ISSN 1342-2804

Research Reports on Mathematical and Computing Sciences

SFSDP: a **S**parse Version of **F**ull **S**emi**D**efinite **P**rogramming Relaxation for Sensor Network Localization Problems

Sunyoung Kim, Masakazu Kojima,
Hayato Waki, and Makoto Yamashita

July 2009
Revised January 2010, B–457

Department of
Mathematical and
Computing Sciences
Tokyo Institute of Technology

SERIES B: Operations Research

B-457 **SFSDP**: a **S**parse Version of **F**ull **S**emi**D**efinite **P**rogramming Relaxation for Sensor Network Localization Problems

Sunyoung Kim[⋆], Masakazu Kojima[†], Hayato Waki[‡] and Makoto Yamashita[♯]

July 2009, Revised in January 2010.

Abstract.
SFSDP is a Matlab package for solving sensor network localization (SNL) problems. These types of problems arise in monitoring and controlling applications using wireless sensor networks. SFSDP implements the semidefinite programming (SDP) relaxation proposed in Kim et al. [2009] for sensor network localization problems, as a sparse version of the full semidefinite programming relaxation (FSDP) by Biswas and Ye [2004]. To improve the efficiency of FSDP, SFSDP exploits the aggregated and correlative sparsity of a sensor network localization problem. As a result, SFSDP can handle much larger-sized problems than other softwares, and three-dimensional anchor-free problems. SFSDP analyzes the input data of a sensor network localization problem, solves the problem, and displays the computed locations of sensors. SFSDP also includes the features of generating test problems for numerical experiments.

Key words. Sensor network localization problems, semidefinite programming relaxation, sparsity exploitation, Matlab software package.

⋆ Department of Mathematics, Ewha W. University, 11-1 Dahyun-dong, Sudaemoon-gu, Seoul 120-750 Korea. The research was supported by KOSEF 2009-007-1314.
skim@ewha.ac.kr

† Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, 2-12-1 Oh-Okayama, Meguro-ku, Tokyo 152-8552 Japan. This research was partially supported by Grant-in-Aid for Scientific Research (B) 19310096.
kojima@is.titech.ac.jp

‡ Department of Computer Science, The University of Electro-Communications, 1-5-1 Chofu-gaoka, Chofu-shi, Tokyo 182-8585 Japan. This research was partially supported by Grant-in-Aid for JSPS Fellows 20003236.
waki@cs.uec.ac.jp

♯ Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, 2-12-1 Oh-Okayama, Meguro-ku, Tokyo 152-8552 Japan. M. Yamashita's research was supported by Grant-in-Aid for Young Scientists (B) 21710148.
Makoto.Yamashita@is.titech.ac.jp

1 Introduction

We introduce a Matlab package SFSDP for solving sensor network localization (SNL) problems using semidefinite programming (SDP) relaxation. SNL problems arise in monitoring and controlling applications using wireless sensor networks such as inventory management and gathering environment data. Locating sensors accurately in a wireless sensor network is an important problem for the efficiency of applications. It is also closely related to distance geometry problems like predicting molecule structures and to graph rigidity.

For a network of n sensors, an SNL problem is to locate m sensors of unknown positions ($m < n$) that match the given distances if a subset of distances and $m_a = n - m$ sensors of known positions (called anchors) are provided. Various approaches [1, 8, 9, 12, 13, 27] have been proposed for the problem to approximate the solution. Full semidefinite programming (FSDP) relaxation was introduced by Biswas and Ye in [3], and a number of solution methods based on SDP relaxation have followed [4, 5, 6, 22, 30].

The SNL problem was formulated as a quadratic optimization problem (QOP) by Biswas and Ye [3], and was solved with SDP relaxation. We call their method as the FSDP relaxation in this paper. Solving nonlinear optimization problems using SDP relaxations has been a widely popular approach for the accuracy of approximated solutions and the efficiency of the computation. Software packages based on the primal-dual interior-point methods [10, 28, 26] are used to solve SDP relaxations.

For the SNL problem with a larger number of sensors, distributed methods in [4, 7] were introduced, and a method combined with a gradient method [20] was proposed to improve the accuracy. The second-order cone programming (SOCP) relaxation was studied first in [8] and then in [27]. The solution obtained by the SOCP relaxation is inaccurate compared to that by the SDP relaxation [27]. Edge-based SDP (ESDP) and node-based SDP (NSDP) relaxations were introduced in [30] to improve the computational efficiency of Biswas-Ye's FSDP relaxation in [3]. These SDP relaxations are weaker than the FSDP relaxation in theory, however, computational results show that the quality of solution is comparable to that of FSDP. It is also shown that much larger-sized problems can be handled.

SFSDP is an implementation of the SDP relaxation proposed in Kim et al. [14], which is called the sparse FSDP relaxation as opposed to the FSDP relaxation in [3]. Both SDP relaxations are derived from the SNL problem formulated as a QOP. When solving a SDP relaxation problem by the primal-dual interior-point methods, the size of SDP relaxation problem generated from an SNL problem is one of the important factors that decides the computational efficiency. As we try to solve an SNL problem with an increasing number of sensors or in higher dimension, it is obvious that the size of SDP relaxation increases. Thus, reducing the size of SDP relaxation is essential to solve a larger-sized SNL problem. This motivates us to utilize the sparsity of problem, in particular, the aggregated and correlative sparsity [11, 18, 21] of SNL problems.

When we want to decrease the size of FSDP relaxation, the quality of obtained solution becomes an important issue. For a QOP as the SNL problem, it is shown in [29] that the solution quality of the sparse SDP relaxation is equivalent to that of the FSDP relaxation. In fact, the quality of obtained solution by SFSDP remains equivalent to that by FSDP. See Proposition 3.3 of [14]. As a result, SFSDP can handle larger-sized SNL problems, e.g., up

to 20000 sensors for 2-dimensional problems, than FSDP without deteriorating the quality of solutions.

For the SNL problem, distance data usually contain noise. SFSDP can solve the problem with both exact and noisy distances. It is designed for users who want to solve their own SNL problems and to experiment with various test problems generated by SFSDP. One feature of SFSDP is that users can select a primal-dual interior-point solver, either SDPA [10], available at [25], or SeDuMi [26], available at [24]. SDPA is known to be faster for solving large-sized problems, hence, shorter CPU time can be expected if SDPA is used. When an SNL problem is given, SFSDP analyzes input data, and calls SDPA or SeDuMi to solve the SDP relaxation problem. It also displays the figures of location of sensors at the end of computation.

Main features and capabilities of SFSDP are presented in the rest of the paper. We provide some background information on SNL problems in Section 2. In Section 3, the FSDP and sparse SDP relaxations of the problem are discussed after the description of how to extract the aggregated and correlative sparsity from the given problem. Section 4 includes implementation issues for constructing the sparse SDP relaxations. In Section 5, we explain the usage of SFSDP with illustrative examples. Numerical results are presented in Section 6. Section 7 is for concluding remarks.

2 SNL Problems

We consider a problem with m sensors and m_a $(= n - m)$ anchors to describe a form of the SNL problem that can be solved by SFSDP. Let $\rho > 0$ be a radio range, which determines the set $\mathcal{N}_x^\rho$ for pairs of sensors p and q such that their (Euclidean) distance d_{pq} is not greater than ρ, and the set $\mathcal{N}_a^\rho$ for pairs of a sensor p and an anchor r such that their distance d_{pr} does not exceed ρ;

$$\left.\begin{array}{rcl} \mathcal{N}_x^\rho & = & \{(p,q) : 1 \le p < q \le m, \|\boldsymbol{x}_p - \boldsymbol{x}_q\| \le \rho\}, \\ \mathcal{N}_a^\rho & = & \{(p,r) : 1 \le p \le m,\ m+1 \le r \le n, \|\boldsymbol{x}_p - \boldsymbol{a}_r\| \le \rho\}, \end{array}\right\}$$

where $\boldsymbol{x}_p \in \mathbb{R}^\ell$ denotes the unknown location of sensor p and $\boldsymbol{a}_r \in \mathbb{R}^\ell$ the known location of anchor r. SFSDP can solve the problem of $\ell = 2$ or 3.

Let $\mathcal{N}_x$ be a subset of $\mathcal{N}_x^\rho$ and $\mathcal{N}_a$ a subset of $\mathcal{N}_a^\rho$. By introducing zero objective function and the distance equations as constraints, we have the following form of SNL problem with exact distances.

$$\left.\begin{array}{ll} \text{minimize} & 0 \\ \text{subject to} & d_{pq}^2 = \|\boldsymbol{x}_p - \boldsymbol{x}_q\|^2 \quad (p,q) \in \mathcal{N}_x, \\ & d_{pr}^2 = \|\boldsymbol{x}_p - \boldsymbol{a}_r\|^2 \quad (p,r) \in \mathcal{N}_a. \end{array}\right\} \tag{1}$$

For the problems with noise, we consider

$$\left.\begin{array}{ll} \text{minimize} & \displaystyle\sum_{(p,q)\in\mathcal{N}_x} \left(\xi_{pq}^+ + \xi_{pq}^-\right) + \sum_{(p,r)\in\mathcal{N}_a} \left(\xi_{pr}^+ + \xi_{pr}^-\right) \\ \text{subject to} & \hat{d}_{pq}^2 = \|\boldsymbol{x}_p - \boldsymbol{x}_q\|^2 + \xi_{pq}^+ - \xi_{pq}^- \quad (p,q) \in \mathcal{N}_x, \\ & \hat{d}_{pr}^2 = \|\boldsymbol{x}_p - \boldsymbol{a}_r\|^2 + \xi_{pr}^+ - \xi_{pr}^- \quad (p,r) \in \mathcal{N}_a, \\ & \xi_{pq}^+ \ge 0,\ \xi_{pq}^- \ge 0,\ (p,q) \in \mathcal{N}_x,\ \xi_{pr}^+ \ge 0,\ \xi_{pr}^- \ge 0, (p,r) \in \mathcal{N}_a, \end{array}\right\} \tag{2}$$

where $\xi^+_{pq} + \xi^-_{pq}$ (or $\xi^+_{pr} + \xi^-_{pr}$) indicates 1-norm error in the square of estimated distance $\hat{d}_{pq}$ between sensors p and q (or estimated distance $\hat{d}_{pr}$ between sensor p and anchor r).

