal{Q}^* := \mathbb{R}^p \times \mathbb{R}^q_+, \end{array} \tag{3.2}$$

where $\Pi_{\mathcal{K}^n_-}(Y)$ is the orthogonal projection of $Y \in \mathcal{S}^n$ onto the cone $\mathcal{K}^n_-$. It follows from [11, Eq. 29]

$$\Pi_{\mathcal{K}^n_-}(Y) = Y - \Pi_{\mathcal{S}^n_+}(JYJ), \qquad \forall\ Y \in \mathcal{S}^n. \tag{3.3}$$

If $\mathbf{y}^*$ is the optimal solution to (3.2), the optimal solution to (3.1) is given by $D^* = \Pi_{\mathcal{K}_-^n}(\Delta + \mathcal{A}^*\mathbf{y}^*)$. Note that $\theta(\cdot)$ is convex and continuously differentiable. Moreover, its gradient is given by

$$\nabla\theta(\mathbf{y}) = \mathcal{A}(\Pi_{\mathcal{K}_-^n}(\Delta + \mathcal{A}^*\mathbf{y})) - \mathbf{b} = \mathcal{A}(\Delta + \mathcal{A}^*\mathbf{y} - \Pi_{\mathcal{S}_+^n}(J(\Delta + \mathcal{A}^*\mathbf{y})J)) - \mathbf{b}. \tag{3.4}$$

Since the dual problem (3.2) is convex, it is equivalent to solving its KKT condition:

$$\begin{aligned} &(\nabla\theta(\mathbf{y}))_i \geq 0, \quad i = p+1, \ldots, p+q, \\ &\mathbf{y} \in \mathcal{Q}^*, \quad \langle \mathbf{y},\ \nabla\theta(\mathbf{y})\rangle = 0. \end{aligned}$$

By using the plus operator $t_+ := \max(0, t)$, the KKT condition is equivalent to (see [10])

$$F(\mathbf{y}) := \mathbf{y} - \Pi_{\mathcal{Q}^*}(\mathbf{y} - \nabla\theta(\mathbf{y})) = 0, \tag{3.5}$$

where $\Pi_{\mathcal{Q}^*}(\mathbf{x}) : \mathbb{R}^{p+q} \mapsto \mathbb{R}^{p+q}$ is defined as

$$\Big(\Pi_{\mathcal{Q}^*}(\mathbf{x})\Big)_i = \begin{cases} x_i, & \text{if } i = 1, \cdots, p, \\ (x_i)_+, & \text{if } i = p+1, \cdots, p+q. \end{cases} \tag{3.6}$$

(b) Smoothing equation approximation. In (3.5), there are two parts that give rise to non-differentiability, namely $\Pi_{\mathcal{Q}^*}(\cdot)$ and $\Pi_{\mathcal{S}_+^n}(\cdot)$. Fortunately, those two types of nonsmooth functions can be well approximated by many smoothing functions. We use the Huber smoothing function as in [13] for $(t)_+$:

$$\phi(\epsilon, t) = \begin{cases} t, & \text{if } t \geq \frac{|\epsilon|}{2}, \\ \frac{1}{2|\epsilon|}(t + \frac{|\epsilon|}{2})^2, & \text{if } -\frac{|\epsilon|}{2} < t < \frac{|\epsilon|}{2}, \\ 0, & \text{otherwise}, \end{cases} \tag{3.7}$$

where $\epsilon > 0$ is the smoothing parameter. Hence, the smoothing function for $\Pi_{\mathcal{Q}^*}(\mathbf{x})$ is given by $\psi(\epsilon, \mathbf{x}) \in \mathbb{R}^{p+q}$, whose elementwise component is given by

$$\psi_i(\epsilon, \mathbf{x}) = \begin{cases} x_i, & \text{if } i = 1, \cdots, p, \\ \phi(\epsilon, x_i) & \text{if } i = p+1, \cdots, p+q. \end{cases} \tag{3.8}$$

We now describe how to approximate $\Pi_{\mathcal{S}_+^n}(X)$. For $X \in \mathcal{S}^n$ with spectral decomposition

$$X = P\mathrm{Diag}(\lambda_1, \cdots, \lambda_n)P^T, \tag{3.9}$$

where $\lambda_1 \geq \cdots \geq \lambda_n$ are its eigenvalues and P consists of the corresponding orthonormal eigenvectors of X, we know that $\Pi_{\mathcal{S}_+^n}(X) = P\mathrm{Diag}((\lambda_1)_+, \cdots, (\lambda_n)_+)P^T$. The smoothing function for $\Pi_{\mathcal{S}_+^n}(X)$ is

$$\Phi(\epsilon, X) := P\mathrm{Diag}(\phi(\epsilon, \lambda_1), \cdots, \phi(\epsilon, \lambda_n))P^T, \tag{3.10}$$

with

$$\Phi'_\epsilon(\epsilon, X) = P\mathrm{Diag}(\phi'_\epsilon(\epsilon, \lambda_1), \cdots, \phi'_\epsilon(\epsilon, \lambda_n))P^T \tag{3.11}$$

and for $H \in \mathcal{S}^n$,

$$\Phi'_X(\epsilon, X)(H) = P(\Omega(\epsilon, \lambda) \circ (P^T H P))P^T, \tag{3.12}$$

where $\Omega(\epsilon, \lambda)$ is defined as

$$\Omega(\epsilon, \lambda)_{ij} := \begin{cases} \frac{\phi(\epsilon,\lambda_i)-\phi(\epsilon,\lambda_j)}{\lambda_i-\lambda_j} \in [0,1], & \text{if } \lambda_i \neq \lambda_j, \\ \phi'(\lambda_i) \in [0,1], & \text{otherwise.} \end{cases} \tag{3.13}$$

Denote $Z := \Delta + \mathcal{A}^*\mathbf{y}$. Let $g(\epsilon, \mathbf{y})$ be the smoothing function for $\nabla\theta(\mathbf{y})$, i.e.,

$$g(\epsilon, \mathbf{y}) := \mathcal{A}(Z - \Phi(\epsilon, JZJ)) - \mathbf{b}. \tag{3.14}$$

The smoothing function for $F(\mathbf{y})$ is

$$\Upsilon(\epsilon, \mathbf{y}) := \mathbf{y} - \psi(\epsilon, \mathbf{y} - g(\epsilon, \mathbf{y})),$$

and the smoothing equation we are going to solve is:

$$E(\epsilon, \mathbf{y}) = \begin{bmatrix} \epsilon \\ G(\epsilon, \mathbf{y}) \end{bmatrix} := \begin{bmatrix} \epsilon \\ \Upsilon(\epsilon, \mathbf{y}) + \kappa|\epsilon|\mathbf{y} \end{bmatrix} = 0, \tag{3.15}$$

where $\kappa > 0$ is the regularization parameter.

