SIAM J. OPTIM.
Vol. 20, No. 1, pp. 192–215

EXPLOITING SPARSITY IN SDP RELAXATION FOR SENSOR NETWORK LOCALIZATION*

SUNYOUNG KIM†, MASAKAZU KOJIMA‡, AND HAYATO WAKI§

Abstract. A sensor network localization problem can be formulated as a quadratic optimization problem (QOP). For QOPs, semidefinite programming (SDP) relaxation by Lasserre with relaxation order 1 for general polynomial optimization problems (POPs) is known to be equivalent to the sparse SDP relaxation by Waki et al. with relaxation order 1, except for the size and sparsity of the resulting SDP relaxation problems. We show that the sparse SDP relaxation applied to the QOP is at least as strong as the Biswas–Ye SDP relaxation for the sensor network localization problem. A sparse variant of the Biswas–Ye SDP relaxation, which is equivalent to the original Biswas–Ye SDP relaxation, is also derived. We compare numerically the sparse SDP relaxation applied to the QOP, the Biswas–Ye SDP relaxation, its sparse variant, and the edge-based SDP relaxation by Wang et al. to confirm the effectiveness of the proposed techniques for exploiting the sparsity in SDP relaxation for sensor network localization problems. The sparse SDP relaxation applied to the QOP is much faster than the Biswas–Ye SDP relaxation, and the sparse variant of the Biswas–Ye SDP relaxation outperforms all other SDP relaxations in speed.

Key words. sensor network localization problem, polynomial optimization problem, semidefinite relaxation, sparsity

AMS subject classifications. 65K05, 90C22, 90C35

DOI. 10.1137/080713380

1. Introduction.

A sensor network localization problem arises in monitoring and controlling applications using wireless sensor networks such as inventory management and gathering environmental data. Positioning sensors accurately in a wireless sensor network is an important problem for the efficiency of the applications where including GPS capability on every sensor in a network of inexpensive sensors is not an option. It is also closely related to distance geometry problems arising in predicting molecule structures and to graph rigidity.

The problem is to locate m sensors that fit the distances when a subset of distances and some sensors of known position (called anchors) are provided in a sensor network of n sensors, where $n > m$. Finding the solutions of this problem is a difficult problem. It is known to be NP-hard in general [22]. Various approaches thus have been proposed [1, 9, 10, 13, 14] for approximating the solutions.

Biswas and Ye [2] proposed a semidefinite programming (SDP) relaxation, which is called full SDP (FSDP) relaxation, for the sensor network localization problem. Many studies [3, 4, 5, 25, 30, 33] have followed in recent years. Compared with other methods for the problem, the SDP relaxation by [2] aimed to compute an accurate

*Received by the editors January 15, 2008; accepted for publication (in revised form) October 13, 2008; published electronically April 1, 2009.
http://www.siam.org/journals/siopt/20-1/71338.html

†Department of Mathematics, Ewha Women's University, 11-1 Dahyun-dong, Sudaemoon-gu, Seoul 120-750, Korea (skim@ewha.ac.kr). This author's research was supported by Kosef R01-2005-000-10271-0 and KRF 2007-313-C00089.

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

§Department of Computer Science, The University of Electro-Communications, 1-5-1 Chofugaoka, Chofu-shi, Tokyo 182-8585, Japan (hayato.waki@jsb.cs.uec.ac.jp). This author's research was partially supported by Grant-in-Aid for JSPS Fellows 18005736 and 20003236.

solution of the problem. Solving a large-scale SDP relaxation using software packages based on the primal-dual interior-point method [24, 29, 34] is, however, known to be a computational challenge. As a result, the size of the sensor network localization problem that can be handled by the SDP relaxation is limited, as mentioned in [3]. For the sensor network localization problem with a larger number of sensors, a distributed method in [3] was introduced, and a method combined with a gradient method [21] was proposed to improve the accuracy. The second-order cone programming (SOCP) relaxation was studied first in [9] and then in [30]. The solutions obtained by the SOCP relaxation are inaccurate compared to those obtained by the SDP relaxation [30]. Edge-based SDP (ESDP) and node-based SDP (NSDP) relaxations were introduced in [33] to improve the computational efficiency of the original Biswas–Ye SDP relaxation in [2]. These SDP relaxations are weaker than the original SDP relaxation in theory; however, computational results show that the quality of the solution is comparable to that of the original Biswas–Ye SDP relaxation. It is also shown that much larger-sized problems can be handled.

In the sum-of-squares (SOS) method by Nie in [25], the sensor network localization problem was formulated as minimizing a polynomial with degree 4, and the solutions were found at the global minimizer. The sparsity of the polynomial objective function with degree 4 was utilized to reduce the size of the SOS relaxation. The advantage of this approach is that it provides highly accurate solutions. Numerical results for $n = 500$ in [25] showed that accurate solutions were found if exact distance information was given.

When solving a polynomial optimization problem (POP) as in [25], one of the deciding factors of computational efficiency is the degree of the polynomials in the POP. The degree (and sparsity, if exploited) decides the size of the SDP relaxation problem generated from the POP. Whether the global minimizer of a POP can be obtained computationally depends on the solvability of the SDP relaxation problem. It is thus imperative to have polynomials of lower degree in POPs to find the global minimizer of the POPs.

For general POPs, Lasserre [19] presented a hierarchical SDP relaxation whose convergence to a global minimizer is theoretically guaranteed. More accurate solutions can be computed if increasingly larger-sized SDP relaxations, whose size is decided by a positive number called the relaxation order, are solved. The size of POPs that can be solved by Lasserre's SDP relaxation remains relatively small because the size of the SDP relaxation grows rapidly with the degree of the polynomials and the number of variables. A sparse SDP (SSDP) relaxation for POPs using the correlative sparsity of POPs was introduced to reduce the size of the SDP relaxation in [31]. We call Lasserre's relaxation the dense SDP relaxation, as opposed to the sparse SDP relaxation in [31]. Although the theoretical convergence of the sparse SDP relaxation is shown in [20] for correlatively sparse POPs, the sparse SDP relaxation is theoretically weaker than the dense SDP relaxation in general.

For quadratic optimization problems (QOPs), however, the sparse SDP relaxation with the relaxation order 1 and the dense SDP relaxation with the same relaxation order 1 are theoretically equivalent in the quality of the solutions, as mentioned in section 4.5 of [31]. Thus, the solution obtained using the sparse SDP relaxation is as accurate as that obtained by the dense SDP relaxation. Motivated by this observation, we study a QOP formulation for the sensor network localization problem. We note that the dense SDP relaxation with the relaxation order 1 is a special case of the SDP relaxation by Shor [26, 27] for general QOPs. See also the work of Fujie and Kojima in [11].

The main objective of this paper is to improve the speed of solving the sensor network localization problem by exploiting the sparsity. The QOP is solved numerically by SparsePOP [32], a MATLAB package for solving POPs using the correlative sparsity. A technique introduced in [16] is employed to improve the computational efficiency. The relationship between FSDP (the Biswas–Ye SDP relaxation [2]) and the dense SDP relaxation [19] of the proposed QOP formulation is examined. We show that the dense SDP relaxation with the relaxation order 1 and the sparse SDP relaxation with the relaxation order 1, which is called SSDP, applied to the QOP formulation of the sensor network localization problem are at least as strong as FSDP. We also derive a sparse variant of FSDP, which we call SFSDP, by exploiting its sparsity, and show that it is equivalent to FSDP except for the size and the sparsity of the resulting SDP relaxation problems.

The effectiveness of the proposed techniques of exploiting sparsity is demonstrated with numerical results on SSDP and SFSDP in comparison to the original Biswas–Ye SDP relaxation (FSDP) and the edge-based SDP relaxation (ESDP) by Wang et al. We show that (i) SSDP is faster than FSDP, (ii) SFSDP outperforms all other SDP relaxations in speed, and (iii) SFSDP attains comparable or better accuracy than all other SDP relaxations in most cases.

The sensor network localization problem is stated in detail in section 2. A QOP formulation for the sensor network localization problem with exact and noisy distance measurements is presented. In section 3, we explain the dense and the sparse SDP relaxations and describe a sparse variant of the Biswas–Ye SDP relaxation. We also show that the dense SDP relaxation is at least as strong as the Biswas–Ye SDP relaxation. Additional techniques for reducing the size of the SDP relaxation, refining solutions, and converting sparse SDP relaxation problems into standard-form SDPs are shown in section 4. Section 5 includes the comparison of numerical results from SSDP, FSDP, SFSDP, and ESDP. Section 6 contains concluding remarks and future directions.

2. Sensor network localization problems. Consider m sensors and m_a anchors, both located in the ℓ-dimensional Euclidean space $\mathbb{R}^\ell$, where ℓ is 2 or 3 in practice. Let $n = m + m_a$. The sensors are indexed with $p = 1, \ldots, m$ and the anchors with $r = m+1, \ldots, n$. We assume that the location $\boldsymbol{a}_r \in \mathbb{R}^\ell$ of anchor r is known for every $r = m+1, \ldots, n$, but the location $\boldsymbol{a}_p \in \mathbb{R}^\ell$ of sensor p is unknown for any $p = 1, \ldots, m$. We denote the exact distance $\|\boldsymbol{a}_p - \boldsymbol{a}_q\| > 0$ between sensors p and q by d_{pq} and the exact distance $\|\boldsymbol{a}_p - \boldsymbol{a}_r\| > 0$ between sensors p and r by d_{pr}. The exact values are not usually known in practice. Let $\mathcal{N}_x$ be a subset of $\{(p,q) : 1 \le p < q \le m\}$ (the set of pairs of sensors) and $\mathcal{N}_a$ be a subset of $\{(p,r) : 1 \le p \le m,\ m+1 \le r \le n\}$ (the set of pairs of sensors and anchors). Then, a sensor network localization problem is described as follows: Given distances (often containing noise) $\hat{d}_{pq} \approx d_{pq}$ between sensors p and q ($(p,q) \in \mathcal{N}_x$) and distances $\hat{d}_{pr} \approx d_{pr}$ between sensor p and anchor r ($(p,r) \in \mathcal{N}_a$), compute or estimate locations $\boldsymbol{a}_p$ of sensor p ($p = 1, \ldots, m$). We consider the problem with exact distances in section 2.1 and the problem with noisy distances in section 2.2. Both problems are reduced to QOPs. The SSDP relaxation [31] with the relaxation order 1, which is equivalent to the dense SDP relaxation [20] as mentioned in section 3.2, can be applied to the QOPs.