We use two kinds of subsets $\mathcal{N}_x$ of $\mathcal{N}^\rho_x$ and $\mathcal{N}_a$ of $\mathcal{N}^\rho_a$ instead of $\mathcal{N}^\rho_x$ and $\mathcal{N}^\rho_a$ for two reasons. First, some of the distances d_{pq} (or the estimated distances $\hat{d}_{pq}$) ($(p,q) \in \mathcal{N}^\rho_x$) and d_{pr} (or the estimated distances $\hat{d}_{pr}$) ($(p,r) \in \mathcal{N}^\rho_a$) may be unavailable in practice. Second, for numerical efficiency, we use smaller subsets $\mathcal{N}_x$ of $\mathcal{N}^\rho_x$ and $\mathcal{N}_a$ of $\mathcal{N}^\rho_a$ even when $\mathcal{N}^\rho_x$ and $\mathcal{N}^\rho_a$ are available. With smaller $\mathcal{N}_x$ and $\mathcal{N}_a$ for the SNL problems (1) and (2), the sizes of SDP relaxations become smaller, but the accuracy of locations of sensors from the SDP solutions may deteriorate. Thus, how $\mathcal{N}_x$ and $\mathcal{N}_a$ are selected is an important issue for efficiency and accuracy. This will be discussed in Section 4.1.

To transform the problems (1) and (2) into SDP relaxation [3], we first introduce an $\ell \times m$ matrix variable $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m) \in \mathbb{R}^{\ell \times m}$. Then, the system of equations (1) can be written as

$$\left.\begin{array}{ll} d^2_{pq} = \displaystyle\sum_{i=1}^{\ell} X^2_{ip} - 2\sum_{i=1}^{\ell} X_{ip}X_{iq} + \sum_{i=1}^{\ell} X^2_{iq} & (p,q) \in \mathcal{N}_x, \\ d^2_{pr} = \displaystyle\sum_{i=1}^{\ell} X^2_{ip} - 2\sum_{i=1}^{\ell} X_{ip}a_{ir} + \|\boldsymbol{a}_r\|^2 & (p,r) \in \mathcal{N}_a, \end{array}\right\} \tag{3}$$

where X_{ip} denotes the (i,p)th element of the matrix $\boldsymbol{X}$ or the ith element of $\boldsymbol{x}_p$. Now, a QOP for the SNL without noise is obtained.

$$\text{minimize } 0 \text{ subject to the equality constraints (3).} \tag{4}$$

Using the matrix variable $\boldsymbol{X}$, the problem (2) becomes

$$\left.\begin{array}{ll} \text{minimize} & \displaystyle\sum_{(p,q) \in \mathcal{N}_x} (\xi^+_{pq} + \xi^-_{pq}) + \sum_{(p,r) \in \mathcal{N}_a} (\xi^+_{pr} + \xi^-_{pr}) \\ \text{subject to} & \hat{d}^2_{pq} = \displaystyle\sum_{i=1}^{\ell} X^2_{ip} - 2\sum_{i=1}^{\ell} X_{ip}X_{iq} + \sum_{i=1}^{\ell} X^2_{iq} + \xi^+_{pq} - \xi^-_{pq} \quad (p,q) \in \mathcal{N}_x, \\ & \hat{d}^2_{pr} = \displaystyle\sum_{i=1}^{\ell} X^2_{ip} - 2\sum_{i=1}^{\ell} X_{ip}a_{ir} + \|\boldsymbol{a}_r\|^2 + \xi^+_{pr} - \xi^-_{pr} \quad (p,r) \in \mathcal{N}_a, \\ & \xi^+_{pq} \geq 0,\ \xi^-_{pq} \geq 0 \quad (p,q) \in \mathcal{N}_x, \quad \xi^+_{pr} \geq 0,\ \xi^-_{pr} \geq 0 \quad (p,r) \in \mathcal{N}_a. \end{array}\right\} \tag{5}$$

3 SDP Relaxations of the SNL Problem

We first describe the construction of FSDP for the SNL problem (4) with exact distances in Section 3.1, and then the exploitation of sparsity of FSDP, which leads to SFSDP, in Section 3.2. Most of the discussion is valid for the problem (5) with noisy distances.

3.1 The Biswas-Ye SDP Relaxation of the SNL Problem

Let

$$Y_{pq} = \sum_{i=1}^{\ell} X_{ip}X_{iq}, \quad \text{or} \quad \boldsymbol{Y} = \boldsymbol{X}^T\boldsymbol{X} \in \mathbb{S}^m.$$

Here $\mathbb{S}^m$ denotes the set of $m \times m$ symmetric matrices. Then, the problem (4) can be written as

$$\left.\begin{array}{ll} \text{minimize} & 0 \\ \text{subject to} & d_{pq}^2 = Y_{pp} + Y_{qq} - 2Y_{pq} \quad (p,q) \in \mathcal{N}_x, \\ & d_{pr}^2 = \|\boldsymbol{a}_r\|^2 - 2\sum_{i=1}^{\ell} X_{ip}a_{ir} + Y_{pp} \ (p,r) \in \mathcal{N}_a, \\ & \boldsymbol{Y} = \boldsymbol{X}^T\boldsymbol{X}. \end{array}\right\}$$

We now relax the nonconvex equality constraint $\boldsymbol{Y} = \boldsymbol{X}^T\boldsymbol{X}$ to the matrix inequality constraint $\boldsymbol{Y} \succeq \boldsymbol{X}^T\boldsymbol{X}$. Let $\boldsymbol{I}_\ell$ denotes the $\ell \times \ell$ identity matrix. Using the relation

$$\boldsymbol{Y} \succeq \boldsymbol{X}^T\boldsymbol{X} \iff \begin{pmatrix} \boldsymbol{I}_\ell & \boldsymbol{X} \\ \boldsymbol{X}^T & \boldsymbol{Y} \end{pmatrix} \succeq \boldsymbol{O},$$

we obtain the Biswas-Ye SDP relaxation [3], called the FSDP relaxation, of the SNL problem (4) without noise

$$\left.\begin{array}{ll} \text{minimize} & 0 \\ \text{subject to} & d_{pq}^2 = Y_{pp} + Y_{qq} - 2Y_{pq} \quad (p,q) \in \mathcal{N}_x, \\ & d_{pr}^2 = \|\boldsymbol{a}_r\|^2 - 2\sum_{i=1}^{\ell} X_{ip}a_{ir} + Y_{pp} \ (p,r) \in \mathcal{N}_a, \\ & \begin{pmatrix} \boldsymbol{I}_\ell & \boldsymbol{X} \\ \boldsymbol{X}^T & \boldsymbol{Y} \end{pmatrix} \succeq \boldsymbol{O}. \end{array}\right\} \tag{6}$$

Similarly, we obtain the FSDP relaxation of the SNL problem (5) with noise

$$\left.\begin{array}{ll} \text{minimize} & \displaystyle\sum_{(p,q) \in \mathcal{N}_x} (\xi_{pq}^+ + \xi_{pq}^-) + \sum_{(p,r) \in \mathcal{N}_a} (\xi_{pr}^+ + \xi_{pr}^-) \\ \text{subject to} & \hat{d}_{pq}^2 = Y_{pp} + Y_{qq} - 2Y_{pq} + \xi_{pq}^+ - \xi_{pq}^- \quad (p,q) \in \mathcal{N}_x, \\ & \hat{d}_{pr}^2 = \|\boldsymbol{a}_r\|^2 - 2\displaystyle\sum_{i=1}^{\ell} X_{ip}a_{ir} + Y_{pp} + \xi_{pr}^+ - \xi_{pr}^- \quad (p,r) \in \mathcal{N}_a, \\ & \xi_{pq}^+ \geq 0, \ \xi_{pq}^- \geq 0 \quad (p,q) \in \mathcal{N}_x, \quad \xi_{pr}^+ \geq 0, \ \xi_{pr}^- \geq 0 \quad (p,r) \in \mathcal{N}_a, \\ & \begin{pmatrix} \boldsymbol{I}_\ell & \boldsymbol{X} \\ \boldsymbol{X}^T & \boldsymbol{Y} \end{pmatrix} \succeq \boldsymbol{O}. \end{array}\right\} \tag{7}$$

3.2 Exploiting Sparsity

The sparsity of the SNL problem (4) and its SDP relaxation (6) is first extracted and then exploited. To describe the procedure, we introduce a graph $G(V_x, \mathcal{N}_x)$ consisting of the node set $V_x = \{1, 2, \ldots, m\}$ (the index set of sensors) and the edge set $\mathcal{N}_x$. Let $G(V_x, \widetilde{\mathcal{N}}_x)$ be a chordal extension of $G(V_x, \mathcal{N}_x)$, and $C_1, \ldots, C_k$ the maximal cliques of $G(V_x, \widetilde{\mathcal{N}}_x)$. Recall that $\mathcal{N}_x$, a subset of $\mathcal{N}_x^\rho$, is chosen before constructing the SNL problem (4) in Section 2. Assume that the sizes of the maximum cliques $C_1, \ldots, C_k$ are small. Selecting $\mathcal{N}_x$ that satisfies this assumption will be discussed in Section 4.

Now, we derive the sparse SDP relaxation of the SNL problem. For details, we refer to [14]. Let $\boldsymbol{Z} \in \mathbb{S}^{\ell+m}$ be a variable matrix of the form

$$\boldsymbol{Z} \;=\; \begin{pmatrix} \boldsymbol{W} & \boldsymbol{X} \\ \boldsymbol{X}^T & \boldsymbol{Y} \end{pmatrix},\ \boldsymbol{W} \in \mathbb{S}^{\ell},\ \boldsymbol{X} \in \mathbb{R}^{\ell \times m},\ \boldsymbol{Y} \in \mathbb{S}^{m}. \tag{8}$$

We can rewrite the SDP (6) as a standard primal SDP form

$$\text{minimize } \boldsymbol{A}_0 \bullet \boldsymbol{Z} \text{ subject to } \boldsymbol{A}_t \bullet \boldsymbol{Z} = b_t \ (t \in \Lambda)\ \boldsymbol{Z} \succeq \boldsymbol{O}, \tag{9}$$

where Λ denotes a finite index set. We then apply the conversion method [21] to (9) as follows. Consider the index set $\mathcal{V}$ of rows (and columns) of the matrix variable $\boldsymbol{Z}$. Let us assume that the rows and columns of matrix $\boldsymbol{Z}$ in (8) are indexed in the lexicographical order as $10, \ldots, \ell 0, *1, \ldots, *m$. Then, $\mathcal{V} = \{10, \ldots, \ell 0, *1, \ldots, *m\}$ for (6), where $*$ denotes a fixed symbol or integer larger than ℓ, and each element of $\boldsymbol{A}_t$ can be written as $[\boldsymbol{A}_t]_{ipjq}$ with $ip \in \mathcal{V}$ and $jq \in \mathcal{V}$.