We emphasize that the main reason that we described the smoothing equation (3.15) with the smoothing function is that we are going to use them to simplify the heavy computations in the next section.

(c) Inexact smoothing Newton method (ISNM). The last part in this subsection is to apply the inexact smoothing Newton method [13] on (3.15). We omit the repeat of the algorithm, but only cite its quadratic convergence result below.

Theorem 3.1. *Let $(\bar{\epsilon}, \bar{\mathbf{y}})$ be an accumulation point of the infinite sequence $\{(\epsilon^k, \mathbf{y}^k)\}$ generated by the inexact smoothing Newton method* [13]*. Assume that the constraint nondegeneracy holds at $\overline{Y}$. Then the whole sequence $\{(\epsilon^k, \mathbf{y}^k)\}$ converges to $(\bar{\epsilon}, \bar{\mathbf{y}})$ quadratically, i.e.,*

$$\|(\epsilon^{k+1} - \bar{\epsilon}, \mathbf{y}^{k+1} - \bar{\mathbf{y}})\| = \mathcal{O}(\|(\epsilon^k - \bar{\epsilon}, \mathbf{y}^k - \bar{\mathbf{y}})\|^2).$$

3.2. A variant for a large number of ordinal constraints

As mentioned in Section 2, the number of inequality constraints could be $O(n^4)$, leading to an extremely high computational cost as n grows. To reduce this number, a reasonable way is to run ISNM iteratively, and add the violated ordinal constraints step by step. Recall $\mathcal{C}$ denotes the set of ordinal constraints. At the beginning, we start with an initial subset of $\mathcal{C}$, denoted by $\mathcal{C}^0$. An obvious choice is $\mathcal{C}^0 = \emptyset$, as used below. But we could start with a small subset from $\mathcal{C}$. We refer to the following variant of ISNM as vISNM.

Algorithm 3.1. *vISNM*

S0 $j := 0$, *let* $\mathcal{C}^j = \emptyset$.

S1 Replace $\mathcal{C}$ *in* (3.1) *with ordinal constraint set* $\mathcal{C}^j$ *and solve it by ISNM to get* D^{j+1}.

S2 Check whether D^{j+1} *satisfies the ordinal constraints in* $\mathcal{C}$. *Denote the set of violated constraints as* $\mathcal{C}^v(D^{j+1})$. *If* $\mathcal{C}^v(D^{j+1}) = \emptyset$, *stop, otherwise, let* $\mathcal{C}^{j+1} = \mathcal{C}^j \cup \mathcal{C}^v(D^{j+1})$, $j := j + 1$, *go to S1.*

As we will show in the numerical part, vISNM can be a good alternative of ISNM, particularly for a large number of ordinal constraints. One particular reason for the effectiveness of vISNM is that we used a warm-start strategy where we used the iterate obtained from the previous step as the starting point at the current step. The convergence of vISNM is given as follows.

Theorem 3.2. *Let $\{D^j\}$ be generated by vISNM, i.e., D^j is the optimal solution of $(\mathcal{P}_j)$*

$$(\mathcal{P}_j)\quad \begin{array}{ll} \min_{D\in\mathcal{S}^n} & \frac{1}{2}\|D-\Delta\|^2 := f(D) \\ \text{s.t.} & \text{diag}(D)=0,\quad D\in\mathcal{K}^n_-, \\ & D_{is}\le D_{kl} \qquad (i,s,k,l)\in\mathcal{C}^j. \end{array} \tag{3.16}$$

Suppose vISNM stops at $j=m$, i.e., $\mathcal{C}^v(D^m)=\emptyset$. Then D^m is an optimal solution of (2.3).

Proof. For contradiction, suppose D^m is not an optimal solution of (2.3). Let $D^*\neq D^m$ be an optimal solution of (2.3). Note that $\mathcal{C}^v(D^m)=\emptyset$ implies that D^m is a feasible point of (2.3). Therefore, $f(D^*)<f(D^m)$. On the other hand, $\mathcal{C}^1\subseteq\cdots\subseteq\mathcal{C}^m\subseteq\mathcal{C}$ implies that D^* is a feasible point of $(\mathcal{P}_m)$. There is $f(D^*)\ge f(D^m)$, which contradicts with $f(D^*)<f(D^m)$. Consequently, D^m is an optimal solution of (2.3). The proof is finished. □

3.3. The general weighted case

The proposed method can be extended to a more general case where some of the elements in Δ could be missing. For example, define the symmetric matrix $H\in\mathcal{S}^n$ by

$$H_{ij}:=\begin{cases} 0 & \text{if } \delta_{ij} \text{ is missing,} \\ 1 & \text{otherwise.} \end{cases}$$

Then our H-weighted problem takes the following form:

$$\begin{array}{ll} \min_{D\in\mathcal{S}^n} & \frac{1}{2}\|H\circ(D-\Delta)\|^2 := f(D) \\ \text{s.t.} & \mathcal{A}(D)-\mathbf{b}\in\mathcal{Q},\ D\in\mathcal{K}^n_-. \end{array} \tag{3.17}$$

The majorization approach proposed in [14,21] can be borrowed to solve (3.17). Given D^j, the majorization approach is to minimize the majorization function $f_j(D)$ at each iteration to get D^{j+1}:

$$f_j(D):=f(D^j)+\langle H\circ H\circ(D^j-\Delta),D-D^j\rangle+\frac{1}{2}\|W^{\frac{1}{2}}(D-D^j)W^{\frac{1}{2}}\|^2 \tag{3.18}$$

where $W:=\text{Diag}(\mathbf{w})$, $w_i=\max\{\tau,\max\{H_{is},s=1,\cdots,n\}\}$, $i=1,\cdots,n$, $\tau>0$ is a constant. That is to solve the following diagonally weighted problem in each outer iteration:

$$\begin{array}{ll} \min_{D\in\mathcal{S}^n} & \frac{1}{2}\|W^{\frac{1}{2}}(D-\Delta^j)W^{\frac{1}{2}}\|^2 \\ \text{s.t.} & \mathcal{A}(D)-\mathbf{b}\in\mathcal{Q},\ D\in\mathcal{K}^n_-, \end{array} \tag{3.19}$$

where $\Delta^j:=D^j-W^{-1}(H\circ H\circ(D^j-\Delta))W^{-1}$. The iterates stop if $D^{j+1}=D^j$. Denote the resulting algorithm as majorized inexact smoothing Newton method (referred as mISNM) for (3.17).