Naturally, we can represent a sensor network localization problem in terms of a geometrical network. Let $N = \{1, 2, \ldots, n\}$ denote the node set of sensors $p = 1, 2 \ldots, m$ and anchors $r = m+1, m+2, \ldots, n$, and $\mathcal{N}_x \cup \mathcal{N}_a$ the set of undirected

edges. We assume that all the nodes are located in the ℓ-dimensional space. To construct a geometrical network representation of the problem, we consider a graph $G(N, \mathcal{N}_x \cup \mathcal{N}_a)$ and add a positive number $\hat{d}_{pq}$ on each edge $(p,q) \in \mathcal{N}_x$ and a positive number $\hat{d}_{pr}$ on each edge $(p,r) \in \mathcal{N}_a$. Note that the node set N is partitioned into two subsets—the set of sensors $p = 1, 2, \ldots, m$ whose locations are to be approximated and the set of anchors $r = m+1, m+2, \ldots, n$ whose locations are known. Our main concern is to compute the locations of sensors with *accuracy* and *speed*. Thus, we focus on methods of extracting a small-sized and sparse subgraph $G(N, E')$ from $G(N, \mathcal{N}_x \cup \mathcal{N}_a)$ in section 4.1. We then replace the graph $G(N, \mathcal{N}_x \cup \mathcal{N}_a)$ by such a subgraph $G(N, E')$ in numerical computation. In this section, however, we formulate a sensor network localization problem with the graph $G(N, \mathcal{N}_x \cup \mathcal{N}_a)$ in a QOP.

2.1. Problems with exact distances. When all of the given distances $\hat{d}_{pq}$ $((p,q) \in \mathcal{N}_x)$ and $\hat{d}_{pr}$ $((p,r) \in \mathcal{N}_a)$ are exact, that is, $\hat{d}_{pq} = d_{pq}$ $((p,q) \in \mathcal{N}_x)$, $\hat{d}_{pr} = d_{pr}$ $((p,r) \in \mathcal{N}_a)$, the locations $\boldsymbol{x}_p = \boldsymbol{a}_p$ of sensors $p = 1, \ldots, m$ are characterized in terms of a system of nonlinear equations

$$d_{pq} = \|\boldsymbol{x}_p - \boldsymbol{x}_q\| \ (p,q) \in \mathcal{N}_x \ \text{ and } \ d_{pr} = \|\boldsymbol{x}_p - \boldsymbol{a}_r\| \ (p,r) \in \mathcal{N}_a.$$

To apply SDP relaxations, we transform this system into an equivalent system of quadratic equations

$$d_{pq}^2 = \|\boldsymbol{x}_p - \boldsymbol{x}_q\|^2 \ (p,q) \in \mathcal{N}_x \ \text{ and } \ d_{pr}^2 = \|\boldsymbol{x}_p - \boldsymbol{a}_r\|^2 \ (p,r) \in \mathcal{N}_a. \tag{1}$$

In practice, a radio range $\rho > 0$ often determines $\mathcal{N}_x$ and $\mathcal{N}_a$:

$$\left.\begin{array}{rcl} \mathcal{N}_x & = & \{(p,q) : 1 \le p < q \le m, \ \|\boldsymbol{a}_p - \boldsymbol{a}_q\| \le \rho\}, \\ \mathcal{N}_a & = & \{(p,r) : 1 \le p \le m, \ m+1 \le r \le n, \ \|\boldsymbol{a}_p - \boldsymbol{a}_r\| \le \rho\}. \end{array}\right\} \tag{2}$$

If the radio range $\rho > 0$ is sufficiently large, the sets $\mathcal{N}_x$ and $\mathcal{N}_a$ coincide with the entire sets $\{(p,q) : 1 \le p < q \le m\}$ and $\{(p,r) : 1 \le p \le m, \ m+1 \le r \le n\}$, respectively. Decreasing the radio range $\rho > 0$ reduces the size of the sets $\mathcal{N}_x$ and $\mathcal{N}_a$ monotonically. For smaller-sized $\mathcal{N}_x$ and $\mathcal{N}_a$, the system of quadratic equations (1), which is rewritten as (3) below, is more likely to satisfy a structured sparsity called the correlative sparsity in the literature [15, 31]. This sparsity can be utilized to increase the effectiveness of the sparse SDP relaxation [31] when applied to the QOPs (4) and (6).

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 above can be written as

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

Here, X_{ip} denotes the (i,p)th element of the matrix $\boldsymbol{X}$ or the ith element of $\boldsymbol{x}_p$. We call (3) *a system of sensor network localization equations*, and a matrix variable or a solution $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m) \in \mathbb{R}^{\ell \times m}$ of the system (3) *a sensor location matrix*.

By introducing an objective function that is identically zero, the QOP for the sensor network localization without noise is obtained:

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

Let L_0 denote the set of solutions $\boldsymbol{X} \in \mathbb{R}^{\ell \times m}$ of the system of sensor network localization equations (3) (or the QOP (4)).

2.2. Problems with noisy distances. When the given distances $\hat{d}_{pq} > 0$ $((p,q) \in \mathcal{N}_x)$ and $\hat{d}_{pr} > 0$ $((p,r) \in \mathcal{N}_a)$ contain noise, the system of sensor network localization equations (3) with $d_{pq} = \hat{d}_{pq}$ $((p,q) \in \mathcal{N}_x)$ and $d_{pq} = \hat{d}_{pr}$ $((p,r) \in \mathcal{N}_a)$ may not be feasible. In such a case, we can estimate the locations of sensors with a least square solution $\boldsymbol{X}$ of (3), i.e., an optimal solution of the problem

$$\text{(5)} \qquad \text{minimize} \quad \sum_{(p,q)\in\mathcal{N}_x} (\hat{d}_{pq}^2 - \|\boldsymbol{x}_p - \boldsymbol{x}_q\|^2)^2 + \sum_{(p,r)\in\mathcal{N}_a} (\hat{d}_{pr}^2 - \|\boldsymbol{x}_p - \boldsymbol{a}_r\|^2)^2.$$

Notice that this is an unconstrained POP. Nie [25] applied the SDP relaxation [19] to the POP of this form. He also proposed a sparse SDP relaxation exploiting a special structure of the POP (5) and reported some numerical results. His sparse SDP relaxation possesses a nice theoretical property that if the system of sensor network localization equations (3) with $d_{pq} = \hat{d}_{pq}$ $((p,q) \in \mathcal{N}_x)$ and $d_{pr} = \hat{d}_{pr}$ $((p,r) \in \mathcal{N}_a)$ is feasible or if the POP (5) attains the optimal value 0, then so dose its sparse SDP relaxation. As a result, the sparse SDP relaxation is exact. A disadvantage of this formulation (5) lies in the high degree of the polynomial objective function, degree 4, in the unconstrained POP (5). Note that degree 4 is twice the degree of polynomials in the system of quadratic equations (3). This increases the size of the sparse SDP relaxation of the unconstrained POP (5).

We can reformulate the POP (5) as a QOP,

$$\left.\begin{array}{ll} \text{minimize} & \displaystyle\sum_{(p,q)\in\mathcal{N}_x} \xi_{pq}^2 + \sum_{(p,r)\in\mathcal{N}_a} \xi_{pr}^2 \\ \text{subject to} & \hat{d}_{pq}^2 = \displaystyle\sum_{i=1}^{\ell} X_{ip}^2 - 2\sum_{i=1}^{\ell} X_{ip}X_{iq} + \sum_{i=1}^{\ell} X_{iq}^2 + \xi_{pq}, \quad (p,q) \in \mathcal{N}_x, \\ & \hat{d}_{pr}^2 = \displaystyle\sum_{i=1}^{\ell} X_{ip}^2 - 2\sum_{i=1}^{\ell} X_{ip}a_{ir} + \|\boldsymbol{a}_r\|^2 + \xi_{pr}, \quad (p,r) \in \mathcal{N}_a, \end{array}\right\} \qquad \text{(6)}$$

where ξ_{pq} denotes an error variable. Now, the polynomials in the problem (6) are of degree 2, a half of the degree of the objective polynomial of the POP (5), which makes the size of the resulting dense SDP relaxation [19] smaller. In addition, the SSDP relaxation [31] with the relaxation order 1 is equivalent to the dense SDP relaxation with the same order.

We may replace the objective function of the QOP (6) by

$$\sum_{(p,q)\in\mathcal{N}_x} |\xi_{pq}| + \sum_{(p,r)\in\mathcal{N}_a} |\xi_{pr}|.$$

This type of objective function is used in the papers [2, 33]. In this case, we have a QOP

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

3. SDP relaxations. We describe SDP relaxations for the QOP (4) derived from sensor network localization with exact distances. The SDP relaxation described for the QOP (4) in section 3.1 is a special case of the SDP relaxation proposed by Lasserre [19] (the dense SDP relaxation) for general POPs in the sense that the relaxation order is fixed to 1 for the QOP (4). We also mention that the dense SDP relaxation described there is essentially a classical SDP relaxation proposed by Shor [26, 27] for QOPs. Instead of just referring to [11, 19, 26, 27], we describe the dense SDP relaxation in detail to compare it with the Biswas–Ye SDP relaxation [2, 28] of the QOP (4) in section 3.2. In section 3.3, we discuss sparse variants of the dense SDP relaxation given in section 3.1 and the Biswas–Ye SDP relaxation [2]. Most of the discussions here are valid for the QOP (6) for sensor network localization with noisy distances, but the details are omitted.