We define the *aggregated sparsity pattern* $\mathcal{E}$ of the data matrices as in [11] for the description of sparsity exploitation in the sparse SDP relaxation

$$\mathcal{E} \;=\; \{(ip, jq) \in \mathcal{V} \times \mathcal{V} : ip \neq jq, [\boldsymbol{A}_t]_{ipjq} \neq 0 \text{ for some } t \in \Lambda\}.$$

A geometrical representation of the aggregated sparsity pattern is considered with a graph $G(\mathcal{V}, \mathcal{E})$. To construct a chordal extension $G(\mathcal{V}, \widetilde{\mathcal{E}})$ of $G(\mathcal{V}, \mathcal{E})$ by simulating a chordal extension from $G(V_x, \mathcal{N}_x)$ to $G(V_x, \widetilde{\mathcal{N}}_x)$, we let

$$\begin{aligned} \widetilde{\mathcal{E}} \;&=\; \{(i0, j0) : 1 \leq i < j \leq \ell\} \cup \{(i0, *p) : 1 \leq i \leq \ell,\ 1 \leq p \leq m\} \\ &\quad \cup \{(*p, *q) : (p, q) \in \widetilde{\mathcal{N}}_x\}, \\ \widetilde{\mathcal{C}}_h \;&=\; \{10, \ldots, \ell 0\} \cup \{*p : p \in C_h\} \quad (1 \leq h \leq k). \end{aligned}$$

Using the information on the chordal graph $G(\mathcal{V}, \widetilde{\mathcal{E}})$ and its maximal cliques $\widetilde{\mathcal{C}}_1, \ldots, \widetilde{\mathcal{C}}_k$, application of the conversion method [21] to (9) leads to an SDP problem

$$\left.\begin{array}{ll} \text{minimize} & \boldsymbol{A}_0 \bullet \boldsymbol{Z} \\ \text{subject to} & \boldsymbol{A}_t \bullet \boldsymbol{Z} = b_t \ (t \in \Lambda),\ \boldsymbol{Z}_{\widetilde{\mathcal{C}}_h, \widetilde{\mathcal{C}}_h} \succeq \boldsymbol{O} \ (1 \leq h \leq k), \end{array}\right\} \tag{10}$$

where $\boldsymbol{Z}_{\widetilde{\mathcal{C}}_h, \widetilde{\mathcal{C}}_h}$ denotes a submatrix of $\boldsymbol{Z}$ consisting of the elements $\boldsymbol{Z}_{ipjq}$ $(ip \in \widetilde{\mathcal{C}}_h,\ jq \in \widetilde{\mathcal{C}}_h)$.

Rewriting (10), we obtain the sparse SDP relaxation of the SNL problem (4) without noise

$$\left.\begin{array}{ll} \text{minimize} & 0 \\ \text{subject to} & d_{pq}^2 = Y_{pp} + Y_{qq} - 2Y_{pq} \quad (p, q) \in \mathcal{N}_x, \\ & d_{pr}^2 = \|\boldsymbol{a}_r\|^2 - 2\displaystyle\sum_{i=1}^{\ell} X_{ip} a_{ir} + Y_{pp} \ (p, r) \in \mathcal{N}_a, \\ & \begin{pmatrix} \boldsymbol{I}_\ell & (\boldsymbol{x}_p : p \in C_h) \\ (\boldsymbol{x}_p : p \in C_h)^T & \boldsymbol{Y}_{C_h, C_h} \end{pmatrix} \succeq \boldsymbol{O} \quad (1 \leq h \leq k), \end{array}\right\} \tag{11}$$

where $(\boldsymbol{x}_p : p \in C_h)$ denotes the $\ell \times \#C_h$ matrix variable with $\boldsymbol{x}_p$ $(p \in C_h)$ and $\boldsymbol{Y}_{C_h, C_h}$ a submatrix of $\boldsymbol{Y}$ with elements $\boldsymbol{Y}_{pq}$ $(p \in C_h,\ q \in C_h)$.

Note that (11) is not the standard SDP form because some of the variables $\boldsymbol{x}_p$ ($p \in V_x$) and Y_{pq} ($(p,q) \in \mathcal{N}_x$) appear more than once in the positive semidefinite constraints. To convert (11) to the standard SDP form, we use the domain-space conversion method [16, 17], which was successfully implemented in SparsePOP, a Matlab package for polynomial optimization problems [29]. The resulting SDP involves k small-sized positive semidefinite matrix variables induced from the k positive semidefinite constraints in (11). In contrast, the SDP (6) involves a single large-sized $(\ell + m) \times (\ell + m)$ matrix variable $\boldsymbol{Z}$ given in (8). Furthermore, the resulting SDP is expected to satisfy the correlative sparsity, which characterizes the sparsity of the Schur complement matrix. We note that the Schur complement matrix is the coefficient matrix of a system of linear equations that needs to be solved for a search direction by Cholesky factorization at every iteration of the primal-dual interior-point methods. These two properties for the resulting SDP, multiple but small-sized positive-semidefinite matrix variables and the correlative sparsity, greatly enhance the computational efficiency. See Kobayashi et al. [16] for more details of the correlative sparsity.

Let us denote

$$L_d \;=\; \left\{ \boldsymbol{X} \in \mathbb{R}^{\ell \times m} : \begin{array}{l} (\boldsymbol{X}, \boldsymbol{Y}) \text{ is a solution of (6)} \\ \text{for some } \boldsymbol{Y} \in \mathbb{S}^m \end{array} \right\}.$$

In addition, let L_s denote the set of solutions of the sparse SDP relaxation (11). We note that $L_d = L_s$ is shown in [14], which indicates that the same quality of solutions can be obtained by the sparse SDP relaxation as FSDP. Hence, the sparse SDP relaxation can achieve better computational performance for the same quality of solutions as FSDP.

The ESDP and NSDP relaxations of SNL problems were proposed in [30] as further relaxations of the FSDP relaxation. In essence, a generalization of the sparse SDP relaxation (11) includes NSDP and ESDP. In other words, if we denote by L_n and L_e the solution sets of the NSDP and ESDP relaxation, respectively, then, by construction, we know $L_d \subseteq L_n \subseteq L_e$. It was shown in [30] that if the underlying graph $G(V_x, \mathcal{N}_x)$ is chordal, then $L_d = L_n$. In this case, we know $G(V_x, \mathcal{N}_x) = G(V_x, \widetilde{\mathcal{N}}_x)$ and that $L_d = L_s = L_n$ also follows from Proposition 3.3 in [14]. For details, we refer to [14]. We used ESDP for the numerical experiments shown in Section 6 because it is known to be more efficient than NSDP.

We mention that the technique described in this section is a fairly general technique for exploiting the sparsity of SDPs. The important feature of this technique is to convert a sparse SDP into another equivalent SDP that can be solved efficiently by exploiting its sparsity. From the construction of SDP relaxations, FSDP and SFSDP can be expected to attain more accurate solutions than ESDP and NSDP. Moreover, if the size of the maximal cliques $C_1, C_2, \ldots, C_k$ of $G(V_x, \widetilde{\mathcal{N}}_x)$ are small, then the performance of SFSDP will be better than other SDP relaxations.

We conclude this section by describing the sparse SDP relaxation of the SNL problem

(5) with noise

$$\left.\begin{array}{ll}
\text{minimize} & \displaystyle\sum_{(p,q)\in\mathcal{N}_x}(\xi^+_{pq}+\xi^-_{pq}) + \sum_{(p,r)\in\mathcal{N}_a}(\xi^+_{pr}+\xi^-_{pr}) \\
\text{subject to} & \hat{d}^2_{pq} = Y_{pp}+Y_{qq}-2Y_{pq}+\xi^+_{pq}-\xi^-_{pq} \quad (p,q)\in\mathcal{N}_x, \\
& \hat{d}^2_{pr} = \|\boldsymbol{a}_r\|^2 - 2\displaystyle\sum_{i=1}^{\ell} X_{ip}a_{ir} + Y_{pp}+\xi^+_{pr}-\xi^-_{pr} \quad (p,r)\in\mathcal{N}_a, \\
& \xi^+_{pq}\geq 0,\ \xi^-_{pq}\geq 0 \quad (p,q)\in\mathcal{N}_x,\quad \xi^+_{pr}\geq 0,\ \xi^-_{pr}\geq 0 \quad (p,r)\in\mathcal{N}_a, \\
& \begin{pmatrix} \boldsymbol{I}_\ell & (\boldsymbol{x}_p : p\in C_h) \\ (\boldsymbol{x}_p : p\in C_h)^T & \boldsymbol{Y}_{C_h,C_h} \end{pmatrix} \succeq \boldsymbol{O} \quad (1\leq h\leq k).
\end{array}\right\} \tag{12}$$

4 Some Implementation Issues

Let $\overline{\mathcal{N}}_x \subset \mathcal{N}^\rho_x$ be input set of pairs of sensors and $\overline{\mathcal{N}}_a \subset \mathcal{N}^\rho_a$ input set of pairs of a sensor and an anchor, and the (noisy) distance information be given for $\overline{\mathcal{N}}_x$ and $\overline{\mathcal{N}}_a$. Although $\overline{\mathcal{N}}_x$ and $\overline{\mathcal{N}}_a$ can be used for $\mathcal{N}_x$ and $\mathcal{N}_a$ to construct the sparse SDP relaxations (11) and (12) , we take a subset of $\overline{\mathcal{N}}_x$ for $\mathcal{N}_x$ and a subset of $\overline{\mathcal{N}}_a$ for $\mathcal{N}_a$ to reduce their sizes. In general, more accurate locations of sensors are obtained in longer computational time as the size of $\mathcal{N}_x \subset \overline{\mathcal{N}}_x$ and $\mathcal{N}_a \subset \overline{\mathcal{N}}_a$ increases. SFSDP incorporates a method for selecting $\mathcal{N}_x$ from $\overline{\mathcal{N}}_x$ and $\mathcal{N}_a$ from $\overline{\mathcal{N}}_a$ so that the resulting sparse SDP relaxation can be solved efficiently and find accurate locations of sensors.

In addition, SFSDP provides a regularization term [19] in the objective function to obtain accurate locations of sensors for anchor-free problems.

The method for selecting $\mathcal{N}_a$ is presented in Section 4.1, $\mathcal{N}_x$ in Section 4.2. Four types objective functions used in SFSDP are shown in Section 4.3. We use the notation V_x for the set of sensors, and $\deg(p,\overline{\mathcal{N}})$ for the number of edges incident to a sensor $p \in V_x$ for every subset $\overline{\mathcal{N}}$ of $\overline{\mathcal{N}}_x \cup \overline{\mathcal{N}}_a$.