Problem (3.19) can be solved in a similar fashion as (3.1). Indeed, let $\widetilde{D} = W^{\frac{1}{2}} D W^{\frac{1}{2}}$, $\widetilde{\Delta} = W^{\frac{1}{2}} \Delta^j W^{\frac{1}{2}}$, $\mathcal{K}^n_{\mathbf{w}} = \{X \in \mathcal{S}^n,\ X \preceq 0 \text{ on } \{W^{\frac{1}{2}}\mathbf{1}\}^{\perp}\}$. (3.19) is equivalent to the following problem

$$\begin{array}{ll} \min_{\widetilde{D}\in\mathcal{S}^n} & \frac{1}{2}\|\widetilde{D} - \widetilde{\Delta}\|^2 \\ \text{s.t.} & \widetilde{\mathcal{A}}(\widetilde{D}) - \mathbf{b} \in \mathcal{Q},\ \widetilde{D} \in \mathcal{K}^n_{\mathbf{w}}, \end{array} \tag{3.20}$$

where

$$\widetilde{\mathcal{A}}(\widetilde{D}) := \left[\begin{array}{c} \mathrm{diag}(\widetilde{D}) \\ (w_i^{-\frac{1}{2}} w_t^{-\frac{1}{2}} \widetilde{D}_{it} - w_l^{-\frac{1}{2}} w_s^{-\frac{1}{2}} \widetilde{D}_{ls})_{(i,t,l,s)\in\mathcal{C}} \end{array} \right].$$

The dual problem of (3.20) is

$$\begin{array}{ll} \min_{\mathbf{y}\in\mathbb{R}^{p+q}} & \theta_{\mathbf{w}}(\mathbf{y}) := \|\Pi_{\mathcal{K}^n_{\mathbf{w}}}(\widetilde{\Delta} + \widetilde{\mathcal{A}}^*\mathbf{y})\|^2 - \mathbf{y}^T\mathbf{b} \\ \text{s.t.} & \mathbf{y} \in \mathcal{Q}^*, \end{array} \tag{3.21}$$

where $\Pi_{\mathcal{K}^n_{\mathbf{w}}}(Y) := Y - \Pi_{S^n_+}(\widetilde{J}Y\widetilde{J})$, $\widetilde{J} := I - \frac{1}{\mathbf{w}^T_e}\mathbf{w}^{\frac{1}{2}}(\mathbf{w}^{\frac{1}{2}})^T$. By modifying the input matrix $\widetilde{\Delta}$, $\widetilde{\mathcal{A}}(Y)$ and $\widetilde{J}Y\widetilde{J}$, ISNM can be extended to the diagonally weighted case (3.20).

4. Implementations and Numerical Test

In order for the smoothing Newton method to work, one has to resolve the heavy computation being involved in the multiplications between matrices, encountered in the Newton equation in the method. In this part, we first illustrate a few technical tricks that significantly reduce the computational cost. We note that some of the matrices are very sparse. The second part includes some of numerical results. One test problem is from real data; while the others are standard test problems used in wireless sensor network localization.

4.1. Key implementation issues

In ISNM, BiCGStab with diagonal preconditioner is used to solve the following linear system

$$G(\epsilon^k, \mathbf{y}^k) + G'_{\mathbf{y}}(\epsilon^k, \mathbf{y}^k)\Delta\mathbf{y}^k = 0 \tag{4.1}$$

in an inexact way, where $\Delta\mathbf{y}^k$ is the Newton step. One may worry about the complicated calculation of the Jacobian matrix as well as the potential high computational cost due to the presence of ordinal constraints. However, it turns out that ordinal constraints enjoy some nice properties which can be explored to further simplify the calculations. Below, we pick up a couple of key observations to show how the computational cost can be reduced.

The Jacobian $G'_{\mathbf{y}}(\epsilon^k, \mathbf{y}^k)$ is characterized in the following implicit way. Let $X := JZJ$, and $\mathbf{z} := \mathbf{y} - g(\epsilon, \mathbf{y})$. For $\mathbf{h} \in \mathbb{R}^{p+q}$ (we drop the superscript k for simplicity),

$$\begin{array}{rcll} G'_y(\epsilon,\mathbf{y})\mathbf{h} & = & \Upsilon'_{\mathbf{y}}(\epsilon,\mathbf{y})\mathbf{h} + \kappa|\epsilon|\mathbf{h} & \\ & = & \mathbf{h} - \psi'_{\mathbf{z}}(\epsilon,\mathbf{z}) \circ (\mathbf{h} - \mathcal{A}(\mathcal{A}^*\mathbf{h} - \Phi'_X(\epsilon, X)(J\mathcal{A}^*\mathbf{h}J))) + \kappa|\epsilon|\mathbf{h} & \\ & = & \left\{ \begin{array}{ll} \langle A_i, \mathcal{A}^*\mathbf{h} - \Phi'_X(\epsilon, X)(J\mathcal{A}^*\mathbf{h}J)\rangle + \kappa|\epsilon|h_i, & i \le p, \\ h_i - \psi'_{z_i}(\epsilon, z_i)\left(h_i - \langle A_i, \mathcal{A}^*\mathbf{h} - \Phi'_X(\epsilon, X)(J\mathcal{A}^*\mathbf{h}J)\rangle\right) + \kappa|\epsilon|h_i, & i \ge p+1. \end{array} \right. \end{array}$$