The following symbols are used to describe the dense and sparse SDP relaxations. Let

$$\begin{aligned} \mathcal{I} &= \{ip : 1 \leq i \leq \ell,\ 1 \leq p \leq m\}, \\ &\quad \text{the set of subscripts of the matrix variable } \boldsymbol{X}, \\ \#\mathcal{C} &= \text{the number of elements in } \mathcal{C}\ (\mathcal{C} \subseteq \mathcal{I}). \end{aligned} \tag{7}$$

For every sensor location matrix variable $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m) \in \mathbb{R}^{\ell\times m}$ and $\mathcal{C} \subseteq \mathcal{I}$, define

$$\begin{aligned} (X_{ip} : ip \in \mathcal{C}) &= \text{the row vector variable consisting of } X_{ip}\ (ip \in \mathcal{C}), \text{ where the} \\ &\quad \text{elements are arranged according to the lexicographical} \\ &\quad \text{order of the subscripts } ip \in \mathcal{C}; \text{ for example,} \\ &\quad \text{if } \mathcal{C} = \{11, 12, 21, 22\}, \text{ then } (X_{ip} : ip \in \mathcal{C}) = (X_{11}, X_{12}, X_{21}, X_{22}), \end{aligned}$$

and

$$(X_{ip} : ip \in \mathcal{C})^T (X_{ip} : ip \in \mathcal{C}) = \sum_{ip\in\mathcal{C}} \sum_{jq\in\mathcal{C}} \boldsymbol{E}(\mathcal{C})_{ipjq} X_{ip}X_{jq}, \tag{8}$$

where $\boldsymbol{E}(\mathcal{C})_{ipjq}$ denotes the $\#\mathcal{C} \times \#\mathcal{C}$ matrix whose (ip, jq)th element is 1 and all others are 0. Specifically, we write $\boldsymbol{E}_{ipjq} = \boldsymbol{E}(\mathcal{I})_{ipjq}$ $((ip, jq) \in \mathcal{I} \times \mathcal{I})$;

$$(X_{ip} : ip \in \mathcal{I})^T (X_{ip} : ip \in \mathcal{I}) = \sum_{ip\in\mathcal{I}} \sum_{jq\in\mathcal{I}} \boldsymbol{E}_{ipjq} X_{ip}X_{jq}. \tag{9}$$

If $\mathcal{C} \subseteq \mathcal{I}$ and $ip,\ jq \in \mathcal{C}$, then each $\boldsymbol{E}(\mathcal{C})_{ipjq}$ forms a submatrix of $\boldsymbol{E}_{ipjq}$. Hence, the matrix on the right-hand side of (8) is a submatrix of that of (9). We also note that

$$\boldsymbol{E}(\mathcal{C})_{jqip} = \boldsymbol{E}(\mathcal{C})_{ipjq}^T \quad (ip,\ jq \in \mathcal{C});$$

as a result, the matrices in (8) and (9) are symmetric.

Replacing each $X_{ip}X_{jq}$ by a single variable U_{ipjq} in (9), we define an $\#\mathcal{I} \times \#\mathcal{I}$ matrix variable $\boldsymbol{U} = \sum_{ip\in\mathcal{I}}\sum_{jq\in\mathcal{I}} \boldsymbol{E}_{ipjq}U_{ipjq}$. From the identity $X_{ip}X_{jq} = X_{jq}X_{ip}$ $(ip,\ jq \in \mathcal{I})$, we impose $U_{ipjq} = U_{jqip}$, or that the matrix $\boldsymbol{U}$ is symmetric. The matrix variable $\boldsymbol{U}$ serves as a linearization of the polynomial (quadratic) matrix $(X_{ip} : ip \in \mathcal{I})^T(X_{ip} : ip \in \mathcal{I})$. We also use the notation $\boldsymbol{U}(\mathcal{C})$ for the submatrix variable of $\boldsymbol{U}$ consisting of elements U_{ipjq} $(ip,\ jq \in \mathcal{C})$: $\boldsymbol{U}(\mathcal{C}) = \sum_{ip\in\mathcal{C}}\sum_{jq\in\mathcal{C}} \boldsymbol{E}(\mathcal{C})_{ipjq}U_{ipjq}$.

3.1. The dense SDP relaxation. A polynomial (quadratic) matrix inequality

$$\boldsymbol{O} \preceq \begin{pmatrix} 1 & (X_{ip} : (ip \in \mathcal{I})) \\ (X_{ip} : (ip \in \mathcal{I}))^T & (X_{ip} : (ip \in \mathcal{I}))^T(X_{ip} : (ip \in \mathcal{I})) \end{pmatrix} \in \mathcal{S}^{\#\mathcal{I}+1} \tag{10}$$

is added to the system of sensor network localization equations (3) to derive the dense SDP relaxation. Since (10) holds for any sensor location matrix $\boldsymbol{X} \in \mathbb{R}^{\ell\times m}$, the solution set L_0 of (3) or the set of feasible solutions the QOP (4) remains the same. Using (9), we rewrite the polynomial matrix inequality (10) as

$$\boldsymbol{O} \preceq \begin{pmatrix} 1 & (X_{ip} : (ip \in \mathcal{I})) \\ (X_{ip} : (ip \in \mathcal{I}))^T & \displaystyle\sum_{ip\in\mathcal{I}}\sum_{jq\in\mathcal{I}} \boldsymbol{E}_{ipjq}X_{ip}X_{jq} \end{pmatrix} \in \mathcal{S}^{\#\mathcal{I}+1}, \tag{11}$$

where E_{ipjp} are constant symmetric matrices. Now we linearize (3) and (11) by replacing every $X_{ip}X_{jq}$ with a single variable U_{ipjq} $(ip,\ jq \in \mathcal{I})$ and obtain SDP relaxations of the system (3) and the QOP (4):

$$\left.\begin{array}{ll} d_{pq}^2 = \displaystyle\sum_{i=1}^{\ell} U_{ipip} - 2\sum_{i=1}^{\ell} U_{ipiq} + \sum_{i=1}^{\ell} U_{iqiq}, & (p,q) \in \mathcal{N}_x, \\ d_{pr}^2 = \displaystyle\sum_{i=1}^{\ell} U_{ipip} - 2\sum_{i=1}^{\ell} X_{ip}a_{ir} + \|\boldsymbol{a}_r\|^2, & (p,r) \in \mathcal{N}_a, \\ \boldsymbol{O} \preceq \begin{pmatrix} 1 & (X_{ip} : (ip \in \mathcal{I})) \\ (X_{ip} : (ip \in \mathcal{I}))^T & \boldsymbol{U} \end{pmatrix} & \end{array}\right\} \tag{12}$$

and

$$\text{minimize } 0 \text{ subject to the constraints (12).} \tag{13}$$

Let

$$L_{1d} = \left\{ \boldsymbol{X} \in \mathbb{R}^{\ell\times m} : \begin{array}{l} (\boldsymbol{X},\boldsymbol{U}) \text{ is a solution of (12)} \\ \text{for some } \boldsymbol{U} \in \boldsymbol{S}^{\ell m\times \ell m} \end{array} \right\}.$$

PROPOSITION 3.1. $L_0 \subseteq L_{1d}$.

Proof. Let $\boldsymbol{X} \in L_0$. Then, $\boldsymbol{X}$ satisfies (3) and (10). Let $\boldsymbol{U}$ be an $\#\mathcal{I} \times \#\mathcal{I}$ symmetric matrix whose components are given by $U_{ipjq} = X_{ip}X_{jq}$ $(ip,\ jq \in \mathcal{I})$. Then, $(\boldsymbol{X},\boldsymbol{U})$ satisfies (12). Therefore, $\boldsymbol{X} \in L_{1d}$.

3.2. Comparison of the dense SDP relaxation with the Biswas–Ye SDP relaxation. The Biswas–Ye SDP relaxation [2] of the system of sensor network

localization equations (3) is of the form

$$\left.\begin{array}{l} d_{pq}^2 = Y_{pp} + Y_{qq} - 2Y_{pq}, \quad (p,q) \in \mathcal{N}_x, \\ d_{pr}^2 = \|\boldsymbol{a}_r\|^2 - 2\sum_{i=1}^{\ell} X_{ip} a_{ir} + Y_{pp}, \quad (p,r) \in \mathcal{N}_a, \\ \boldsymbol{O} \preceq \begin{pmatrix} \boldsymbol{I}_\ell & \boldsymbol{X} \\ \boldsymbol{X}^T & \boldsymbol{Y} \end{pmatrix}. \end{array}\right\} \tag{14}$$

Here $\boldsymbol{I}_\ell$ denotes the $\ell \times \ell$ identity matrix and

$$\boldsymbol{Y} = \begin{pmatrix} Y_{11} & \dots & Y_{1q} & \dots & Y_{1m} \\ \vdots & & \vdots & & \vdots \\ Y_{p1} & \dots & Y_{pq} & \dots & Y_{pm} \\ \vdots & & \vdots & & \vdots \\ Y_{m1} & \dots & Y_{mq} & \dots & Y_{mm} \end{pmatrix} \in \boldsymbol{S}^m \tag{15}$$

is a matrix variable. Let

$$L_{2d} = \left\{ \boldsymbol{X} \in \mathbb{R}^{\ell \times m} : \begin{array}{c} (\boldsymbol{X}, \boldsymbol{Y}) \text{ is a solution of (14)} \\ \text{for some } \boldsymbol{Y} \in \boldsymbol{S}^m \end{array} \right\}.$$

PROPOSITION 3.2. $L_{1d} \subseteq L_{2d}$.

Proof. Suppose that $\boldsymbol{X} \in L_{1d}$. Then, there exists a $\boldsymbol{U} \in \mathcal{S}^{\#\mathcal{I}}$ such that $(\boldsymbol{X}, \boldsymbol{U})$ satisfies (12). Let $\mathcal{C}_i = \{ip : 1 \le p \le m\}$ $(1 \le i \le \ell)$. Then, $\#\mathcal{C}_i = m$. Define an $m \times m$ symmetric matrix $\boldsymbol{Y}$ by $\boldsymbol{Y} = \sum_{i=1}^{\ell} \boldsymbol{U}(\mathcal{C}_i)$ or $Y_{pq} = \sum_{i=1}^{\ell} U_{ipiq}$ $(1 \le p \le m,\ 1 \le q \le m)$. We show that $(\boldsymbol{X}, \boldsymbol{Y})$ satisfies (14); then $\boldsymbol{X} \in L_{2d}$ follows. By definition, we observe that

$$Y_{pp} = \sum_{i=1}^{\ell} U_{ipip},\ Y_{pq} = \sum_{i=1}^{\ell} U_{ipiq}, \quad \text{and} \ \ Y_{qq} = \sum_{i=1}^{\ell} U_{iqiq}$$

for every $p = 1, \dots, m$ and $q = 1, \dots, m$. Thus, the first two relations of (14) follow from the first two relations of (12), respectively. Now, we consider the matrices

$$\begin{pmatrix} 1 & (X_{ip} : (i,p) \in \mathcal{C}_i) \\ (X_{ip} : (i,p) \in \mathcal{C}_i)^T & \boldsymbol{U}(\mathcal{C}_i) \end{pmatrix} \quad (1 \le i \le m). \tag{16}$$

Note that the matrices in (16) are positive semidefinite because they are submatrices of the positive semidefinite matrix

$$\begin{pmatrix} 1 & (X_{ip} : (ip \in \mathcal{I})) \\ (X_{ip} : (ip \in \mathcal{I}))^T & \boldsymbol{U} \end{pmatrix}$$

in (12). The positive semidefiniteness of the matrices in (16) implies that

$$\boldsymbol{O} \preceq \boldsymbol{U}(\mathcal{C}_i) - (X_{ip} : ip \in \mathcal{C}_i)^T (X_{ip} : ip \in \mathcal{C}_i) \quad (1 \le i \le \ell).$$

As a result,

$$\boldsymbol{O} \preceq \sum_{i=1}^{\ell} \big(\boldsymbol{U}(\mathcal{C}_i) - (X_{ip} : ip \in \mathcal{C}_i)^T (X_{ip} : ip \in \mathcal{C}_i) \big) = \boldsymbol{Y} - \boldsymbol{X}^T \boldsymbol{X}.$$

Finally, the relation $\boldsymbol{O} \preceq \boldsymbol{Y} - \boldsymbol{X}^T \boldsymbol{X}$ is equivalent to the last relation of (14).