4.1 Selecting edges from $\overline{\mathcal{N}}_a$ for $\mathcal{N}_a$

For the problem with exact distances, we need to select at least $\ell+1$ edges, incident to each sensor, from $\overline{\mathcal{N}}_x \cup \overline{\mathcal{N}}_a$ to locate all sensors. In case of the problem with noisy distances, more edges are needed to have accurate locations of sensors. We also note that the total number of elements in $\mathcal{N}_x$ and $\mathcal{N}_a$ is equal to the number of equality constraints in the sparse SDP relaxations (11) and (12). In addition, the construction of k positive semidefinite constraints in (11) and (12) does not depend on $\mathcal{N}_a$ because the maximal cliques $C_1, C_2, \ldots, C_k$ of a chordal extension of the graph $G(V_x, \mathcal{N}_x)$ determine the positive semidefinite constraints as shown in Section 3.2. As the number of edges selected from $\overline{\mathcal{N}}_a$ for $\mathcal{N}_a$ increases, the number of edges that need to be selected from $\overline{\mathcal{N}}_x$ for $\mathcal{N}_x$ becomes smaller. Thus, selecting edges from $\overline{\mathcal{N}}_a$ first for $\mathcal{N}_a$ provides smaller-sized maximal cliques than selecting edges from $\overline{\mathcal{N}}_x$ first for $\mathcal{N}_x$. Since it is more efficient to solve the resulting sparse SDP relaxation from smaller-sized maximal cliques, we give a priority to the edges of $\overline{\mathcal{N}}_a$ over the edges of $\overline{\mathcal{N}}_x$ when selecting edges from $\overline{\mathcal{N}}_x$ and $\overline{\mathcal{N}}_a$ for the construction of the sparse SDP relaxations

(11) and (12). Let $\nu = \ell + 1$ if all distances are exact and $\nu = 2 \times (\ell + 1)$ otherwise. SFSDP selects $\min\{\nu, \deg(p, \overline{\mathcal{N}}_a)\}$ edges from $\overline{\mathcal{N}}_a$ for each $p \in V_x$ to form $\mathcal{N}_a$.

4.2 Selecting edges from $\overline{\mathcal{N}}_x$ for $\mathcal{N}_x$

Assume that $\mathcal{N}_a$ has been constructed by the method described in Section 4.1 and fixed. For selection of edges from $\overline{\mathcal{N}}_x$ for $\mathcal{N}_x$, we try to satisfy two conditions whenever possible: (i) at least κ edges from $\overline{\mathcal{N}}_x \cup \overline{\mathcal{N}}_a$ are incident to each sensor, where κ is taken to be not less than $\ell + 1$, (ii) at least one edge from $\overline{\mathcal{N}}_x$ needs to be selected to form $\mathcal{N}_x$ for each sensor. Define

$$\kappa_p = \kappa - \min\{\deg(p, \mathcal{N}_a),\ \ell\} \text{ for every sensor } p \in V_x.$$

We consider a family $\mathcal{G}_\kappa$ of subgraphs $G(V_x, \mathcal{N}_x)$ of $G(V_x, \overline{\mathcal{N}}_x)$ satisfying $\deg(p, \mathcal{N}_x) \geq \kappa_p$ for every sensor $p \in V_x$. SFSDP selects edges from $\overline{\mathcal{N}}_x$ for $\mathcal{N}_x$ such that $G(V_x, \mathcal{N}_x)$ is a minimal graph in the class $\mathcal{G}_\kappa$ by applying a heuristic method:

Step 0: Let $V_x = \{1, 2, \ldots, m\}$, $p = 1$ and $\mathcal{N}_x = \emptyset$.

Step 1: If $p = m$, stop.

Step 2: For every edge $(p, q) \in \overline{\mathcal{N}}_x$ with $q \geq p+1$, if either $\deg(p, \mathcal{N}_x) < \kappa_p$ or $\deg(q, \mathcal{N}_x) < \kappa_q$, then let $\mathcal{N}_x = \mathcal{N}_x \cup \{(p, q)\}$.

Step 3: Let $p = p + 1$ and go to Step 1.

This heuristic method is easy to implement. From the numerical experiments in Section 6, we have confirmed the effectiveness of the method. More precisely, this method effectively constructs a graph $G(V_x, \mathcal{N}_x)$ that leads to a chordal extension with small-sized maximal cliques $C_1, C_2, \ldots, C_k$, even when the original graph $G(V_x, \overline{\mathcal{N}}_x)$ is dense. We can verify that if $G(V_x, \overline{\mathcal{N}}_x)$ is a complete graph, as an extreme case, the method generates $\mathcal{N}_x = \widetilde{\mathcal{N}}_x = \{(p, q) : p < q \leq m, 1 \leq p \leq \kappa\}$. Thus, the graph $G(V_x, \mathcal{N}_x)$ induces a chordal extension $(V_x, \widetilde{\mathcal{N}}_x)$ having maximal cliques $\{1, 2, \ldots, \kappa, q\}$ $(q = \kappa + 1, \ldots, m)$ with the size $\kappa + 1$.

With an increasingly large value for κ, we can expect to obtain more accurate locations of sensors, although it takes longer to solve the sparse SDP relaxation problem. Some applications of the SNL problem, *e.g.*, molecular conformation, have not enough distance information to obtain accurate solutions. For these applications, κ can be set not to reduce the size of $\overline{\mathcal{N}}_x$. In SFSDP, a parameter named pars.minDegree can be specified for the value of κ. If users choose pars.minDegree $= \kappa \geq 100$, then $\mathcal{N}_x = \overline{\mathcal{N}}_x$ and $\mathcal{N}_a = \overline{\mathcal{N}}_a$ are used. The default value for pars.minDegree is $\ell + 2$.

4.3 Selecting Objective Functions

SFSDP provides four objective functions to select. Depending on whether the problem contains noise and whether no or a small number of anchors are available, one of the four objective functions can be selected by setting a value of 0-3 for the parameter called pars.objSW.

Functions	Description
SFSDPplus.m	A function that analyzes the given input data and calls SFSDP.m.
SFSDP.m	A solver that solves the problem and returns the locations of sensors, and can be called directly by users.
generateProblem.m	A function that generates an SNL problem with the given parameters.
test_SFSDP.m	A function that solves the problem generated by generateProblem.m using SFSDPplus.m.

Table 1: Functions provided in SFSDP

Users can set pars.objSW = 0 for the zero objective function to solve the SNL problem (4) with exact distances, and pars.objSW = 1 for the sum of the 1-norm error to solve the SNL problem (5) with noisy distances. But these objective functions often fail to provide accurate locations of sensors if no or only a small number of anchors are available. In such cases, an objective function with the regularization term [2, 19]

$$- \sum_{(p,q) \in \widetilde{\mathcal{N}}_x} \|\boldsymbol{x}_p - \boldsymbol{x}_q\|^2 \tag{13}$$

gives more accurate locations. Recall that $\widetilde{\mathcal{N}}_x$ denotes the edge set of a chordal extension $G(V_x, \widetilde{\mathcal{N}}_x)$ of the graph of $G(V_x, \mathcal{N}_x)$, which was introduced for constructing the sparse SDP relaxations (11) and (12) in Section 3. Take pars.objSW = 2 and 3 to add the regularization term described in (13) to the zero objective function and to the sum of the 1-norm error, respectively.

5 Examples on Executing SFSDP

SFSDP includes four Matlab functions. Their names and functionality are described in Table 1. We will briefly describe the usage of the function SFSDPplus.m using SNL examples. See the user's guide [15] for more details.

Input for SFSDPplus.m consists of the dimension of space where sensors and anchors are located, the number of sensors, the number of anchors, the locations of anchors, a distance matrix between sensors and anchors, and optional control parameters. SFSDPplus.m outputs the locations of sensors and anchors in the form of a matrix called xMatrix. Detailed description on input is given in Table 2.

Input data for a simple SNL example with 3 sensors and 4 anchors is stored in example1.mat.

```
>> load example1.mat
```

Then,

$$\text{sDim} = 2,\ \text{noOfSensors} = 3,\ \text{noOfAnchors} = 4,$$

Variable name	Description
sDim	The dimension of space where sensors and anchors are located.
noOfSensors	The number m of sensors
noOfAnchors	The number m_a of anchors
xMatrix0	sDim$\times m_a$ matrix of the location of anchors in the sDim-dimensional space; if noOfAnchors = 0, then xMatrix0 can be []. Or sDim$\times(m+m_a)$ matrix of sensors and anchors in the sDim-dimensional space; anchors are placed in the last m_a columns.
distanceMatrix0	The sparse (and noisy) distance matrix. distanceMatrix0(p,q) = (noisy) distance between xMatrix0(:,p) and xMatrix0(:,q) if $p<q$, and distanceMatrix0$(p,q)=0$ if $p \geq q$ or the (noisy) distance between xMatrix0(:,p) and xMatrix0(:,q) is not available.
pars	Control parameters in constructing an SDP relaxation problem and solving it by SeDuMi or SDPA, which include pars.minDegree $\geq$ sDim+1, pars.objSW $\in \{0,\ 1,\ 2,\ 3\}$, and pars.SDPsolver $\in$ { 'sedumi', 'sdpa' }. See Table 4 of the user's guide [15].

Table 2: Input for SFSDPplus.m and SFSDP.m

$$\text{xMatrix0} = \begin{pmatrix} 0.0 & 0.0 & 1.0 & 1.0 \\ 0.0 & 1.0 & 0.0 & 1.0 \end{pmatrix},$$

$$\text{distanceMatrix0} = \begin{pmatrix} 0 & 0.2196 & 0.4147 & 0.5016 & 0.7566 & 0 & 0 \\ 0 & 0 & 0.3562 & 0.6290 & 0.4713 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.6471 & 0.4689 \end{pmatrix},$$

pars.minDegree = 4, pars.SDPsolver = 'sdpa', pars.objSW = 1

are loaded in the workspace of Matlab. Execute SFSDP by

```
>> [xMatrix] = SFSDPplus(sDim,noOfSensors,noOfAnchors,xMatrix0,...
               distanceMatrix0,pars);
```

As a result, we obtain estimated locations of the sensors

$$\text{xMatrix(:,1:noOfSensors)} \ = \ \begin{pmatrix} 0.3621 & 0.3025 & 0.6926 \\ 0.3525 & 0.5809 & 0.6019 \end{pmatrix}.$$

At the end of computation, SFSDPplus.m refines the locations with the function refineposition.m, which is a Matlab implementation of the gradient method by Toh [20], and displays two figures, the one before the refinement and the other after the refinement, with computed locations of the sensors.