Furthermore, with (3.12), we have

$$\begin{array}{rcl} \langle A_i,\ \Phi'_X(\epsilon, X)(J\mathcal{A}^*\mathbf{h}J)\rangle & = & \langle A_i, P(\Omega(\epsilon,\lambda) \circ (P^T J\mathcal{A}^*\mathbf{h}JP))P^T\rangle \\ & = & \langle P^T A_i P,\ \Omega(\epsilon,\lambda) \circ (P^T J\mathcal{A}^*\mathbf{h}JP)\rangle \end{array}$$

and

$$\begin{aligned} P^T J\mathcal{A}^*\mathbf{h}JP &= P^T\mathcal{A}^*\mathbf{h}P - \frac{1}{n}P^T\mathbf{1}\mathbf{1}^T\mathcal{A}^*\mathbf{h}P - \frac{1}{n}P^T A^*\mathbf{h}\mathbf{1}\mathbf{1}^T P + \frac{1}{n^2}(\mathbf{1}^T\mathcal{A}^*\mathbf{h}\mathbf{1})P^T\mathbf{1}\mathbf{1}^T P \\ &:= P^T\mathcal{A}^*\mathbf{h}P - M - M^T + N. \end{aligned} \tag{4.2}$$

We focus on how to simplify M and N. This is heavily based on the following key observations.

(i) Let $i \geq p+1$. Suppose $A_i = \frac{1}{2}(\mathbf{e}_{r_i}\mathbf{e}_{j_i}^T + \mathbf{e}_{j_i}\mathbf{e}_{r_i}^T - \mathbf{e}_{s_i}\mathbf{e}_{t_i}^T - \mathbf{e}_{t_i}\mathbf{e}_{s_i}^T)$. There is

$$\mathbf{1}^T A_i h_i = \frac{1}{2}h_i(\mathbf{e}_{r_i} + \mathbf{e}_{j_i} - \mathbf{e}_{s_i} - \mathbf{e}_{t_i}), \tag{4.3}$$

$$\mathbf{1}^T A_i h_i \mathbf{1} = 0, \tag{4.4}$$

which gives

$$\mathbf{1}^T\mathcal{A}_2^*\mathbf{h}_{p+1:p+q}\mathbf{1} = \sum_{i=p+1}^{p+q}\mathbf{1}A_i h_i\mathbf{1}^T = 0.$$

(ii) Recall $P\mathrm{Diag}(\lambda_1,\cdots,\lambda_n)P^T = J(\Delta + \mathcal{A}^*\mathbf{y})J$. Note that $J\mathbf{1} = 0$, $J(\Delta + \mathcal{A}^*\mathbf{y})J$ has an eigenvalue 0 with eigenvector $\frac{1}{\sqrt{n}}\mathbf{1}$. Suppose 0 is the l-th eigenvalue and the corresponding eigenvector is $\mathbf{p}_l$. We have $\mathbf{p}_l = \frac{1}{\sqrt{n}}\mathbf{1}$ or $\mathbf{p}_l = -\frac{1}{\sqrt{n}}\mathbf{1}$. For simplicity, we denote $\mathbf{p}_l = \mathrm{sign}(\mathbf{p}_l)\mathbf{1}$. Consequently,

$$P^T\mathbf{1} = \mathrm{sign}(\mathbf{p}_l)\sqrt{n}\mathbf{e}_l, P^T\mathbf{1}(\mathbf{1}^T\mathcal{A}^*\mathbf{h}P) = \mathrm{sign}(\mathbf{p}_l)\sqrt{n}\mathbf{e}_l(\mathbf{1}^T\mathcal{A}^*\mathbf{h}P) = \begin{bmatrix} 0 \\ \mathbf{1}^T\mathcal{A}^*\mathbf{h}P \\ 0 \end{bmatrix}.$$

(iii) Based on (4.3), there is

$$\mathbf{1}^T\mathcal{A}_2\mathbf{h}_{p+1:p+q}P = \sum_{i=p+1}^{p+q} h_i(\mathbf{e}_{r_i} + \mathbf{e}_{j_i} - \mathbf{e}_{s_i} - \mathbf{e}_{t_i})P = \sum_{i=p+1}^{p+q} h_i(\mathbf{p}_{r_i} + \mathbf{p}_{j_i} - \mathbf{p}_{s_i} - \mathbf{p}_{t_i}).$$

Consequently,

$$\begin{aligned} M &= \frac{1}{n}P^T\mathbf{1}\mathbf{1}^T\mathcal{A}^*\mathbf{h}P = \frac{1}{\sqrt{n}}\mathrm{sign}(\mathbf{p}_l)\mathbf{e}_l(\mathbf{h}_{1:n}^T P + \sum_{i=p+1}^{p+q} h_i(\mathbf{p}_{r_i} + \mathbf{p}_{j_i} - \mathbf{p}_{s_i} - \mathbf{p}_{t_i})) \\ N &= \frac{1}{n^2}(\mathbf{1}^T\mathcal{A}^*\mathbf{h}\mathbf{1})P^T\mathbf{1}\mathbf{1}^T P = \frac{1}{n}(\mathbf{h}^T\mathbf{1})\mathbf{e}_l\mathbf{e}_l^T. \end{aligned}$$

One can observe that M is a sparse matrix with only the l-th row nonempty, and N is a sparse matrix with only (l,l) entry nonzero. Further, the computational cost for M is $O(\max(p+q,n)n)$, and for N is $O(n)$.

4.2. Numerical results

One typical application of nonmetric MDS is the sensor network localization problem, where the position of some points are known (referred as anchors), and the rest are unknown (referred as sensors). Very often, the observed dissimilarities δ_{ij}'s are monotonically in distances, such as the time of arrival (TOA) or received signal strength (RSS). Therefore, nonmetric MDS model

is more appropriate in this situation. Given dissimilarities δ_{ij}, $i, j = 1, \cdots, n$, we run ISNM with $\Delta = (\delta_{ij}^2)$ to get D, then the estimated location of points $\overline{X} \in \mathbb{R}^{r\times n}$ is obtained by cMDS (1.1) and (1.2). Procrustes process is used to get the final estimated location X. To evaluate the result, we compare the Root Mean Squared Distance (RMSD), which is given by

$$RMSD = \sqrt{\frac{\|X - X_{\mathtt{true}}\|^2}{n}}, \tag{4.5}$$

where $X_{\mathtt{true}} \in \mathbb{R}^{r\times n}$ are the true positions.