From the proof above, we can say that the Biswas–Ye SDP relaxation is an aggregation of the dense SDP relaxation described in section 3.1. Proposition 3.2 makes it possible to apply some of the results on the Biswas–Ye SDP relaxation in [2] to the dense SDP relaxation given in section 3.1. Among many results, it is worthy to mention that if the system of sensor network localization equations (3) is strongly uniquely localizable, then $L_0 = L_{1d} = L_{2d}$. See Definition 1 and Theorem 2 of [28].

3.3. Exploiting sparsity in the SDP relaxation problems (13) and (14). We notice that the size of the positive semidefinite constraint in (13), $1+m\ell$, is about ℓ times larger than that in (14), $m+\ell$, when the dimension ℓ is fixed and the number m of sensors is large. Therefore, solving (13) is expected to be more expensive than solving (14). In this section, we derive the sparse versions of (13) and (14), called SSDP and SFSDP, respectively, by exploiting the sparsity of the positive semidefinite constraint in (13) and (14). In section 5, numerical results of SSDP and SFSDP are shown to demonstrate how effective it is to exploit the sparsity as described below. We will observe there that not only SFSDP but also SSDP is much faster than FSDP (the original Biswas–Ye SDP relaxation (14)).

It is convenient to introduce an undirected graph $G(V, E)$ associated with the sensor network localization problem to discuss sparsity exploitation in its SDP relaxations, where $V = \{1, \ldots, m\}$ denotes the set of sensors and $E = \mathcal{N}_x$ indicates the set of undirected edges of the graph. We identify (p, q) and (q, p) to denote an edge $(p, q) \in E$. Let $G(V, \overline{E})$ be a chordal extension of $G(V, E)$ and $C_1, \ldots, C_k$ be the family of all maximal cliques of $G(V, \overline{E})$. A graph is called chordal if every (simple) cycle of length ≥ 4 has a chord (an edge joining two nonconsecutive vertices of the cycle). For the definition and basic properties of chordal graphs, we refer the reader to [6]. We note that $k \leq m$ since $G(V, \overline{E})$ is chordal [6]. We assume that $G(V, \overline{E})$ is sparse or that the size of each maximal clique of $G(V, \overline{E})$ is small. When the set $\mathcal{N}_x$ is determined by (2) for a small radio range $\rho > 0$, this assumption is expected to hold.

It should be noted that $G(V, \mathcal{N}_a)$ is obtained as a subgraph of the graph $G(N, \mathcal{N}_x \cup \mathcal{N}_a)$, which has been introduced as a geometrical representation of a sensor network localization problem, by eliminating all anchor nodes $\{m+1, m+2, \ldots, n\}$ from N and all edges in $\mathcal{N}_a$. This means that anchors are not relevant at all to the discussion in this section on exploiting sparsity in the SDP relaxation problems. However, the edges in $\mathcal{N}_a$ play a crucial role in extracting a smaller and sparser subgraph $G(V, \mathcal{N}_x \cap \widetilde{E})$ of $G(V, \mathcal{N}_x)$. We note that the method of this section is applied to the subgraph $G(V, \mathcal{N}_x \cap \widetilde{E})$ for some $\widetilde{E} \subseteq \mathcal{N}_x \cup \mathcal{N}_a$. The main purpose of extracting a subgraph is that the sensor network localization problem with the reduced graph $G(N, \widetilde{E})$ can be solved more efficiently, resulting in highly accurate approximations of the locations of sensors. This will be discussed in section 4.1.

A sparse SDP relaxation problem of the QOP (4) can be derived in two different ways. The first one is an application of the sparse SDP relaxation by Waki et al. [31] for solving a general sparse POP to the QOP (4), and the other is an application of the positive definite matrix completion based on [12, 23] to the dense SDP relaxation problem (13). The correlative sparsity (see [15]) in the QOP is a key property to construct a sparse SDP relaxation in the first approach. It corresponds to the aggregated sparsity (see [12]) in its dense SDP relaxation problem by Lasserre [19] with relaxation order 1, which is exploited using the positive definite matrix completion in the second approach. These two approaches provide equivalent sparse SDP relaxation problems of the QOP.

Here we derive sparse relaxation problems simultaneously using the second approach based on the positive definite matrix completion for the dense SDP relaxation (13) and (14). The derivation of a sparse SDP relaxation problem of the QOP (4) via the positive definite matrix completion has not been dealt with previously. It is simple to describe, and, more importantly, it is consistent with the derivation of a sparse counterpart of the Biswas–Ye SDP relaxation problem (14).

First, we rewrite the SDPs (13) and (14) in an equality standard primal SDP of the form

$$\text{minimize } \boldsymbol{A}_0 \bullet \boldsymbol{Z} \text{ subject to } \boldsymbol{A}_t \bullet \boldsymbol{Z} = b_t \ (t \in T)\ \boldsymbol{Z} \succeq \boldsymbol{O}, \tag{17}$$

where T denotes a finite index set. For (13), we define

$$\boldsymbol{Z} = \begin{pmatrix} X_{00} & (X_{ip} : (ip \in \mathcal{I})) \\ (X_{ip} : (ip \in \mathcal{I}))^T & \boldsymbol{U} \end{pmatrix}, \tag{18}$$

and for (14),

$$\boldsymbol{Z} = \begin{pmatrix} \boldsymbol{W} & \boldsymbol{X} \\ \boldsymbol{X}^T & \boldsymbol{Y} \end{pmatrix}. \tag{19}$$

After the equality standard-form SDP is obtained, the positive definite matrix completion method [12], or the conversion method using positive definite matrix completion [23], can be applied. Specifically, the application of the conversion method to (17) leads to a sparse SDP relaxation problem of (13) and a sparse variant of the Biswas–Ye SDP relaxation problem (14). It should be emphasized that the resulting SDP problems are equivalent to the dense relaxation problems (13) and (14). We explain the derivation in detail below.

Let $\mathcal{V}$ denote the index set of rows (and columns) of the matrix variable $\boldsymbol{Z}$. We assume that the rows and columns of the matrix $\boldsymbol{Z}$ in (18) are indexed by 00 and ip $(i = 1, \ldots, \ell,\ p = 1, \ldots, m)$ in lexicographical order and those of the matrix $\boldsymbol{Z}$ in (19) by $01, \ldots, 0\ell, *1, \ldots, *m$. Hence, we can write $\mathcal{V} = \mathcal{V}_0 \cup \mathcal{V}_1$, $\mathcal{V}_0 = \{00\}$, $\mathcal{V}_1 = \{ip\ (i = 1, \ldots, \ell,\ p = 1, \ldots, m)\}$ for (13) and $\mathcal{V} = \mathcal{V}_0 \cup \mathcal{V}_1$, $\mathcal{V}_0 = \{10, \ldots, \ell 0\}$, $\mathcal{V}_1 = \{*1, \ldots, *m\}$ for (14), where $*$ denotes a fixed symbol or integer larger than ℓ so that each element of $\boldsymbol{A}_t$ can be written as $[\boldsymbol{A}_t]_{ipjq}$.

As in [12], we introduce the *aggregated sparsity pattern* $\mathcal{E}$ of the data matrices

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

where $[\boldsymbol{A}_t]_{ipjq}$ denotes the (ip, jq)th element of the matrix $\boldsymbol{A}_t$. The aggregated sparsity pattern can be represented geometrically with a graph $G(\mathcal{V}, \mathcal{E})$. Note that the edge set of the graph $\{(ip, jq) \in \mathcal{E} : ip$ is lexicographically smaller than $jq\}$ has been identified as $\mathcal{E}$ itself.

Now, we construct a chordal extension $G(\mathcal{V}, \overline{\mathcal{E}})$ of $G(\mathcal{V}, \mathcal{E})$ by simulating the chordal extension from $G(V, E)$ to $G(V, \overline{E})$. For the SDP relaxation problem (14), define

$$\begin{aligned}
\overline{\mathcal{E}}_0 &= \{(i0, j0) \in \mathcal{V}_0 \times \mathcal{V}_0\ (1 \leq i < j \leq \ell)\},\\
\overline{\mathcal{E}}_1 &= \{(i0, *p) \in \mathcal{V}_0 \times \mathcal{V}_1\ (1 \leq i \leq \ell,\ 1 \leq p \leq m)\},\\
\overline{\mathcal{E}}_2 &= \{(*p, *q) \in \mathcal{V}_1 \times \mathcal{V}_1\ ((p, q) \in \overline{E})\},\\
\overline{\mathcal{E}} &= \overline{\mathcal{E}}_0 \cup \overline{\mathcal{E}}_1 \cup \overline{\mathcal{E}}_2.
\end{aligned}$$

We note that $G(\mathcal{V}_0, \overline{\mathcal{E}}_0)$ forms a complete graph and $G(\mathcal{V}_0 \cup \mathcal{V}_1, \overline{\mathcal{E}}_1)$ a complete bipartite graph. From the construction of the coefficient matrices $\boldsymbol{A}_t$ $(t \in T)$ of (14) and the extension $G(V, \overline{E})$ from $G(V, E)$, we see that $[\boldsymbol{A}_t]_{ipjq} = 0$ (i.e., $(ip, jq) \notin \mathcal{E}$) if $(ip, jq) \notin \overline{\mathcal{E}}$. Hence, $\mathcal{E} \subset \overline{\mathcal{E}}$. This implies that $G(\mathcal{V}, \overline{\mathcal{E}})$ is an extension of $G(\mathcal{V}, \mathcal{E})$. To see that $G(\mathcal{V}, \overline{\mathcal{E}})$ is chordal, we assume that $\mathcal{E}' = \{(i_0p_0, i_1p_1), (i_1p_1, i_2p_2), \ldots, (i_{r-1}p_{r-1}, i_rp_r)\} \subset \overline{\mathcal{E}}$ with $i_0p_0 = i_rp_r$ forms a cycle for some $r \geq 4$ and show that there is a chord in $\mathcal{E}'$ or an edge $(i_sp_s, i_tp_t) \in \overline{\mathcal{E}}$ with $0 \leq s < t < r$. We consider two cases. First, assume that all edges in the cycle $\mathcal{E}'$ are from $\overline{\mathcal{E}}_2$. Then $\{(p_0, p_1), (p_1, p_2), \ldots, (p_{r-1}, p_r)\}$ forms a cycle in $G(V, \overline{E})$. Since $G(V, \overline{E})$ is chordal, there is a chord $(p_s, p_t) \in \overline{E}$ with $0 \leq s < t < r$ in the cycle. Hence $(*p_s, *p_t) \in \overline{\mathcal{E}}$ is a desired chord in the cycle $\mathcal{E}'$. Now assume that at least one edge of $\mathcal{E}'$ is from $\overline{\mathcal{E}}_0 \cup \overline{\mathcal{E}}_1$, say, $(i_0p_0, i_1p_1) = (i_00, i_1p_1) \in \overline{\mathcal{E}}_0 \cup \overline{\mathcal{E}}_1$. Then $i_0p_0 = i_00$ is adjacent to every node of $\mathcal{E}'$. Hence (i_0p_0, i_1p_2) serves as a chord. Thus we have shown that $G(\mathcal{V}, \overline{\mathcal{E}})$ is a chordal extension of $G(\mathcal{V}, \mathcal{E})$. Define

$$\overline{\mathcal{C}}_h = \{10, \ldots, \ell 0, *p \ (p \in C_h)\} \ (1 \leq h \leq k).$$