When we want to evaluate the performance of SFSDP using an SNL problem with known locations of sensors, we can include the locations of sensors in xMatrix0. We show such an SNL example with 500 sensors and 100 anchors distributed randomly in $[0,1]\times[0,1]$.

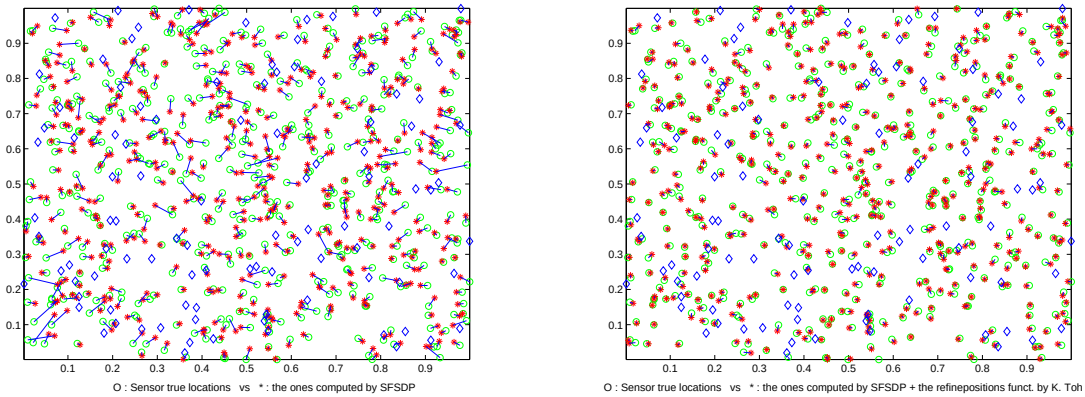

Figure 1: A 2-dimensional problem with 500 sensors, 100 anchors and noisy distances. Before and after the refinement by the gradient method. A circle denotes the true location of a sensor, $\star$ the computed location of a sensor, and a line segment a deviation from the true and computed location.

```
>> load d2n01s500a100.mat
>> [xMatrix] = SFSDPplus(sDim,noOfSensors,noOfAnchors,xMatrix0,...
               distanceMatrix0,pars);
## sDim = 2, noOfSensors = 500, noOfAnchors = 100
## the number of dist. eq. between two sensors  = 8171
## the number of dist. eq. between a sensor & an anchor = 3000
## the min., max. and ave. degrees over sensor nodes = 24, 167,  38.68

. . . . . .

## elapsed time for SDP solver =     2.31
## mean error in dist. eq. = 2.93e-04, max. error in dist. eq. = 9.63e-02
## rmsd = 2.27e-02
## see Figure 101
## elapsed time for a gradient method =     0.54
## mean error in dist. eq. = 1.99e-04, max. error in dist. eq. = 6.36e-02
## rmsd = 7.76e-03
## see Figure 103
```

At the end of computation, SFSDPplus will display the results as shown in Figure 1.

6 Numerical Results

One feature of SFSDP is that either of SDPA and SeDuMi can be used for solving the SDP relaxation. We first compare the performance of

- ESDP using SeDuMi and SDPA

- FSDP using SeDuMi and SDPA
- SFSDP using SeDuMi and SDPA

applied to 2-dimensional SNL problems with 1000 to 5000 sensors. This comparison exhibits that SFSDP using SDPA is more efficient to compute the locations of sensors with higher accuracy than the other methods. After the comparison, the performance of SFSDP with SDPA is further tested on larger-sized 2-dimensional problems with 20000 sensors, 3-dimensional problems with 3000 to 5000 sensors, and 3-dimensional anchor-free problems with 3000 to 5000 sensors. Numerical experiments except for Section 6.2 were performed on 2.8GHz Quad-Core Intel Xeon with 4GB memory using SDPA 7.3.1 and SeDuMi 1.1R3. The problems with 20000 sensors in Section 6.2 were solved on 2.8GHz Quad-Core Intel Core i7 with 16GB memory using SDPA 7.3.1 and SeDuMi 1.21.

Throughout our numerical experiments, we generated sensors $\boldsymbol{a}_p$ $(p = 1, 2, \ldots, m)$ randomly in the unit square $[0,1]^2$ for 2-dimensional problems or in the unit cube $[0,1]^3$ for 3-dimensional problems. The number of sensors ranged from 1000, 3000, 5000 to 20000 for 2-dimensional problems, and 3000 to 5000 for 3-dimensional problems. Two types of radio ranges were used. The first type is a constant radio range $\rho = 0.1$ for 2-dimensional problems (or $\rho = 0.250$ for 3-dimensional problems), which is independent of the number of sensors. The second type is $\rho = \sqrt{10/m}$ for 2-dimensional problems (or $\rho = (15/m)^{1/3}$ for 3-dimensional problems), where m denotes the number of sensors. The second type indicates that each square of size ρ in $[0,1]^2$ contains 10 randomly generated sensors on average for 2-dimensional problems or each cube of size ρ in $[0,1]^3$ contains 15 randomly generated sensors on average for 3-dimensional problems. Anchors were randomly distributed in the unit square $[0,1]^2$ for 2-dimensional problems or in the unit cube $[0,1]^3$ for 3-dimensional problems. Or, they were placed at the corners of the unit square $[0,1]^2$ or the unit cube $[0,1]^3$. In the problems with randomly distributed anchors, the number of anchors was 10% or 5% of the number of sensors. The noisy factor was changed from 0.0 to 0.2. The distances were perturbed to create problems with noise:

$$\begin{array}{l} \hat{d}_{pq} = \max\{(1+\sigma\epsilon_{pq}),\ 0.1\}\|\boldsymbol{a}_p - \boldsymbol{a}_q\| \ ((p,q) \in \overline{\mathcal{N}}_x = \mathcal{N}_x^\rho), \\ \hat{d}_{pr} = \max\{(1+\sigma\epsilon_{pr}),\ 0.1\}\|\boldsymbol{a}_p - \boldsymbol{a}_r\| \ ((p,r) \in \overline{\mathcal{N}}_a = \mathcal{N}_a^\rho), \end{array} \tag{14}$$

where $\sigma \geq 0$ denotes a noisy factor, and ϵ_{pq} and ϵ_{pr} are chosen from the standard normal distribution $N(0,1)$, and $\boldsymbol{a}_p$ denotes the true location of the pth sensor. Note that we set $\overline{\mathcal{N}}_x = \mathcal{N}_x^\rho$ and $\overline{\mathcal{N}}_x = \mathcal{N}_x^\rho$ when generating the test problems. As in [3, 4, 5, 27, 30], the root mean square distance (RMSD)

$$\left(\frac{1}{m}\sum_{p=1}^{m}\|\boldsymbol{x}_p - \boldsymbol{a}_p\|^2\right)^{1/2},$$

where $\boldsymbol{x}_p$ denotes the computed locations of the pth sensor, is used to measure the accuracy of locations of m sensors computed by SDPA or SeDuMi, and the accuracy of refined solutions by the gradient method.

In the numerical experiments, each type of the test problems was generated 5 times and tested. The values of RMSD and elapsed time in the Tables are the average of values from

those 5 experiments. In particular, the value of RMSD was computed by

$$\left(\frac{1}{5m}\sum_{k=1}^{5}\sum_{p=1}^{m}\|\boldsymbol{x}_p^k-\boldsymbol{a}_p^k\|^2\right)^{1/2},$$

where $\boldsymbol{a}_p^k$ and $\boldsymbol{x}_p^k$ denote the true and computed locations of the pth sensor of the kth instance of test problems.

6.1 Comparison of SFSDP with FSDP and ESDP for Two-dimensional Problems

In Table 3, the RMSD and elapsed time for ESDP, FSDP and SFSDP are compared for the problems with 1000 sensors and $\rho = 0.1$. We let λ denote an upper bound for the degree of any sensor node in ESDP, and κ a lower bound for the degree of any sensor node described in Section 4.2. See also Section 4.1 of [14] for their precise definitions and the comparison. FSDP, ESDP and SFSDP were tested with SDPA and SeDuMi. After obtaining a solution by an SDP solver, the solution was refined using the gradient method. The two columns under RMSD indicate the values of RMSD before and after the refinement. In all tested problems, we observe that SDPA provides a solution much faster than SeDuMi with comparable values of RMSD, and FSDP is much slower than ESDP and SFSDP. Notice that SFSDP attains solutions as accurately as FSDP. From these observations, we compare ESDP and SFSDP both using SDPA in the rest of Section 6.1.

Numerical tests with increasing values of the parameters λ and κ for ESDP and SFSDP, respectively, were performed and the results are shown in the bottom of Table 3. As the parameters increase, both ESDP and SFSDP with the gradient method spent longer elapsed time and obtained more accurate values of RMSD. Notice that SFSDP(3) resulted in more accurate RMSD than ESDP(8).

Numerical results from test problems with 3000 and 5000 sensors are shown in Table 4. The number of anchors was changed from 4 to 10% of m, the radio range was fixed to 0.1 or computed by $\sqrt{10/m}$, and the noisy factor was changed from 0.0 to 0.2. SDPA was used to solve the SDPs. We observe:

(i) Total time spent by SFSDP and elapsed time by SDPA are shorter for the problems with exact distances ($\sigma = 0.0$) than noisy distances ($\sigma = 0.1$ or 0.2). As mentioned in Section 4.3, the objective function is set to zero for the problems with exact distances. This makes the size of SDP relaxations smaller.

(ii) With a smaller number of anchors, SFSDP with SDPA consumed longer total time and elapsed time, respectively. Recall that if the number of anchors decreases, then the candidates for $\mathcal{N}_a$ becomes less and more edges need to be selected from $\mathcal{N}_x^\rho$ for $\mathcal{N}_x$ in the construction of the SDP relaxation. Thus, the graph $G(V_x, \mathcal{N}_x)$ becomes denser and the sizes of the positive semidefinite constraints in the resulting SDP relaxation grow. See Sections 4.1 and 4.2.