We ran all the tests in Matlab 2013b on a computer with Pentium(R)D CPU 2.30GHz, RAM 4GB. For ISNM, the linear system (4.1) is solved by BiCGStab with diagonal preconditioner. The stopping criteria is $res = \|E(\epsilon^k, y^k)\| \le tol$, with $tol = 10^{-6}$. Other parameters are set as default.

Sensor network localization example [20]. This is a test problem from real data, whose generating process is described in [20]. There are $n = 44$ points, with 4 anchors, and 40 unknown sensors. The observed dissimilarity data δ_{ij} is measured by TOA ($\delta_{ij}^{TOA} = T_{TOA} \times v\_p\_TOA$) or RSS ($\delta_{ij}^{RSS} = tilde\_d\_RSS$) which can be download via http://web.eecs.umich.edu/ $\sim$ hero/localize/. The data contains large errors and hence this test problem has been challenging for many metric- and nonmetric-MDS embedding methods.

For this example, the procrustes process is first done on the 4 anchors to get the transformation information, and then map the rest 40 sensors to get the final estimated positions according to the transformation information. This is contrast to many reported results, where the procrustes process was done on all 44 sensors. This is not reasonable as we only know the true positions of some sensors (called anchors). Doing a procrustes process on all the sensors (using their true positions) usually leads to better RMSD results.

As for order information, there are different ways to obtain a list of ordinal constraints. One way is to use the information given by δ_{ij}. Note that δ_{ij} may not stand for the true ordinal information. For example, in Fig. 4.1, we plot the elements of true distances d_{ij}'s in an ascending order, and then plot the corresponding δ_{ij}^{TOA} in terms of d_{ij}'s order. It turns out that the order given by δ_{ij}^{TOA} is quite different from that given by d_{ij}.

Therefore, we use different ways of choosing ordinal constraints, and the results are reported in Table 4.1, where ISNM0 did not use any ordinal information, ISNM uses the ordinal information given by δ_{ij}, ISNM1, ISNM2 and ISNM3 use 40, 70, and 100 percent of the true ordinal constraints, respectively. From Table 4.1, one can easily get the following observations: (i) Compared with ISNM0, which did not use ordinal constraints, ISNM and ISNM1-ISNM3 give better localization results, showing that ordinal constraints do help in this situation; (ii) The order information is also important in the sense that correct ordinal constraints help a lot, since ISNM1-ISNM3 provide better embedding results than ISNM, by using some priori information of true ordinal constraints. Typical embedding results are shown in Fig. 4.2-4.4.

We also compare our results with `mdscal`, SMACOF and SDP-SNL [3]. SDP-SNL is a popular method based on semidefinite programming for sensor network localization [3]. For SMACOF, we run '`mds`' Matlab package developed by Bronstein et al [5, 6] by choosing the options '`smacof`'. Since random point is used as the initial point in SMACOF, we run 50 times and report the averaged RMSD. Comparing with other methods, ISNM is as good as mdscale, and performs better than SDN-SNL and SMACOF. ISNM1-ISNM3 outperform the other methods, which verifies again the importance of the priori ordinal information.

Next we compare ISNM with vISNM. Fig. 4.5 shows the RMSDs and the number of ordinal constraints used (denoted as $|\mathcal{C}^j|$) in the outer iterations in vISNM. Compared with ISNM,

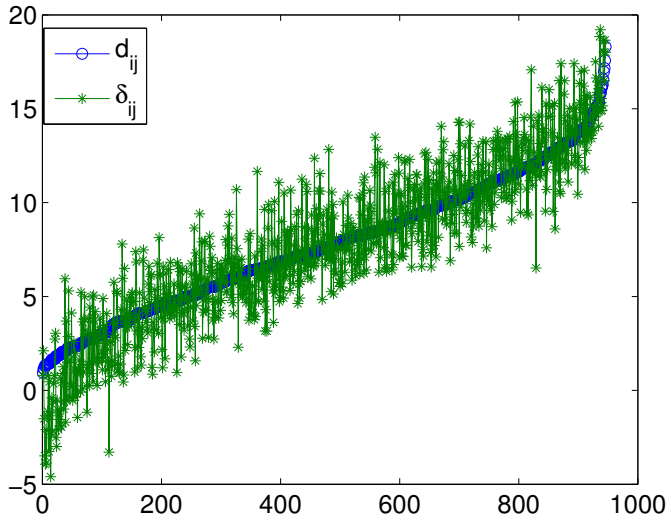

Fig. 4.1. Comparison of the order by d_{ij} and δ_{ij}^{TOA}.

Table 4.1: RMSDs by different methods.

	SMACOF	SDP-SNL	mdscale	ISNM0	ISNM	ISMN1	ISNM2	ISMN3
TOA	1.18	1.18	1.19	1.44	1.18	0.57	0.15	0.05
RSS	2.91	2.56	2.05	2.47	2.08	1.24	0.43	0.07

vISNM reaches comparable RMSD and almost the same embedding result, see Fig. 4.6 (a). The number of ordinal constraints in the last iteration $j = 6$ is about 800, which is smaller than the total number 945. After $j = 6$, all ordinal constraints are satisfied. Therefore, by selectively dealing with the violated ordinal constraints, we can reach almost the same output as the original vISNM. This provides us a way to choose effective ordinal constraints instead of tackling all ordinal constraints together. Furthermore, by taking care of the violated ordinal constraints iteratively, the computational cost can be saved, leading to potentially an effective way to deal with a large number of ordinal constraints. Therefore, we can run vISNM instead of ISNM to reach the same output but with smaller computational cost. Fig. 4.5 also implies that the most significant improvement happens at $j = 1, 2, 3$, meaning that we can run vISNM for three outer iterations, to save more cputime and computational cost and also to reach a comparable embedding, especially for large number of ordinal constraints. Similar performance was observed for the RSS input data, which was omitted here to save space.