Then $\overline{\mathcal{C}}_1, \overline{\mathcal{C}}_2, \ldots, \overline{\mathcal{C}}_k$ are the maximal cliques of $G(\mathcal{V}, \overline{\mathcal{E}})$. In fact, if $\overline{\mathcal{C}} = \{i_10, \ldots, i_s0, *p_1, \ldots, *p_t\}$ is a clique of $G(\mathcal{V}, \overline{\mathcal{E}})$, then $\{p_1, \ldots, p_t\}$ is a clique of $G(V, \overline{E})$, which is contained in a maximal clique C_h; hence $\overline{\mathcal{C}}$ is contained in $\overline{\mathcal{C}}_h$.

For (13), define

$$\begin{aligned}
\overline{\mathcal{E}}_1 &= \{(00, ip) \in \mathcal{V}_0 \times \mathcal{V}_1 \ (1 \leq i \leq \ell, \ 1 \leq p \leq m)\},\\
\overline{\mathcal{E}}_2 &= \{(ip, jq) \in \mathcal{V}_1 \times \mathcal{V}_1 \ ((p, q) \in \overline{E}, \ 1 \leq i \leq \ell, \ 1 \leq j \leq \ell)\},\\
\overline{\mathcal{E}}_3 &= \{(ip, jp) \in \mathcal{V}_1 \times \mathcal{V}_1 \ (1 \leq p \leq m, \ 1 \leq i < j \leq \ell)\},\\
\overline{\mathcal{E}} &= \overline{\mathcal{E}}_1 \cup \overline{\mathcal{E}}_2 \cup \overline{\mathcal{E}}_3,\\
\overline{\mathcal{C}}_h &= \{00, \ 1p, \ldots, \ell p \ (p \in C_h)\} \ (1 \leq h \leq k).
\end{aligned}$$

Similarly to the case of (14), we can prove that $G(\mathcal{V}, \overline{\mathcal{E}})$ forms a chordal extension of $G(\mathcal{V}, \mathcal{E})$ and that $\overline{\mathcal{C}}_1, \ldots, \overline{\mathcal{C}}_k$ are its maximal cliques. The argument there may be slightly complicated, but the basic idea of the proof is essentially the same.

If we apply the conversion method [23] to (17) using the information on the chordal graph $G(\mathcal{V}, \overline{\mathcal{E}})$ and its maximal cliques $\overline{\mathcal{C}}_1, \ldots, \overline{\mathcal{C}}_k$, then we obtain an SDP problem

$$(20) \qquad \left.\begin{array}{ll} \text{minimize} & \boldsymbol{A}_0 \bullet \boldsymbol{Z} \\ \text{subject to} & \boldsymbol{A}_t \bullet \boldsymbol{Z} = b_t \ (t \in T), \ \boldsymbol{Z}_{\overline{\mathcal{C}}_h, \overline{\mathcal{C}}_h} \succeq \boldsymbol{O} \ (1 \leq h \leq k). \end{array}\right\}$$

Here $\boldsymbol{Z}_{\overline{\mathcal{C}}_h, \overline{\mathcal{C}}_h}$ denotes a submatrix of $\boldsymbol{Z}$ consisting of the elements $\boldsymbol{Z}_{ipjq}$ $(ip \in \overline{\mathcal{C}}_h, \ jq \in \overline{\mathcal{C}}_h)$. If the size of every $\overline{\mathcal{C}}_h$ is small, we can solve the SDP (20) more efficiently than the original SDP (17). In the SDP problem (20), two distinct maximal cliques $\overline{\mathcal{C}}_{h_1}$ and $\overline{\mathcal{C}}_{h_2}$ may intersect each other. Hence, it is not a standard SDP to which the primal-dual interior-point method can be applied. In section 4.4, we discuss how we transform it into a standard equality-form SDP.

For (13), the SDP (20) is rewritten as

$$\left.\begin{array}{ll} \text{minimize} & 0 \\ \text{subject to} & d_{pq}^2 = \displaystyle\sum_{i=1}^{\ell} U_{ipip} - 2\sum_{i=1}^{\ell} U_{ipiq} + \sum_{i=1}^{\ell} U_{iqiq}, \quad (p,q) \in \mathcal{N}_x, \\ & d_{pr}^2 = \displaystyle\sum_{i=1}^{\ell} U_{ipip} - 2\sum_{i=1}^{\ell} X_{ip}a_{ir} + \|\boldsymbol{a}_r\|^2, \quad (p,r) \in \mathcal{N}_a, \\ & \boldsymbol{O} \preceq \begin{pmatrix} 1 & (X_{ip} : (ip \in \widetilde{\mathcal{C}}_h)) \\ (X_{ip} : (ip \in \widetilde{\mathcal{C}}_h))^T & \boldsymbol{U}(\widetilde{\mathcal{C}}_h) \end{pmatrix} \quad (1 \le h \le k), \end{array}\right\} \tag{21}$$

where $\widetilde{\mathcal{C}}_h = \overline{\mathcal{C}}_h \backslash \{00\} = \{1p, \ldots, \ell p \ (p \in C_h)\}$. For (14), the SDP (20) is rewritten as

$$\left.\begin{array}{ll} \text{minimize} & 0 \\ \text{subject to} & d_{pq}^2 = Y_{pp} + Y_{qq} - 2Y_{pq}, \quad (p,q) \in \mathcal{N}_x, \\ & d_{pr}^2 = \|\boldsymbol{a}_r\|^2 - 2\displaystyle\sum_{i=1}^{\ell} X_{ip}a_{ir} + Y_{pp}, \ (p,r) \in \mathcal{N}_a, \\ & \boldsymbol{O} \preceq \begin{pmatrix} \boldsymbol{I}_\ell & (\boldsymbol{x}_p : p \in C_h) \\ (\boldsymbol{x}_p : p \in C_h)^T & \boldsymbol{Y}_{C_h,C_h} \end{pmatrix} \quad (1 \le h \le k), \end{array}\right\} \tag{22}$$

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

$$L_{1s} = \left\{ \boldsymbol{X} \in \mathbb{R}^{\ell \times m} : \begin{array}{c} (\boldsymbol{X}, \boldsymbol{U}) \ \text{ is a solution of (21)} \\ \text{for some } \boldsymbol{U} \in \mathcal{S}^{\ell m} \end{array} \right\},$$

$$L_{2s} = \left\{ \boldsymbol{X} \in \mathbb{R}^{\ell \times m} : \begin{array}{c} (\boldsymbol{X}, \boldsymbol{Y}) \ \text{ is a solution of (22)} \\ \text{for some } \boldsymbol{Y} \in \mathcal{S}^{m} \end{array} \right\}.$$

PROPOSITION 3.3. $L_{1s} = L_{1d} \subseteq L_{2s} = L_{2d}$.

Proof. By Proposition 3.2, we know that $L_{1d} \subseteq L_{2d}$. The equivalence of the SDPs (17) and (20) is established in [12]. Hence $L_{1s} = L_{1d}$ and $L_{2s} = L_{2d}$ follow.

We briefly mention a generalization of the sparse variant (22) of the Biswas–Ye SDP relaxation, which includes the node-based and edge-based SDP relaxations of sensor network localization problems proposed in [33]. They are regarded as further relaxations of the Biswas–Ye SDP relaxation. We call them NSDP and the ESDP, respectively, as in [33]. Our sparse SDP relaxation described above is compared with the ESDP with numerical results in section 5.

Let Γ be a family of nonempty subsets of the set $\{1, \ldots, m\}$ of sensors. Then, a relaxation of the Biswas–Ye SDP relaxation (14) is

$$\left.\begin{array}{ll} \text{minimize} & 0 \\ \text{subject to} & d_{pq}^2 = Y_{pp} + Y_{qq} - 2Y_{pq}, \quad (p,q) \in \mathcal{N}_x, \\ & d_{pr}^2 = \|\boldsymbol{a}_r\|^2 - 2\displaystyle\sum_{i=1}^{\ell} X_{ip}a_{ir} + Y_{pp}, \ (p,r) \in \mathcal{N}_a, \\ & \boldsymbol{O} \preceq \begin{pmatrix} \boldsymbol{I}_\ell & \boldsymbol{X}(C) \\ \boldsymbol{X}(C)^T & \boldsymbol{Y}(C) \end{pmatrix} \quad (C \in \Gamma). \end{array}\right\} \tag{23}$$

Obviously, if we take the family of maximum cliques $C_1, \ldots, C_k$ of the chordal extension $G(V, \overline{E})$ of the aggregated sparsity pattern graph $G(V, \overline{E})$ for Γ, the SDP (23)

coincides with the SDP (22). On the other hand, if $\Gamma = \{\{q \in V : (p,q) \in \mathcal{N}_x\} : p \in V\}$ in the SDP (23), it becomes the NSDP relaxation, and if $\Gamma = \{\{p,q\} : (p,q) \in \mathcal{N}_x\}$, the ESDP relaxation is obtained. In the case of the ESDP relaxation, each member of the family Γ consists of two elements, and the matrix in the positive semidefinite condition of (23) is $(\ell+2) \times (\ell+2)$. Let L_n and L_e denote the solution sets of the NSDP and ESDP relaxations, respectively. By construction, we know that $L_{2d} \subseteq L_n \subseteq L_e$. It was shown in [33] that if the underlying graph $G(V,\mathcal{E})$ is chordal, then $L_{2d} = L_n$. In this case, we know that $G(V,\mathcal{E}) = G(V,\overline{\mathcal{E}})$ and that $L_{2d} = L_{2s} = L_n$ also follows from Proposition 3.3.