Test problems				RMSD		Elapsed time		
$m,\ m_a,\ \rho$	σ	SDP($\lambda\|\kappa$)	Solver	SDP	w.Grad.	Solver	Grad.	Total
$m = 1000$,	0.0	FSDP(4)	SeDuMi	4.2e-5	7.1e-6	2907.2	0.1	2910.3
$m_a = 100$			SDPA	5.0e-4	1.0e-5	53.8	0.2	56.9
distributed		ESDP(5)	SeDuMi	1.2e-3	1.4e-4	56.3	0.6	78.5
randomly.			SDPA	1.0e-3	1.5e-4	19.4	0.6	43.2
$\rho = 0.100$		SFSDP(4)	SeDuMi	6.4e-6	2.1e-6	7.4	0.1	12.1
			SDPA	4.9e-5	6.0e-6	2.3	0.3	7.3
	0.1	FSDP(4)	SeDuMi	1.7e-2	7.1e-3	5285.7	1.5	5294.0
			SDPA	1.7e-2	7.1e-3	308.6	1.5	317.3
		ESDP(5)	SeDuMi	1.6e-2	7.9e-3	42.5	1.7	66.3
			SDPA	1.6e-2	7.7e-3	13.2	1.5	39.1
		SFSDP(4)	SeDuMi	1.7e-2	7.0e-3	20.2	6.1	34.9
			SDPA	1.7e-2	7.1e-3	5.1	5.0	18.9
$m = 1000$,	0.0	FSDP(4)	SeDuMi	3.0e-5	1.3e-5	2186.3	0.2	2189.9
$m_a = 4$			SDPA	5.5e-5	2.1e-5	45.4	0.3	48.8
at corners,		ESDP(5)	SeDuMi	3.9e-2	1.7e-2	47.9	13.6	80.9
$\rho = 0.100$			SDPA	3.7e-2	1.3e-2	16.7	13.6	49.9
		SFSDP(4)	SeDuMi	1.6e-5	9.0e-6	29.3	0.1	34.4
			SDPA	5.8e-5	2.5e-5	12.6	0.3	17.8
	0.1	FSDP(4)	SeDuMi	4.5e-2	1.0e-2	2921.6	14.6	2943.4
			SDPA	4.5e-2	1.0e-2	157.2	14.9	179.5
		ESDP(5)	SeDuMi	4.8e-2	2.0e-2	43.2	13.5	75.7
			SDPA	4.8e-2	1.9e-2	16.8	13.8	50.5
		SFSDP(4)	SeDuMi	4.5e-2	1.0e-2	45.6	12.2	66.9
			SDPA	4.5e-2	1.0e-2	18.3	12.5	39.8
		ESDP(4)	SDPA	4.7e-02	2.4e-02	13.6	8.9	40.1
		ESDP(6)	SDPA	4.9e-02	1.5e-02	20.7	10.9	52.9
		ESDP(8)	SDPA	5.2e-02	1.3e-02	31.6	12.6	70.0
		SFSDP(3)	SDPA	5.1e-02	1.2e-02	7.8	10.9	25.6
		SFSDP(5)	SDPA	4.3e-02	1.0e-02	39.9	8.8	56.5
		SFSDP(7)	SDPA	4.0e-02	7.8e-03	142.9	6.6	158.5

Table 3: Comparing FSDP(4), ESDP(5) and SFSDP(5) with SeDuMi and SDPA to solve 2-dimensional problems with 1000 sensors and $\rho = 0.1$.

(iii) A smaller value of radio range ρ requires longer total time by SFSDP and elapsed time by SDPA. Note that the number of edges in $\mathcal{N}_x^\rho$ decreases as the radio range ρ decreases. Then, selecting edges from $\mathcal{N}_x^\rho$ for $\mathcal{N}_x$ becomes restricted with a smaller number of edges in $\mathcal{N}_x^\rho$, as mentioned in the selection method. As a result, the graph $G(V_x, \mathcal{N}_x)$ becomes denser and the sizes of the positive semidefinite constraints in the resulting SDP relaxation increases. See also Sections 4.1 and 4.2.

Table 4 shows that SFSDP took longer elapsed time for solving an SDP by SDPA than ESDP for the three problems: (a) $m = 3000$, $m_a = 4$, $\rho = \sqrt{10/m} \approx 0.058$ and $\sigma > 0$, (b) $m = 5000$, $m_a = 5\%$ of $m = 250$, $\rho = \sqrt{10/m} \approx 0.045$ and $\sigma > 0$, and (c) $m = 5000$, $m_a = 4$, $\rho = \sqrt{10/m} \approx 0.045$ and $\sigma > 0$. These are due to (i), (ii) and (iii).

We see that SFSDP provides more accurate values of RMSD than ESDP in Table 4. Figure 2 and 3 show the differences in the values of RMSD from SFSDP and ESDP for the problems with 1000 to 5000 sensors in the presence of noise. Figure 2 displays the results for the problems where the number of anchors is 10% of the number of sensors. For the experiments with the noisy factor 0.1 and 0.2, the values of RMSD from SFSDP are smaller than those from ESDP for all test problems. In Figure 3, we observe similar results for the problems with 4 anchors at the corner of $[0, 1]^2$.

6.2 Two-dimensional Larger-scale Problems

In Table 5, we show the numerical results for the problems with 20000 sensors. Anchors are randomly generated or placed at the corners of the unit square $[0, 1]^2$. The noisy factor σ, was changed from 0.0 to 0.2 when generating the problems. SFSDP solved the problems efficiently with accurate values of RMSD. Numerical results in Table 5 support the remarks in (i), (ii) and (iii) in Section 6.1. We note that the test problems shown in [23] involve σ up to 0.01.

6.3 Three-dimensional Problems

Numerical results for three-dimensional problems are shown in Table 6. Test problems include 3000 to 5000 sensors with various anchor distributions. SFSDP efficiently solves three-dimensional problems with 3000 and 5000 sensors, resulting accurate values of RMSD, except for two cases. Out-of-memory error will not occur if experiments are performed on a computer with larger-sized memory. What is mentioned in (i), (ii) and (iii) in Section 6.1 can be observed.

6.4 Anchor-free Problems in Three Dimensions

SFSDP handles anchor-free problems in ℓ dimensions, $\ell = 2$ or 3, by first fixing $\ell+1$ sensors as anchors, and then applying the sparse SDP relaxation to the resulting problem. More precisely, if an anchor-free SNL problem with n sensors in the ℓ-dimensional space is given, SFSDP first chooses $\ell + 1$ sensors, e.g., sensors $n - \ell, n - \ell + 1, \ldots, n$, which are adjacent to

Test problems			ESDP(5)			SFSDP(4)		
			Elapsed time				Elapsed time	
m, m_a	ρ	σ	SDPA	Total	RMSD	RMSD	Total	SDPA
$m = 3000$,	0.100	0.0	68.0	359.2	1.7e-3	2.3e-7	36.9	6.2
$m_a = 10\%$ of m		0.1	71.3	351.3	4.3e-3	2.3e-3	95.8	15.9
$= 300$ distributed		0.2	70.1	348.0	8.1e-3	4.8e-3	98.8	15.1
randomly	$\sqrt{10/m}$	0.0	69.3	241.2	8.1e-4	4.0e-6	43.5	12.3
	≈ 0.058	0.1	47.5	216.9	4.2e-3	1.9e-3	88.2	22.4
		0.2	47.0	216.2	7.9e-3	4.4e-3	88.9	21.5
$m = 3000$,	0.100	0.0	61.8	251.6	2.1e-3	1.3e-6	37.9	7.0
$m_a = 5\%$ of m		0.1	60.3	251.1	5.6e-3	2.5e-3	88.6	15.3
$= 150$ distributed		0.2	58.8	248.9	1.1e-2	5.2e-3	91.8	14.4
randomly	$\sqrt{10/m}$	0.0	68.1	211.1	1.4e-3	9.5e-6	60.8	28.5
	≈ 0.058	0.1	44.9	194.7	7.3e-3	4.5e-3	112.0	39.9
		0.2	44.6	192.0	1.1e-2	5.8e-3	115.7	40.1
$m = 3000$,	0.100	0.0	84.6	274.0	1.2e-2	6.7e-6	74.2	36.8
$m_a = 4$ at corners		0.1	76.3	261.8	1.7e-2	5.6e-3	177.9	62.9
		0.2	76.3	263.4	3.3e-2	9.3e-3	176.7	63.3
	$\sqrt{10/m}$	0.0	71.1	258.3	7.2e-3	7.6e-5	206.6	167.5
	≈ 0.058	0.1	72.8	260.8	1.2e-2	7.1e-3	297.2	172.3
		0.2	77.2	265.4	3.3e-2	1.1e-2	312.4	176.4
$m = 5000$,	0.1	0.0	175.8	1791.1	1.1e-7	2.7e-7	94.7	11.2
$m_a = 10\%$ of m		0.1	160.2	1776.2	3.2e-3	2.2e-3	244.7	30.5
$= 500$ distributed		0.2	159.4	1779.1	6.6e-3	4.5e-3	241.4	28.9
randomly	$\sqrt{10/m}$	0.0	112.9	585.3	9.5e-4	4.4e-6	111.8	26.7
	≈ 0.045	0.1	88.0	562.3	4.2e-3	3.4e-3	219.1	28.5
		0.2	85.2	560.5	6.5e-3	4.7e-3	230.9	29.3
$m = 5000$,	0.100	0.0	140.1	916.6	6.6e-5	5.9e-7	93.6	11.5
$m_a = 5\%$ of m		0.1	123.1	899.6	4.2e-3	2.5e-3	229.8	31.3
$= 250$ distributed		0.2	121.1	911.4	8.9e-3	4.8e-3	231.3	27.5
randomly	$\sqrt{10/m}$	0.0	115.6	508.6	2.7e-3	1.5e-4	156.1	68.3
	≈ 0.045	0.1	83.3	481.0	6.8e-3	5.1e-3	283.0	98.9
		0.2	81.7	476.1	1.0e-2	6.3e-3	281.7	103.4
$m = 5000$,	0.100	0.0	164.9	668.5	1.3e-2	2.2e-5	153.6	56.7
$m_a = 4$ at corners		0.1	145.4	657.6	1.7e-2	5.4e-3	374.9	79.4
		0.2	145.3	658.0	3.1e-2	9.6e-3	328.6	79.5
	$\sqrt{10/m}$	0.0	145.3	644.8	8.1e-3	1.5e-4	604.6	502.2
	≈ 0.045	0.1	149.9	645.8	1.6e-2	6.0e-3	908.8	566.2
		0.2	155.4	659.7	4.2e-2	1.3e-2	1040.4	584.6

Table 4: Comparison between ESDP(5) and SFSDP(4) with SDPA to solve 2-dimensional problems with 3000 and 5000 sensors.