Random examples. We randomly generate some standard test examples used in sensor network localization literature. n points in the square $[-10a, 10a] \times [-10a, 10a]$ are randomly generated with $a = n/100$. The true ordinal constraint set $\mathcal{C}$ can be obtained according to the position of points. $\Delta \in \mathcal{S}^n$ is generated by: $\Delta = D + 20a * rand(n, n); \Delta = 0.5(\Delta + \Delta')$. The results are reported in Table 4.2, where ISNM0 does not use any ordinal information, ISNM1 and ISNM2 use 30 and 100 percent of priori true ordinal information, respectively. vISNM1 and vISNM2 are the variant version of ISNM1 and ISNM2. We run $j = 3$ outer iterations in vISNM.

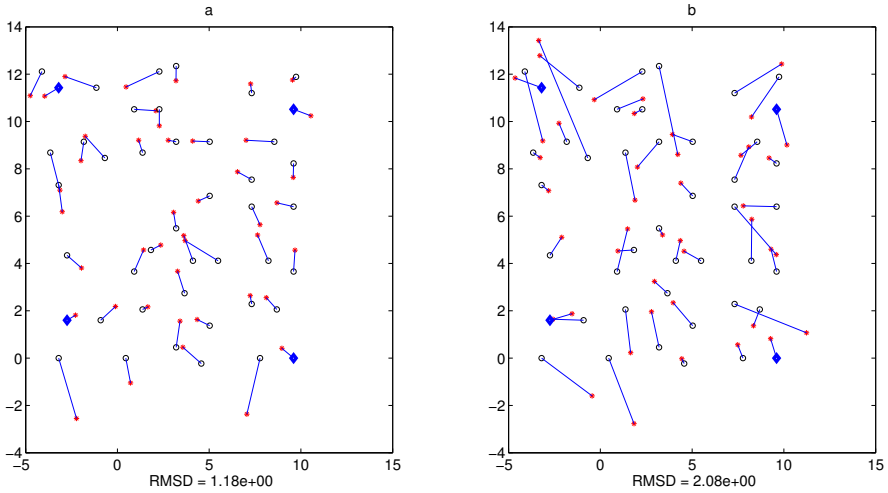

Fig. 4.2. Embedding results by ISNM for (a) TOA and (b) RSS.

RMSD, cputime t (hh:mm:ss), $|\mathcal{C}|$ for ISNM and the number of ordinal constraints $|\mathcal{C}^j|$ involved in $j = 3$ for vISNM are reported in Table 4.2. It can be observed that comparing with ISNM0, which did not use ordinal information, the rest four give better results with lower RMSDs. This verifies again the importance of ordinal constraints. Given the same priori ordinal information, vSNM1 performs not as good as ISNM1. However, vISNM2 gives fairly reasonable result with much less cputime. For example, for $n = 800$, ISNM2 takes more than one hour dealing with $320,000$ ordinal constraints, whereas vISNM2 only used couple of minutes to handle half of the ordinal constraints and returned fairly good RMSD. This indicates that vISNM is extremely useful to handle a large number of ordinal constraints.

Missing data case. Our last test is the H-weighted case (3.17). The data is generated in a similar way as above, where n points in $[-10a, 10a] \times [-10a, 10a]$ are randomly generated and $a = n/100$. The missing data index matrix H is generated in the following way: $H = ones(n, n)$, $u = rand(n, n)$, $H(find(u > b)) = 0$. The dissimilarity matrix $\Delta \in \mathcal{S}^n$ is generated by $\Delta = D + 20a * rand(n, n)$, $\Delta = 0.5(\Delta + \Delta^T)$, $\Delta = \Delta \circ H$. mISNM1 and mISNM2 use 30 and 60 percent of priori true ordinal information respectively. We choose $b = 0.6$ and $b = 0.8$. The results are presented in Table 4.3, which show again the efficiency of mISNM.

5. Conclusions

In this paper, we mainly focused on one embedding situation, where the ordinal constraints often play a more important role than the actual magnitude of the distances and need to be obeyed. This class of methods generally belongs to non-metric methods. However, the difficulty for many existing methods is that there are many ordinal constraints (can be as many as $O(n^4)$, where n is the number of embedding points). Such situations arise when the observed data contain significantly large errors when compared to the magnitudes of the true data. The embedding problem has recently attracted lots of attention, see [2, 19] and references therein for recent development.

Different from most of the existing publications, we propose a Euclidean distance matrix

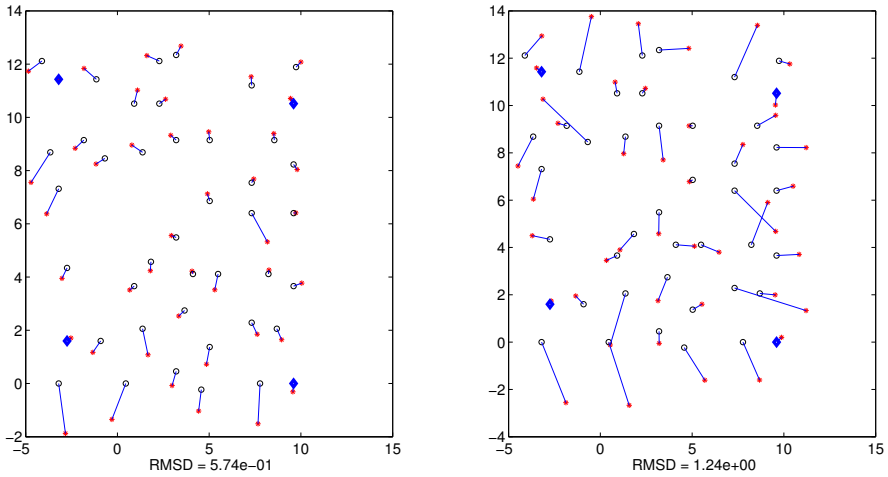

Fig. 4.3. Embedding results by ISNM1 for TOA (left) and RSS (right).

optimization model. Taking advantage of the recent advances in conic matrix optimization, an existing smoothing Newton method is used to solve the model. Further, to deal with the huge number of inequality constraints for large scale problems, a practical algorithm is proposed by adding the violation ordinal constraints step by step. Numerical results demonstrated the efficiency of the proposed model.

This is our initial try on this challenging problem. Based on our promising numerical results, we believe a couple of questions are worth further investigation. One is new strategies on selecting the violated constraints. The second research question is how to add the rank constraint to our numerical procedure. We plan to study those issues in our future work.