4. Additional techniques. We present two methods for reducing the size of the sparse SDP relaxation problem (21) to increase computational efficiency. We note that in the method described in section 4.1, the size of the system of sensor network localization equations (3) is reduced before applying the sparse SDP relaxation. The method in section 4.2 is to decrease the size of the SDP relaxation problem by eliminating free variables [16, 18]. In addition, we address the issues of refining solutions of the SDP relaxation using a nonlinear least square method and how we transform the SDP (20) into a standard form SDP in sections 4.3 and 4.4, respectively.

4.1. Reducing the size of the system of sensor network localization equations (3). Recall that $G(N, \mathcal{N}_x \cup \mathcal{N}_a)$ denotes a graph associated with the system of sensor network localization equations (3), where $N = \{1, \ldots, n\}$ denotes the node set consisting of all sensors and anchors. Consider subgraphs $G(N,E')$ of $G(N,\mathcal{N}_x \cup \mathcal{N}_a)$ with the same node set N and an edge subset E' of $\mathcal{N}_x \cup \mathcal{N}_a$. Let $\deg(p,E')$ denote the degree of a node $p \in N$ in a subgraph $G(N,E')$, i.e., the number of edges incident to a node $p \in N$. In the ℓ-dimensional sensor network localization problem, $\deg(p,E) \geq \ell+1$ for every sensor node p is necessary (but not sufficient) to determine their locations when they are located in generic positions. Therefore, we consider the family $\mathcal{G}_\kappa$ of subgraphs $G(N,E')$ of $G(N,\mathcal{N}_x \cup \mathcal{N}_a)$ such that $\deg(p,E')$ is not less than $\min\{\deg(p,\mathcal{N}_x \cup \mathcal{N}_a), \kappa\}$ for every sensor node p, where κ is a positive integer not less than $\ell+1$. We choose a minimal subgraph $G(N,\widetilde{E})$ from the family $\mathcal{G}_\kappa$ and replace $\mathcal{N}_x$ and $\mathcal{N}_a$ by $\mathcal{N}_x \cap \widetilde{E}$ and $\mathcal{N}_a \cap \widetilde{E}$, respectively, in the system of sensor network localization equations (3). Since this method of reducing the size of (3) is based on heuristics, it may weaken the quality of the relaxation as shown in Table 4 in section 5.

When choosing edges from $\mathcal{N}_x$ and $\mathcal{N}_a$ for a reduced edge set $\widetilde{E}$, we give priority to the edges in $\mathcal{N}_a$ over those in $\mathcal{N}_x$. More precisely, for every sensor $p = 1, 2, \ldots, m$, we first choose at most $\ell+1$ edges, $(p,r) \in \mathcal{N}_a$, and then choose edges from $\mathcal{N}_x$ to produce a minimal subgraph $G(N,\widetilde{E})$ satisfying the desired property. As $\mathcal{N}_a$ involves more edges, the resulting subgraph $G(V,\mathcal{N}_x \cap \widetilde{E})$ becomes sparser and small. Thus, the sparse SDP relaxation in section 3.3 for the reduced problem with the graph $G(V,\mathcal{N}_x \cap \widetilde{E})$ can be solved more efficiently. This will be confirmed through numerical results in section 5.

With an increasingly larger value for κ, we can expect to obtain more accurate locations of the sensors, though it takes longer to solve the sparse SDP relaxation. In the numerical experiments in section 5, we took $\kappa = \ell+2$.

A different way of reducing the size of the system of sensor network localization equations (3) was proposed by Wang et al. [33]. They restricted the degree of each sensor node to a small positive integer λ. Let $\mathcal{G}^\lambda$ denote the family of subgraphs $G(N,E')$ of $G(N,E)$ such that $\deg(p,E') \leq \lambda$ for every sensor p. It was suggested

to take $5 \leq \lambda \leq 10$ in their SDP relaxation method called ESDP proposed for the 2-dimensional sensor network localization problem in [33].

We tested these two methods, the one choosing a minimal subgraph from the family $\mathcal{G}_\kappa$, and the other choosing a maximal subgraph from the family $\mathcal{G}^\lambda$, and found that the former method is more effective when it is combined with SSDP (the sparse SDP relaxation (21) described in section 3.3), FSDP (the Biswas–Ye SDP relaxation (14) described in section 3.2), and SFSDP (the sparse variant (22) of FSDP described in section 3.3).

4.2. Reducing the size of the SDP relaxation problem (21). We rewrite the sparse SDP relaxation problem (21) as a dual standard-form SDP with equalities

$$\left.\begin{array}{ll} \text{maximize} & \displaystyle\sum_{i=1}^{k} b_i y_i \\ \text{subject to} & \boldsymbol{a}_0 - \displaystyle\sum_{i=1}^{k} \boldsymbol{a}_i y_i = \boldsymbol{0},\ \boldsymbol{A}_0 - \sum_{i=1}^{k} \boldsymbol{A}_i y_i \succeq \boldsymbol{O} \end{array}\right\} \tag{24}$$

for some vectors $\boldsymbol{a}_i$ $(i = 0, 1, \ldots, k)$, symmetric matrices $\boldsymbol{A}_i$ $(i = 0, 1, \ldots, k)$, and real numbers $b_i = 0$ $(i = 1, \ldots, k)$. The corresponding primal SDP is of the form

$$\left.\begin{array}{ll} \text{minimize} & \boldsymbol{a}_0^T \boldsymbol{z} + \boldsymbol{A}_0 \bullet \boldsymbol{Z} \\ \text{subject to} & \boldsymbol{a}_i^T \boldsymbol{z} + \boldsymbol{A}_i \bullet \boldsymbol{Z} = b_i\ (i = 1, \ldots, k),\ \boldsymbol{Z} \succeq \boldsymbol{O}, \end{array}\right\} \tag{25}$$

which contains free vector variable $\boldsymbol{z}$. In [16], Kobayashi, Nakata, and Kojima proposed a method to reduce the size of the primal-dual pair SDPs of this form by eliminating the free variable vector $\boldsymbol{z}$ from the primal SDP (25) (and the equality constraint $\boldsymbol{a}_0 - \sum_{i=1}^{k} \boldsymbol{a}_i y_i = \boldsymbol{0}$ from the dual (24)). We used an improved version [18] of their method before applying SeDuMi [29] to solve primal-dual pair SDPs (25) and (24) in the numerical experiments in section 5. The method worked effectively to reduce the computational time for solving the SDPs (24) and (25).

4.3. Refining solutions by a nonlinear least square method. A nonlinear optimization method can be used to refine the solution obtained by an SDP relaxation of the sensor network localization problem as suggested in [5], where the gradient method is used. In the numerical experiments in section 5, the MATLAB function "lsqnonlin," an implementation of the trust-region reflective Newton method [7, 8] for nonlinear least square problems with bound constraints, is used.

4.4. Conversion of the SDP (20) into a standard-form SDP. Two ways of converting the SDP (20) into a standard-form SDP to which the primal-dual interior-point method can be applied exist: a conversion of the SDP (20) into an equality standard form SDP, which was presented in [23], and a conversion into a linear matrix inequality standard-form SDP having equality constraints, which was used in SparsePOP [32]. We implemented the second method in SFSDP (the sparse variant of the Biswas–Ye SDP relaxation described in section 3.3), and a brief description is as follows.

For simplicity of discussion, we assume that the size of data matrices $\boldsymbol{A}_t$ $(t \in T)$ and the matrix variable $\boldsymbol{Z}$ is n and each $\bar{\mathcal{C}}_h$ is a subset of $\{1, 2, \ldots, n\}$. We introduce matrices $\boldsymbol{F}^{ij}$ $(1 \leq i \leq j \leq n)$, which form a basis of $\mathcal{S}^n$. Here $\boldsymbol{F}^{ij}$ denotes an $n \times n$ symmetric matrix such that the elements in the (i, j)th and (j, i)th position are 1 and

all the other elements are 0. Then, we can represent the matrix variable $\boldsymbol{Z}$ as

$$\boldsymbol{Z} = \sum_{1 \le i \le j \le n} \boldsymbol{F}^{ij} z_{ij},\ z_{ij} \in \mathbb{R}\ \ (1 \le i \le j \le n).$$

We also use the notation $\boldsymbol{F}^{ij}(\mathcal{C})$ for a submatrix of $\boldsymbol{F}^{ij}$ consisting of the elements $\boldsymbol{F}^{ij}_{pq}$ $((p,q) \in \mathcal{C} \times \mathcal{C})$, where $\mathcal{C}$ is a subset of $\{1, 2, \ldots, n\}$. Let

$$\begin{aligned} I_h &= \{(i,j) \in \bar{\mathcal{C}}_h \times \bar{\mathcal{C}}_h : 1 \le i \le j \le n\}\ (1 \le h \le k), \\ \bar{I} &= \{(i,j) : [\boldsymbol{A}_t]_{ij} \ne 0 \text{ for some } t = 0\ \text{ or } t \in T,\ 1 \le i \le j \le n\}. \end{aligned}$$

Then, we can rewrite the SDP (20) as

$$\left.\begin{array}{ll} \text{minimize} & \displaystyle\sum_{(i,j) \in \bar{I}} \left(\boldsymbol{A}_0 \bullet \boldsymbol{F}^{ij}\right) z_{ij} \\ \text{subject to} & \displaystyle\sum_{(i,j) \in \bar{I}} \left(\boldsymbol{A}_t \bullet \boldsymbol{F}^{ij}\right) z_{ij} = b_t\ (t \in T), \\ & \displaystyle\sum_{(i,j) \in I_h} \boldsymbol{F}^{ij}(\overline{\mathcal{C}}_h) z_{ij} \succeq \boldsymbol{O}\ (1 \le h \le k), \end{array}\right\} \tag{26}$$

or

$$\left.\begin{array}{ll} \text{minimize} & \displaystyle\sum_{(i,j) \in \bar{I}} \left(\boldsymbol{A}_0 \bullet \boldsymbol{F}^{ij}\right) z_{ij} \\ \text{subject to} & -b_t + \displaystyle\sum_{(i,j) \in \bar{I}} \left(\boldsymbol{A}_t \bullet \boldsymbol{F}^{ij}\right) z_{ij} \ge 0\ (t \in T), \\ & b_t - \displaystyle\sum_{(i,j) \in \bar{I}} \left(\boldsymbol{A}_t \bullet \boldsymbol{F}^{ij}\right) z_{ij} \ge 0\ (t \in T), \\ & \displaystyle\sum_{(i,j) \in I_h} \boldsymbol{F}^{ij}(\overline{\mathcal{C}}_h) z_{ij} \succeq \boldsymbol{O}\ (1 \le h \le k). \end{array}\right\} \tag{27}$$

The resulting SDP becomes a standard linear matrix standard-form SDP, which is often called a dual standard-form SDP. Thus, we can apply the primal-dual interior-point method to the SDP (27). SeDuMi, the software used for solving SDP problems in the numerical experiment in section 5, can handle an SDP of the form (26) without converting it into the form of (27).