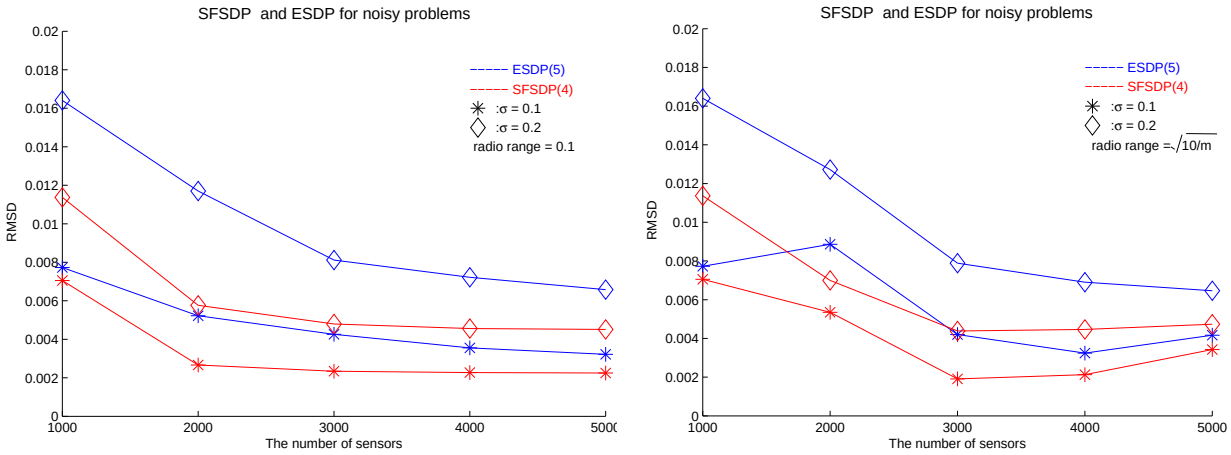

Figure 2: The number of anchors is 10 % of the number of sensors. The anchors are distributed randomly. The radio range used for the left figure is 0.1, and for the right $\sqrt{10/m}$.

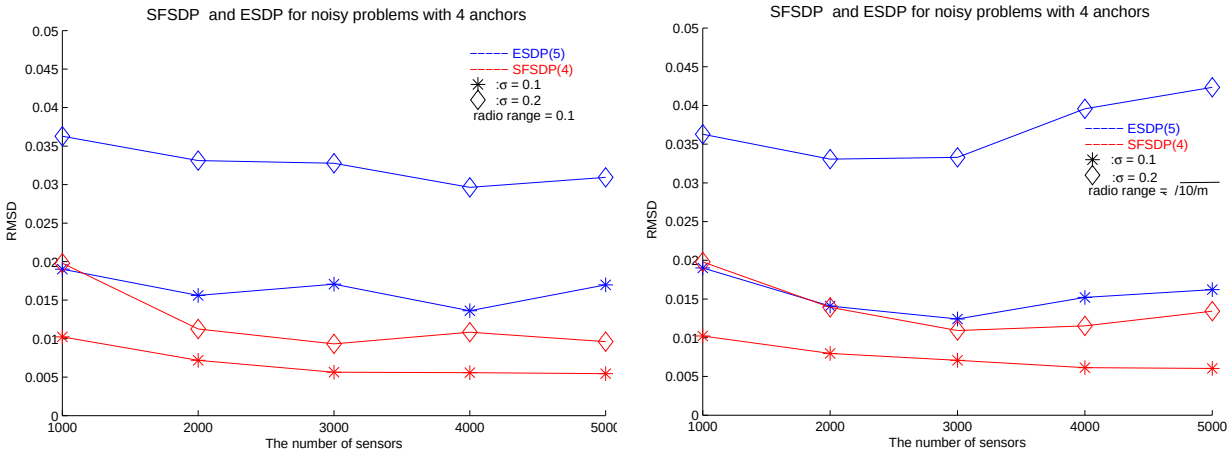

Figure 3: Four anchors are placed at the corner of $[0,1]^2$. The radio range used for the left figure is 0.1, and for the right $\sqrt{10/m}$.

Test problems			RMSD		Elapsed time		
m_a, m	ρ	σ	SDPA	w.Grad.	SDPA	Grad.	Total
$m = 20000$,	0.100	0.0	1.44e-06	3.09e-07	93.4	0.7	326.9
$m_a = 10\%$ of m		0.1	9.54e-03	2.21e-03	159.2	16.8	877.3
$= 2000$ distributed		0.2	1.97e-02	4.43e-03	148.2	23.5	882.1
randomly	$\sqrt{10/m}$	0.0	1.10e-04	4.07e-06	466.2	4.4	708.0
	≈ 0.022	0.1	3.19e-03	7.86e-04	301.5	35.9	823.0
		0.2	6.27e-03	1.50e-03	237.6	41.4	773.1
$m = 20000$,	0.100	0.0	1.44e-06	3.09e-07	88.0	0.7	320.2
$m_a = 5\%$ of m		0.1	9.54e-03	2.21e-03	158.5	14.6	877.8
$= 1000$ distributed		0.2	1.97e-02	4.44e-03	150.1	17.1	880.0
randomly	$\sqrt{10/m}$	0.0	1.96e-04	8.56e-06	1487.6	6.4	1752.8
	≈ 0.022	0.1	3.69e-03	8.76e-04	1879.2	45.5	2388.4
		0.2	7.28e-03	1.96e-03	1577.9	46.6	2096.3
$m = 20000$,	0.100	0.0	4.01e-05	6.92e-06	182.9	2.0	469.2
$m_a = 4$ at corners		0.1	4.18e-02	7.57e-03	403.0	137.7	1140.0
		0.2	6.63e-02	1.08e-02	402.6	146.0	1150.5
	$\sqrt{10/m}$ ≈ 0.022	Out-of-memory					

Table 5: Numerical results on SFSDP(4) with SDPA applied to 2-dimensional problems with 20000 sensors.

each other, and forms a nondegenerate ℓ-simplex. After temporarily fixing their locations, say $\boldsymbol{x}_r = \boldsymbol{a}_r$ $(r = n-\ell, n-\ell+1, \ldots, n)$, as anchors, SFSDP applies the sparse SDP relaxation described in Sections 3 and 4 to the resulting SNL problem with $m = n - \ell + 1$ sensors and $m_a = \ell + 1$ anchors. Then, it computes the location of sensors $\boldsymbol{x}_p$ $(p = 1, 2, \ldots, m)$ relative to the sensors fixed as anchors. Finally, the gradient method is applied to refine the locations $\boldsymbol{x}_p$ $(p = 1, 2, \ldots, m)$.

We can measure the accuracy of computed solutions when the true locations $\boldsymbol{a}_p$ $(p = 1, 2, \ldots, n)$ of all sensors are known. In the numerical experiments whose results are shown in Table 7, the values of RMSD for the computed locations of sensors $\boldsymbol{x}_p$ $(p = 1, 2, \ldots, n)$ are evaluated after applying a linear transformation (translation, reflection, orthogonal rotation, and scaling) T, provided by a Matlab program procrustes.m [2, 19]. This function minimizes the total squared errors $\sum_{p=1}^{n} \|T(\boldsymbol{x}_p) - \boldsymbol{a}_p\|^2$ between the true and transformed approximate locations of sensors. We also observe that SFSDP can accurately find the location of sensors, as shown in the values of RMSD.

As in Figures 4 and 5, SFSDP displays three figures, the locations of sensors from SDPA, after applying the linear transformation to them, and after refining them using the gradient method and applying the linear transformation to the refined locations of sensors.

Test problems			RMSD		Elapsed time		
$m,\ m_a$	ρ	σ	SDPA	w.Grad.	SDPA	Grad.	Total
$m = 3000$,	0.250	0.0	6.2e-06	9.8e-07	12.1	0.4	48.1
$m_a = 10\%$ of m		0.1	3.6e-02	9.5e-03	20.9	19.5	138.2
$= 300$		0.2	6.1e-02	1.7e-02	20.3	26.3	144.7
distributed	$(15/m)^{1/3}$	0.0	1.0e-04	2.9e-06	49.3	1.2	87.3
randomly	≈ 0.171	0.1	3.2e-02	7.0e-03	51.3	27.8	164.4
		0.2	4.9e-02	1.6e-02	51.6	25.3	162.0
$m = 3000$,	0.250	0.0	2.6e-05	1.2e-06	13.5	0.7	51.1
$m_a = 5\%$ of m		0.1	5.0e-02	1.0e-02	20.6	17.9	134.2
$=150$		0.2	7.5e-02	1.9e-02	20.4	25.6	142.0
distributed	$(15/m)^{1/3}$	0.0	1.4e-04	6.3e-06	260.4	2.4	301.0
randomly	≈ 0.171	0.1	4.8e-02	1.8e-02	190.3	40.9	312.5
		0.2	6.7e-02	2.7e-02	194.0	41.5	315.8
$m = 3000$,	0.250	0.0	1.0e-04	7.5e-06	368.4	1.2	413.0
$m_a = 8$ at corners		0.1	9.1e-02	1.4e-02	428.4	64.8	589.5
		0.2	1.4e-01	2.6e-02	422.2	45.9	563.6
	$(15/m)^{1/3}$ ≈ 0.171	Out-of-memory					
$m = 5000$,	0.250	0.0	1.4e-06	4.2e-07	17.5	0.5	112.2
$m_a = 10\%$ of m		0.1	3.2e-02	7.9e-03	41.9	30.1	337.8
$=500$		0.2	6.1e-02	1.6e-02	37.5	31.8	331.8
distributed	$(15/m)^{1/3}$	0.0	8.8e-05	2.1e-06	194.5	2.8	295.4
randomly	≈ 0.144	0.1	2.8e-02	5.8e-03	166.2	51.4	445.8
		0.2	4.2e-02	1.2e-02	170.1	54.4	452.2
$m = 5000$,	0.250	0.0	1.8e-05	7.7e-07	18.7	1.2	117.3
$m_a = 5\%$ of m		0.1	3.7e-02	8.3e-03	40.8	34.7	337.8
$= 250$		0.2	6.3e-02	1.7e-02	39.3	42.6	348.9
distributed	$(15/m)^{1/3}$						
randomly	≈ 0.144	Out-of-memory					

Table 6: Numerical results on SFSDP(5) with SDPA applied to 3-dimensional problems.