Acknowledgments. The authors would like to thank the editor for handling our submission and two anonymous referees for their comments. The first author's research was supported by NSF grant (No. 11671036).

References

[1] A. Y. Alfakih, A. Khandani and H. Wolkowicz, Solving Euclidean distance matrix completion problems via semidefinite programming, *Comput. Optim. Appl.*, **12** (1999), 13-30.

[2] E. Arias-Castro, Some theory for ordinal embedding, *Statistics*, in press.

[3] P. Biswas, T. C. Liang, K. C. Toh, T. C. Wang and Y. Y. Ye, Semidefinite programming approaches for sensor network localization with noisy distance measurements, *IEEE Trans. Autom. Sci. Eng.*, **3**:4 (2006), 360-371.

[4] I. Borg and P. J. F. Groenen, *Modern Multidimensional Scaling: Theory and Applications*, 2nd ed., Springer, New York, 2005.

[5] M. M. Bronstein, A. M. Bronstein and R. Kimmel, *Numerical Geometry of Non-Rigid Shapes*, Springer, New York, 2009.

[6] M. M. Bronstein, A. M. Bronstein, R. Kimmel and I. Yavneh, Multigrid multidimensional scaling, *Numer. Linear Algebra Appl.*, **13**:2-3 (2006), 149-171.

[7] T. F. Cox and M. A. A. Cox, *Multidimensional Scaling*, 2nd ed., Chapman and Hall/CRC, 2001.

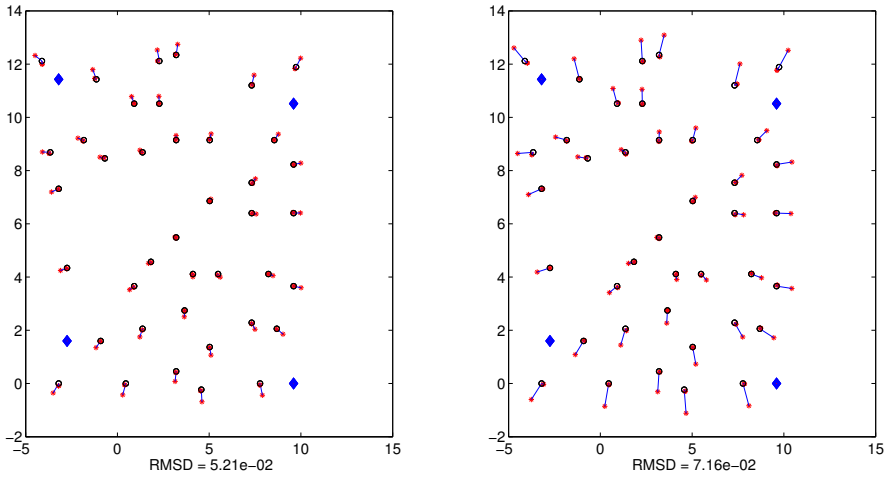

Fig. 4.4. Embedding results by ISNM3 for TOA (left) and RSS (right).

[8] J. Dattorro, *Convex Optimization and Euclidean Distance Geometry*, Mεβoo Publishing, US, 2012.
[9] C. Ding and H. D. Qi, Convex Euclidean distance embedding for collaborative position localization with NLOS mitigation, *Comput. Optim. Appl.*, **66**:1 (2017), 187-218.
[10] F. Facchinei and J. S. Pang, *Finite-Dimensional Variational Inequalities and Complementarity Problems,* Springer, New York, 2003.
[11] N. Gaffke and R. Mathar, A cyclic projection algorithm via duality, *Metrika*, **36** (1989), 29-54.
[12] Y. Gao, *Structured Low Rank Matrix Optimization Problems: A Penalized Approach*, PhD thesis, National University of Singapore, 2010.
[13] Y. Gao and D. F. Sun, Calibrating least squares covariance matrix problems with equality and inequality constraints, *SIAM J. Matrix Anal. Appl.*, **31**:3 (2009), 1432-1457.
[14] Y. Gao and D. F. Sun, A majorized penalty approach for calibrating rank constrained correlation matrix problems, Technical Report, National University of Singapore, 2010.
[15] W. Glunt, T. L. Hayden, S. Hong and J. Wells, An alternating projection algorithm for computing the nearest Euclidean distance matrix, *SIAM J. Marix Anal. Appl.*, **11**:4 (1990), 589-600.
[16] J. C. Gower, Properties of Euclidean and non-Euclidean distance matrices, *Linear Algebra Appl.*, **67** (1985), 81-97.
[17] J. B. Kruskal, Multidimensional Scaling by optimizing goodness of fit to a nonmetric hypothesis, *Psychometrika*, **29**:1 (1964), 1-27.
[18] J. B. Kruskal, Nonmetric multidimensional scaling: a numerical method, *Psychometrika*, **29**:2 (1964), 115-129.
[19] V. D. M. Nhat, D. Vo, S. Challa and S. Y. Lee, Nonmetric MDS for sensor localization, *International Symposium on Wireless Pervasive Computing*, IEEE, 2008, 396-400.
[20] N. Patwari, A. O. Hero, M. Perkins, N. S. Correal and R. J. O'Dea, Relative location estimation in wireless sensor networks, *IEEE Trans. Signal Process.*, **51**:8 (2003), 2137-2148.
[21] H. D. Qi, A semismooth Newton method for the nearest Euclidean distance matrix problem, *SIAM J. Marix Anal. Appl.*, **34**:1 (2013), 67-93.
[22] H. D. Qi and D. F. Sun, A quadratically convergent Newton method for computing the nearest correlation matrix, *SIAM J. Matrix Anal. Appl.*, **28**:2 (2006), 360-385.
[23] H. D. Qi and X. M. Yuan, Computing the nearest Euclidean distance matrix with low embedding dimensions, *Math. Program.*, **147**:1 (2014), 351-389.