In the context of section 3.3, each $\bar{\mathcal{C}}_h$ corresponds to a maximal clique of a chordal graph $(1 \le h \le k)$ and $\bar{I} \subseteq \bigcup_{h=1}^{k} I_h$. If the chordal graph is sparse, the SDP (27) satisfies a structured sparsity, called the correlative sparsity in [15], which makes the application of the primal-dual interior-point method to the SDP (27) very efficient. See [15, section 4].

Remark 4.1. We also tested the other method presented in [23] to convert the SDP (20) into an equality standard-form SDP. But it did not work as effectively as the conversion of the SDP (20) into a linear matrix inequality standard-form SDP having equality constraints described above. See [17] for more details and numerical comparison between these two methods applied to the SDP (22).

5. Numerical results. We compare the following methods in the numerical experiments:

- *FSDP*. The Biswas–Ye SDP relaxation [3], described in (14) in section 3.2.

- *ESDP*. The edge-based SDP relaxation [33], equivalent to (23) with $\Gamma = \{\{p,q\} : (p,q) \in \mathcal{N}_x\}$.
- *SSDP*. The sparse SDP relaxation (21) of the QOP (4) for sensor network localization problems with exact distances, described in section 3.3. SSDP is implemented by applying the sparse SDP relaxation by Waki et al. [32] for a general sparse polynomial optimization problem to the QOP (4). More precisely, SDP relaxation problems are constructed by applying SparsePOP [31] to the QOPs (4) and solved by SeDuMi [29] in SparsePOP.
- *SFSDP*. The sparse variant of FSDP described in sections 3.3 and 4.4. See (22).

Among ESDP, FSDP, the POP method in [25], and the SOCP relaxation in [30], ESDP was shown to be the most efficient, as shown in [33]. MATLAB codes for ESDP and FSDP can be downloaded from the website [35]. The code for ESDP provides the solutions after refining the solution using the gradient method. The solutions before refining are provided by the code for FSDP.

For refining the obtained solutions from FSDP, SSDP, and SFSDP, we use a nonlinear least square method provided by the MATLAB function lsqnonlin, an implementation of the trust-region Newton method [7, 8] for nonlinear least square problems with bound constraints. As mentioned in section 4.2, an improved version [18] of the method in [16] for handling equality constraints to reduce the size of the SDP relaxation problem is employed in SSDP before applying SeDuMi.

Numerical test problems are generated as follows: m sensors $\boldsymbol{a}_p$ $(p = 1, 2, \ldots, m)$ are distributed randomly in the 2-dimensional unit square $[0,1] \times [0,1]$ or the 3-dimensional unit cube $[0,1]^3$, where $m = 500,\ 1000,\ 2000$, and 4000 are used in the 2-dimensional problems and $m = 1000$ in the 3-dimensional problems. Anchors are placed as follows:

bd3: 3 anchors on the boundary points $(0,0), (0.5,0), (0,0.5)$,
corner4: 4 anchors on the corners (a_1, a_2), $a_i \in \{0,1\}$,
5×5: 25 anchors on the grid (a_1, a_2), $a_i \in \{0, 1/4, 2/4, 3/4, 1\}$,
rand50: 50 randomly placed anchors in $[0,1] \times [0,1]$,
rand100: 100 randomly placed anchors in $[0,1] \times [0,1]$

for 2-dimensional problems, and

corner8: 8 anchors on the corners (a_1, a_2, a_3), $a_i \in \{0,1\}$,
$3 \times 3 \times 3$: 27 anchors on the grid (a_1, a_2, a_3), $a_i \in \{0, 1/2, 1\}$,
rand25: 25 randomly placed anchors

for the 3-dimensional problems. A radio range ρ chosen from $\{0.06,\ 0.08,\ 0.1,\ 0.2,\ 0.3\}$ for the 2-dimensional problems and from $\{0.3,\ 0.4,\ 0.5\}$ for the 3-dimensional problems determines the sets $\mathcal{N}_x$ and $\mathcal{N}_a$ by (2). The exact distances

$$d_{pq} = \|\boldsymbol{a}_p - \boldsymbol{a}_q\| \ ((p,q) \in \mathcal{N}_x) \text{ and } d_{pr} = \|\boldsymbol{a}_p - \boldsymbol{a}_r\| \ ((p,r) \in \mathcal{N}_a)$$

are computed. For numerical problems with noisy distances, we further perturb the distances as

$$\hat{d}_{pq} = (1 + \sigma\epsilon_{pq}) d_{pq} \ ((p,q) \in \mathcal{N}_x) \text{ and } \hat{d}_{pr} = (1 + \sigma\epsilon_{pr}) d_{pr} \ ((p,r) \in \mathcal{N}_a). \tag{28}$$

Here σ denotes a nonnegative constant, and ϵ_{pq} and ϵ_{pr} are chosen from the standard normal distribution $N(0,1)$. We call σ a noisy factor and take $\sigma = 0.1$ for noisy problems in sections 5.2, 5.3, and 5.4.

Throughout section 5, ρ denotes radio range, λ an upper bound for the degree of any sensor node described in section 4.1 for ESDP, κ a lower bound for the degree of any sensor node described in section 4.1 for FSDP, SSDP, and SFSDP, "cpu" average cpu time over 5 problems in seconds consumed by SeDuMi with the accuracy parameter pars.eps = 1.0e-5. As in [2, 3, 4, 30, 33], the root mean square distance (rmsd)

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

is used to measure the accuracy of the locations of sensor $p = 1, 2, \ldots, m$ computed by SeDuMi and to measure the accuracy of their refinement by the gradient method in ESDP or the MATLAB function lsqnonlin in FSDP, SSDP, and SFSDP. The values of rmsd after the refinement are included in the parentheses. We note that ESDP provides rmsd only after refining the locations of sensors. The values is of rmsd in all tables denote average rmsd over 5 problems. Numerical experiments were performed on PowerPC 1.83GHz with 2GB memory.

TABLE 1

Numerical results on 2*-dimensional problems with randomly generated* 500 *sensors in* $[0,1] \times [0,1]$ *and exact distances* ($\sigma = 0.0$).

SDP(λ|κ)	Anchor location	ρ=0.1 rmsd	ρ=0.1 cpu	ρ=0.2 rmsd	ρ=0.2 cpu	ρ=0.3 rmsd	ρ=0.3 cpu
ESDP(5)	bd3	(6.3e-01)	21.9	(5.4e-01)	74.7	(3.4e-01)	64.5
ESDP(10)	bd3	(6.5e-01)	47.6	(4.7e-01)	262.0	(2.2e-01)	169.3
FSDP(4)	bd3	2.9e-01 (4.2e-02)	461.6	1.2e-04 (3.1e-06)	399.7	(1.1e-07)	548.9
SSDP(4)	bd3	7.2e-01 (6.3e-01)	172.6	5.8e-02 (7.2e-07)	60.3	(5.8e-07)	34.1
SFSDP(4)	bd3	4.9e-01 (4.7e-01)	21.6	1.0e-02 (3.4e-08)	15.1	(7.8e-09)	12.5
ESDP(5)	corner4	(8.7e-02)	25.9	(1.9e-02)	42.4	(1.1e-02)	77.0
ESDP(10)	corner4	(2.5e-02)	82.2	(4.0e-06)	237.5	(1.9e-07)	106.6
FSDP(4)	corner4	1.7e-02 (6.7e-03)	515.8	1.0e-04 (1.9e-07)	474.2	(1.1e-08)	399.5
SSDP(4)	corner4	2.5e-02 (6.7e-03)	243.8	1.1e-04 (1.1e-07)	49.2	(1.7e-08)	26.8
SFSDP(4)	corner4	5.4e-02 (4.7e-02)	21.5	1.4e-04 (3.8e-08)	11.7	(1.5e-09)	8.9
ESDP(5)	5×5	(4.0e-03)	60.1	(7.6e-05)	38.0	(4.0e-08)	37.1
ESDP(10)	5×5	(2.5e-06)	68.3	(6.0e-08)	93.9	(1.2e-08)	133.4
FSDP(4)	5×5	5.1e-04 (1.4e-07)	340.6	2.7e-05 (6.9e-09)	317.2	(4.2e-08)	367.4
SSDP(4)	5×5	4.0e-04 (7.6e-08)	83.0	5.8e-05 (4.2e-08)	7.6	(6.5e-08)	5.5
SFSDP(4)	5×5	1.0e-03 (2.5e-04)	13.2	1.7e-05 (7.2e-12)	6.2	(2.9e-12)	5.8
ESDP(5)	rand50	(3.1e-02)	68.4	(1.3e-04)	77.8	(6.0e-07)	50.7
ESDP(10)	rand50	(2.0e-05)	144.1	(2.1e-08)	104.9	(2.2e-08)	196.1
FSDP(4)	rand50	5.6e-03 (7.6e-08)	352.0	2.9e-06 (1.6e-07)	386.9	(9.6e-09)	410.2
SSDP(4)	rand50	1.2e-03 (3.6e-07)	27.9	1.4e-06 (7.2e-08)	6.9	(4.9e-14)	5.4
SFSDP(4)	rand50	1.9e-02 (1.4e-02)	10.1	1.0e-04 (1.9e-10)	6.5	(8.5e-10)	5.9

5.1. Problems with exact distances. Table 1 shows numerical results on the problems with 500 sensors randomly generated in $[0,1] \times [0,1]$ and exact distances (or $\sigma = 0$). If we compare the values of rmsd, we see that

(a) FSDP(4), SSDP(4), and SFSDP(4) attain a similar quality of rmsd in almost all test problems; however, cpu time consumed by FSDP(4) to solve SDP

TABLE 2
Numerical results on 2-dimensional problems with randomly generated 1000 sensors in $[0,1] \times [0,1]$ *and and exact distances* ($\sigma = 0.0$).