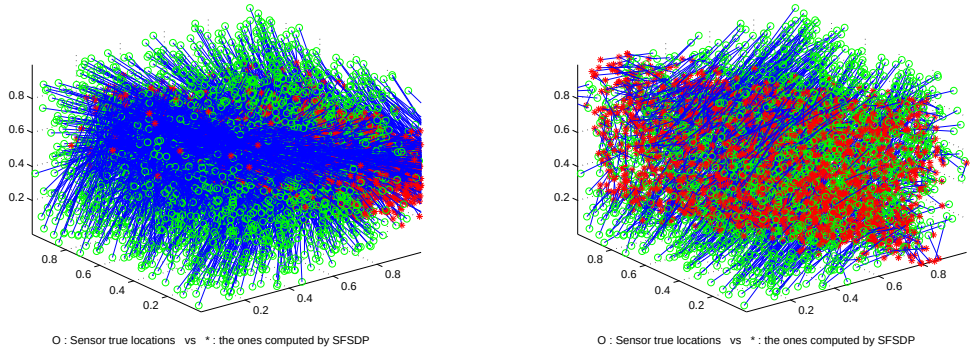

Figure 4: A 3-dimensional anchor-free problem with 3000 sensors, $\rho = 0.25$, and $\sigma = 0.1$. The locations of sensors from SFSDP($\kappa = 4$) on the left and the locations of sensors after applying the linear transformation (translation, reflection, orthogonal rotation, and scaling) which minimizes the squared errors between the true and transformed locations of sensors on the right. A circle denotes the true location of a sensor, $\star$ the computed location of a sensor, and a line segment a deviation from true and computed locations.

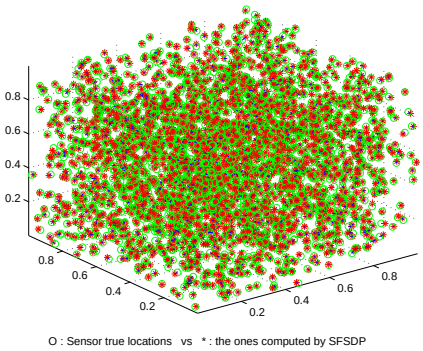

Figure 5: A 3-dimensional anchor-free problem with 3000 sensors, $\rho = 0.25$, and $\sigma = 0.1$. After applying the local refinement (a gradient method) following the linear transformation (translation, reflection, orthogonal rotation, and scaling) which minimizes the squared errors between the true and the transformed locations. A circle denotes the true location of a sensor, $\star$ the computed location of a sensor, and a line segment a deviation from true and computed locations.

Test problems			RMSD		Elapsed time		
$m,\ m_a$	ρ	σ	SDPA	w.Grad.	SDPA	Grad.	Total
$m = 3000,$ $m_a = 0$	0.250	0.0	5.0e-05	5.1e-06	441.8	0.6	490.3
		0.1	1.6e-01	9.4e-03	504.6	11.4	612.3
		0.2	1.8e-01	2.5e-02	497.0	13.4	607.4
$m = 3000,$ $m_a = 0$	$(15/m)^{1/3}$ ≈ 0.171	Out-of-memory					
$m = 5000,$ $m_a = 0$	0.250	0.0	5.2e-05	7.6e-06	723.6	0.9	852.0
		0.1	8.1e-02	9.3e-03	790.3	16.2	1060.3
		0.2	1.3e-01	1.9e-02	776.3	16.4	1046.5

Table 7: Numerical results on SFSDP(5) with SDPA applied to 3-dimensional anchor-free problems.

7 Concluding Remarks

We have described the Matlab package SFSDP. It is designed to solve larger-sized SNL problems than other available softwares. SFSDP can be used for the problems with various anchor locations and anchor-free problems in 2 or 3 dimensions.

SFSDP demonstrates the computational advantages over other methods in solving large-sized SNL problems as shown in Section 6. These come from utilizing the aggregated and correlative sparsity of the problem, which reduces the size of FSDP relaxation. One advantage of SFSDP is that it is equipped with both SeDuMi and SDPA. SFSDP incorporated with SDPA provides a solution faster than that with SeDuMi. It is our experience, however, that the accuracy of solution by SDPA is not as good as that by SeDuMi for some problems.

The SNL problem has a number of applications where computational efficiency is an important issue. SDP approach has been known to be effective in locating sensors, however, solving large-sized problems with this approach has been a challenge. We hope to improve the performance of the sparse SDP relaxation implemented in SFSDP, in particular, when the original problem does not provide enough distance information between sensors and/or no or a few anchors are available.

Acknowledgments

The authors would like to thank Professor Yinyu Ye for the original version of FSDP, Professor Kim Chuan Toh for Matlab programs, refineposition.m, procrustes.m, and helpful comments, and Mr. Zizhuo Wang for ESDP code.

References

[1] A. Y. Alfakih, A. Khandani, and H. Wolkowicz (1999) "Solving Euclidean matrix completion problem via semidefinite programming," *Comput. Opt. Appl.*, **12**, 13–30.

[2] P. Biswas, K.C. Toh, and Y. Ye (2008) "A distributed SDP approach for large scale noisy anchor-free graph realization with applications to molecular conformation," *SIAM J. Scientific Computing*, **30**, 1251–1277.

[3] P. Biswas and Y. Ye (2004) "Semidefinite programming for ad hoc wireless sensor network localization," in *Proceedings of the third international symposium on information processing in sensor networks*, ACM press, 46–54.

[4] P. Biswas and Y. Ye (2006) "A distributed method for solving semidefinite programs arising from Ad Hoc Wireless Sensor Network Localization," in *Multiscale Optimization Methods and Applications*, 69–84, Springer.

[5] P. Biswas, T.-C. Liang, T.-C. Wang, Y. Ye (2006) "Semidefinite programming based algorithms for sensor network localization," *ACM Tran. Sensor Networks*, **2**, 188–220.

[6] P. Biswas, T.-C. Liang, K.-C. Toh, T.-C. Wang, and Y. Ye (2006) "Semidefinite programming approaches for sensor network localization with noisy distance measurements," *IEEE Transactions on Automation Science and Engineering*, **3**, pp. 360–371.

[7] M. W. Carter, H. H. Jin, M. A. Saunders, and Y. Ye (2006) "SpaseLoc: an adaptive subproblem algorithm for scalable wireless sensor network localization," *SIAM J. Optim.,* **17** (4) 1102-1128.

[8] L. Doherty, K. S. J. Pister, and L. El Ghaoui (2001) "Convex position estimation in wireless sensor networks," *Proceedings of 20th INFOCOM*, **3**, 1655–1663.

[9] T. Eren, D. K. Goldenberg, W. Whiteley, Y. R. Wang, A. S. Morse, B. D. O. Anderson, and P. N. Belhumeur (2004) "Rigidity, computation, and randomization in network localization," *in Proceedings of IEEE Infocom.*

[10] K. Fujisawa, M. Fukuda, K. Kobayashi, M. Kojima, K. Nakata, M. Nakata and M. Yamashita (2008), "SDPA (SemiDefinite Programming Algorithm) User's Manual — Version 7.0.5," Research Report B-448, Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology, Oh-Okayama, Meguro, Tokyo 152-8552, Japan.

[11] M. Fukuda, M. Kojima, K. Murota and K. Nakata (2000) "Exploiting sparsity in semidefinite programming via matrix completion I: General framework," *SIAM J. Optim.*, **11**, 647–674.

[12] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S.Wicker (2002) "An empirical study of epidemic algorithms in large scale multihop wireless network," IRB-TR-02-003, Intel Corporation.

[13] A. Howard, M. Matarić and G. Sukhatme (2001) "Relaxation on a mesh: a formalism for generalized localization," In *IEEE/RSJ International conference on intelligent robots and systems*, Wailea, Hawaii, 1055–1060.

[14] S. Kim, M. Kojima and H. Waki (2009) " Exploiting sparsity in SDP relaxation for sensor network localization," *SIAM J. Optim.*, **20**, (1) 192–215.

[15] S. Kim, M. Kojima and H. Waki (2009) " User's manual for SFSDP: a Sparse Version of Full SemiDefinite Programming Relaxation for Sensor Network Localization Problems," Research Report B-449, Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology, Oh-Okayama, Meguro, Tokyo 152-8552, Japan.

[16] S. Kim, M. Kojima, M. Mevissen, and M. Yamashita (2009) " Exploiting Sparsity in Linear and Nonlinear Matrix Inequalities via Positive Semidefinite Matrix Completion," Research Report B-452, Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology, Oh-Okayama, Meguro, Tokyo 152-8552, Japan.

[17] S. Kim, M. Kojima, M. Mevissen, and M. Yamashita (2009) " User's Manual for SparseCoLO: Conversion Methods for SPARSE COnic-form Linear Optimization Problems," Research Report B-453, Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology, Oh-Okayama, Meguro, Tokyo 152-8552, Japan.

[18] K. Kobayashi, S. Kim and M. Kojima (2008) Correlative sparsity in primal-dual interior-point methods for LP, SDP and SOCP, *Appl. Math. Opt.,* **58** (1) 69–88.

[19] N.-H. Z. Leung and K.-C. Toh (2008) ,"An SDP-based divide-and-conquer algorithm for large scale noisy anchor-free graph realization," preprint, National University of Singapore.

[20] T.-C. Lian, T.-C. Wang, and Y. Ye (2004) "A gradient search method to round the semidefinite programming relaxation solution for ad hoc wireless sensor network localization," Technical report, Dept. of Management Science and Engineering, Stanford University.

[21] K. Nakata, K. Fujisawa, M. Fukuda, M. Kojima and K. Murota (2003) "Exploiting sparsity in semidefinite programming via matrix completion II: Implementation and numerical results," *Math. Program.*, **95**, 303–327.

[22] J. Nie (2009) "Sum of squares method for sensor network localization," *Comput. Opt. Appl.,* **43**, 151–179.

[23] T. K. Pong and P. Tseng (2009) "(Robust) Edge-Based Semidefinite Programming Relaxation of Sensor Network Localization," preprint.

[24] SeDuMi Homepage, http://sedumi.mcmaster.ca.

[25] SDPA Homepage, http://sdpa.indsys.chuo-u.ac.jp/sdpa/.

[26] J. F. Strum (1999) "SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones," *Optim. Methods Soft.*, **11 & 12**, 625–653.

[27] P. Tseng (2007) "Second order cone programming relaxation of sensor network localization," *SIAM J. Optim.*, **18**, 156-185.

[28] R. H. Tutuncu, K. C. Toh, and M. J. Todd (2003) "Solving semidefinite-quadratic-linear programs using SDPT3", *Math. Program.* **95**, 189–217.

[29] H. Waki, S. Kim, M. Kojima and M. Muramatsu (2006) "Sums of Squares and Semidefinite Programming Relaxations for Polynomial Optimization Problems with Structured Sparsity," *SIAM J. Optim,* **17**, 218–242.

[30] Z. Wang, S. Zheng, S. Boyd, and Y. Ye (2008) "Further relaxations of the SDP approach to sensor network localization," *SIAM J. Optim.*, **19** (2) 655–673.

[31] Y. Ye's website, http://www.stanford.edu/∼yyye.