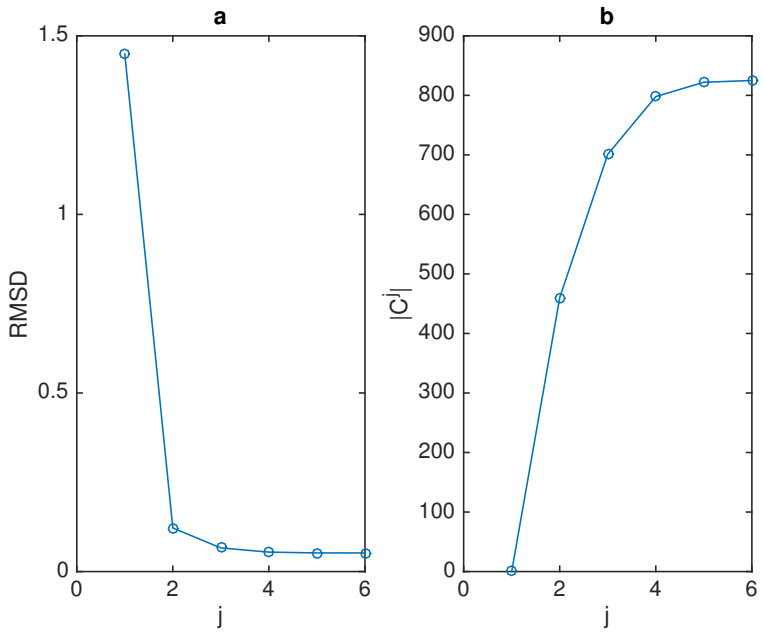

Fig. 4.5. (a) RMSD and (b) $|\mathcal{C}^j|$ by vISNM with TOA as input.

[24] L. Q. Qi, D. F. Sun and G. L. Zhou, A new look at smoothing Newton methods for nonlinear complementarity problems and box constrained variational inequalities, *Math. Program.*, **87**:1 (2000), 1-35.

[25] I. J. Schoenberg, Remarks to Maurice Frechet's Article "Sur La Definition Axiomatique D'Une Classe D'Espace Distances Vectoriellement Applicable Sur L'Espace De Hilbert, *Ann. Math.*, **36**:3 (1935), 724-732.

[26] R. N. Shepard, The analysis of proximities: Multidimensional scaling with an unknown distance function. I, *Psychometrika*, **27**:2 (1962), 125-140.

[27] R. N. Shepard, The analysis of proximities: Multidimensional scaling with an unknown distance function. II, *Psychometrika*, **27**:3 (1962), 219-246.

[28] J. B. Tenenbaum, V. D. Silva and J. C. Langford, A global geometric framework for nonlinear dimensionality reduction, *Science*, **290**:5500 (2000), 2319-2323.

[29] K. C. Toh, An inexact path-following algorithm for convex quadratic SDP, *Math. Program.*, **112**:1 (2008), 221-254.

[30] W. S. Torgerson, Multidimensional Scaling: I. Theory and Method, *Psychometrika*, **17**:4 (1952), 401-419.

[31] G. Young and A. S. Householder, Discussion of a set of points in terms of their mutual distances *Psychometrika*, **3**:1 (1938), 19-22.

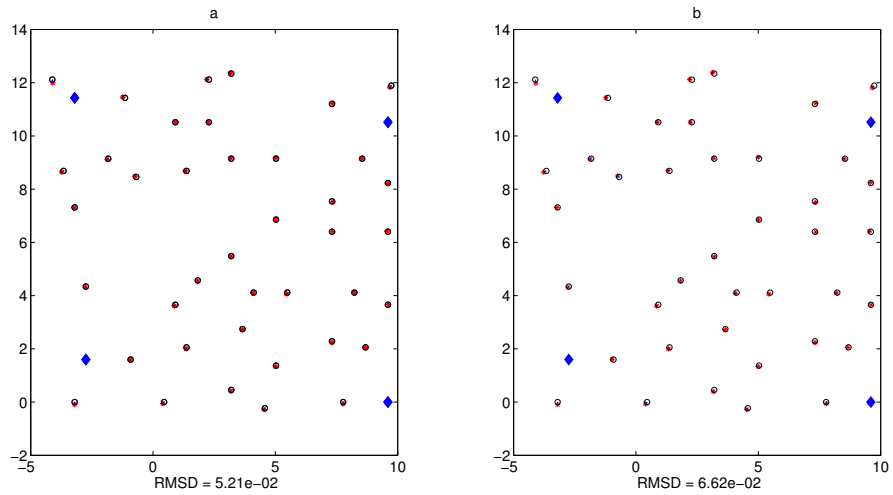

Fig. 4.6. Embedding results for vISNM with (a) j=6, (b) j = 3 and with input TOA.

Table 4.2: Results for random examples.

$n = 100$				$n = 200$		
Method	$\lvert\mathcal{C}\rvert(\lvert\mathcal{C}^j\rvert)$	RMSD	t	$\lvert\mathcal{C}\rvert(\lvert\mathcal{C}^j\rvert)$	RMSD	t
ISNM0	0	0.4427	1	0	0.3524	1
ISNM1	1979	0.3708	1	7970	0.2187	5
vSNM1	979	0.3535	2	4013	0.2360	10
ISNM2	4949	0.0299	32	19899	0.0237	4:44
vISNM2	2451	0.1168	4	9958	0.0599	18
$n = 500$				$n = 800$		
Method	$\lvert\mathcal{C}\rvert(\lvert\mathcal{C}^j\rvert)$	RMSD	t	$\lvert\mathcal{C}\rvert(\lvert\mathcal{C}^j\rvert)$	RMSD	t
ISNM0	0	0.2199	3	0	0.1797	6
ISNM1	49918	0.0965	1:15	128248	0.0660	2:45
vSNM1	24947	0.1252	2:44	64100	0.0913	5:41
ISNM2	124749	0.0119	52:25	319599	0.0054	1:19:36
vISNM2	62566	0.0257	3:55	159903	0.0168	6:15

Table 4.3: Results by mISNM.

	mISNM1		mISNM2		mISNM1		mISNM2	
	$b = 0.6$				$b = 0.8$			
n	RMSD	t	RMSD	t	RMSD	t	RMSD	t
100	0.4636	1	0.5225	1	0.6140	1.7	0.6393	1.2
200	0.6955	5.4	0.6572	5.8	0.7536	5.9	0.8244	6.2
500	0.8173	2:53	0.7936	2:02	0.8169	3:01	0.8246	2:57
800	0.8128	19:00	0.8114	18:00	0.8059	23:01	0.8154	23:13