SDP(λ|κ)	Anchor location	ρ=0.1 rmsd	ρ=0.1 cpu	ρ=0.2 rmsd	ρ=0.2 cpu	ρ=0.3 rmsd	ρ=0.3 cpu
ESDP(5)	bd3	(6.3e-01)	78.9	(5.4e-01)	185.9	(3.5e-01)	142
ESDP(10)	bd3	(6.3e-01)	283.1	(4.4e-01)	560.4	(2.0e-01)	370.0
SSDP(4)	bd3	4.7e-01 (4.4e-01)	557.9	3.0e-02 (7.3e-07)	107.0	(3.2e-08)	84.2
SFSDP(4)	bd3	1.3e-01 (2.6e-03)	55.3	2.9e-03 (5.7e-09)	27.3	(2.7e-09)	27.6
ESDP(5)	corner4	(1.7e-02)	90.9	(6.6e-03)	153.6	(9.8e-04)	167.7
ESDP(10)	corner4	(3.1e-03)	351.8	(5.7e-06)	593.8	(6.3e-07)	278.7
SSDP(4)	corner4	1.1e-03 (1.2e-06)	892.7	1.1e-04 (3.5e-07)	101.3	(2.8e-08)	55.2
SFSDP(4)	corner4	1.9e-03 (2.2e-08)	50.2	1.5e-04 (6.3e-09)	20.5	(5.4e-09)	18.0
ESDP(5)	5×5	(3.8e-03)	148.1	(9.8e-08)	94.1	(3.2e-08)	81.3
ESDP(10)	5×5	(2.0e-06)	283.5	(3.8e-08)	428.1	(9.2e-09)	466.4
SSDP(4)	5×5	6.3e-04 (1.4e-07)	239.7	3.2e-05 (3.0e-08)	19.3	(5.7e-09)	14.2
SFSDP(4)	5×5	2.8e-04 (3.5e-09)	27.9	6.3e-06 (3.8e-11)	11.1	(1.9e-12)	9.9
ESDP(5)	rand100	(1.4e-03)	147.2	(1.5e-07)	145.1	(1.8e-08)	183.1
ESDP(10)	rand100	(1.1e-06)	297.9	(3.8e-08)	445.1	(1.2e-08)	663.0
SSDP(4)	rand100	1.2e-03 (3.5e-08)	35.8	1.6e-06 (3.5e-08)	15.4	(3.0e-15)	15.2
SFSDP(4)	rand100	4.4e-04 (2.0e-08)	13.9	1.5e-06 (8.9e-11)	10.3	(4.1e-11)	10.2

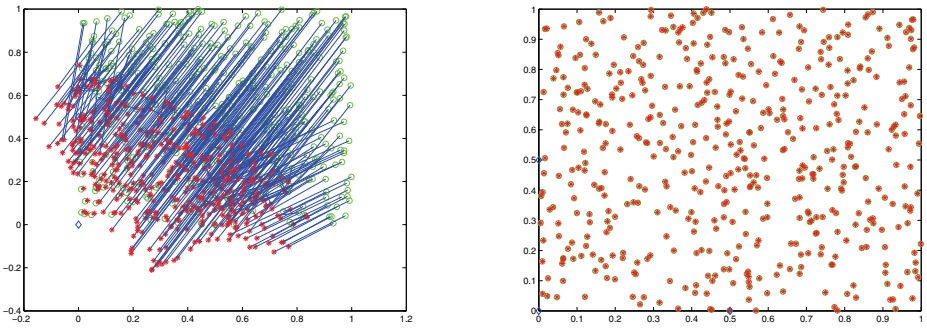

FIG. 1. *ESDP*(10) *after applying the gradient method and SSDP*(4) *after applying lsqnonlin for* 500 *sensors,* 3 *anchors on the boundary and* $\rho = 0.2$. *Here a circle denotes the true location of a sensor,* $\star$ *the computed location of a sensor, and a line segment a deviation from true and computed locations.*

relaxation problems using SeDuMi is longer than SSDP(4) and SFSDP(4). In particular, the difference in cpu time increases as ρ changes to 0.2 and 0.3.

Based on this observation, ESDP, SSDP, and SFSDP are compared on larger-scale problems with 1000 sensors in the 2-dimensional space in the following discussion. Table 2 shows numerical results on the problems with 1000 sensors randomly generated in $[0,1] \times [0,1]$ and exact distances (or $\sigma = 0$). From Tables 1 and 2, we notice the following:

(b) For bd3 with $\rho = 0.1$ in Tables 1 and 2, very large values of rmsd were obtained with ESDP(5), ESDP(10), SSDP(4), and SFSDP(4).

(c) In cases of bd3 with $\rho = 0.2$ and 0.3 in Tables 1 and 2, SSDP(4) and SFSDP(4) attained rmsd small enough to locate the sensors accurately, while ESDP(5) and ESDP(10) did not. See Figure 1.

(d) In Table 2, SSDP(4) spent more cpu time solving SDP relaxation problems by SeDuMi for corner4, 5×5, and bd3 with $\rho = 0.1$, and less cpu time in all other problems than ESDP(5) and ESDP(10).

(e) In all test problems, cpu time consumed by SSDP(4) and SFSDP(4) to solve SDP relaxation problems using SeDuMi decreases as the radio range ρ increases. Note that as the radio range ρ increases, $\mathcal{N}_a$ involves more edges, and the reduced problem after applying the method in section 4.1 becomes sparse and small. As a result, its SDP relaxation can be solved faster. This can be observed in Table 3, where information on the sparsity of the SDP relaxation problems constructed by ESDP(5), FSDP(4), SSDP(4), and SFSDP(4) is exhibited. It is known that the size of the Schur complement matrix, the number of nonzeros of its sparse Cholesky factor, and the maximum size of SDP blocks, which are denoted as sizeR, #nnzL, and maxBl, respectively, are key factors by which to measure the sparsity of the SDP relaxation problems. In SSDP(4) and SFSDP(4) cases, the considerable decrease in sizeR, #nnzL, and maxBl leads to shorter cpu time by SeDuMi as ρ changes from 0.1 to 0.3.

(f) In most of the cases in Tables 1 and 2, SFSDP(4) outperforms all other methods in cpu time and attains comparable or better rmsd than all other methods. Table 3 shows that the number of nonzeros of the sparse Cholesky factor of the Schur complement matrix is much smaller in SFSDP(4) than in all other methods ESDP(5), FSDP(4), and SSDP(4). This is a main reason why SFSDP(4) is faster than all other methods.

TABLE 3

The size of the Schur complement matrix (sizeR), the number of nonzeros in its sparse Cholesky factorization (#nnzL), and the maximum size of SDP blocks (maxBl) when executing SeDuMi for SDP relaxation problems with 500 *sensors and and exact distances* ($\sigma = 0.0$).

	Anchor location	ρ=0.1			ρ=0.3		
		sizeR	#nnzL	maxBl	sizeR	#nnzL	maxBl
ESDP(5)	corner4	18,257	730,478	4	20,431	1,002,821	4
FSDP(4)	corner4	1,690	1,428,895	502	1,966	1,933,916	502
SSDP(4)	corner4	13,137	3,632,361	49	8,481	575,600	27
SFSDP(4)	corner4	4,569	307,470	26	3,396	88,708	15
ESDP(5)	5×5	18,745	765,301	4	24,533	1,107,289	4
FSDP(4)	5×5	1,754	1,539,135	502	2,493	3,108,771	502
SSDP(4)	5×5	10,737	1,618,590	35	4,206	61,440	15
SFSDP(4)	5×5	4,528	306,784	19	3,128	71,805	9

Tables 1 and 2 show that SSDP(4) and SFSDP(4) do not attain small values of rmsd for the problems of bd3 with $\rho = 0.1$, and Table 4 displays how rmsd is improved as κ increases to 5, 6, and 8. See also Figure 2.

ESDP(5) and SFSDP(4) are compared for the problems with 2000 sensors in Table 5. We notice that SFSDP(4) obtained smaller rmsd values in shorter cpu time than ESDP(5) for the tested problems.

5.2. Problems with noisy distances. For numerical experiments in this subsection, the noisy distances are generated by (28), where $\sigma = 0.1$ is used in Tables 6 and 7. In most problems in Tables 6 and 7, we observe the following.

(g) ESDP(5) and/or ESDP(10) spent more cpu time solving SDP relaxation problems using SeDuMi than SFSDP(4). The values of rmsd from the three methods are similar.

TABLE 4
Numerical results on SSDP and SFSDP with $\kappa = 5,\ 6,$ *and* 8 *applied to randomly generated* 2*-dimensional problems with* 500 *sensors and* $\sigma = 0.0$.

	Anchor location	$\kappa = 5$		$\kappa = 6$		$\kappa = 8$	
		rmsd	cpu	rmsd	cpu	rmsd	cpu
SSDP	bd3	4.9e-1(4.5e-1)	381.7	3.1e-1(1.1e-1)	658.5	1.3e-1(7.2e-3)	1767.5
SFSDP	bd3	2.1e-1(1.3e-1)	31.0	5.6e-2(1.5e-2)	50.9	2.9e-2(1.5e-2)	90.9

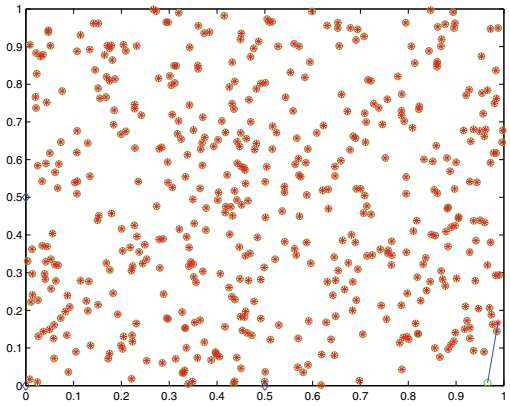

FIG. 2. *SSDP*(8) *after applying nsqnonlin for* 500 *sensors,* 3 *anchors on the boundary, and* $\rho = 0.1$.

TABLE 5
Numerical results on 2*-dimensional problems with randomly generated* 2000 *sensors in* $[0,1] \times [0,1]$ *and exact distances* ($\sigma = 0.0$).

SDP(λ|κ)	Anchor location	ρ=0.1		ρ=0.2		ρ=0.3	
		rmsd	cpu	rmsd	cpu	rmsd	cpu
ESDP(5)	bd3	(6.3e-01)	299.4	(5.8e-01)	319.0	(3.9e-01)	273.1
SFSDP(4)	bd3	1.7e-01 (6.3e-02)	110.8	5.8e-03 (7.6e-08)	69.0	(1.8e-08)	72.5
ESDP(5)	corner4	(9.8e-03)	425.6	(8.9e-03)	449.1	(6.7e-04)	377.0
SFSDP(4)	corner4	5.4e-04 (4.2e-07)	105.7	1.7e-04 (3.0e-08)	45.8	(7.6e-09)	46.3
ESDP(5)	5×5	(1.8e-03)	343.2	(3.5e-07)	163.0	(6.7e-08)	138.2
SFSDP(4)	5×5	3.3e-04 (5.4e-08)	57.0	4.4e-06 (1.2e-12)	25.8	(2.1e-13)	22.7
ESDP(5)	rand100	(3.3e-03)	314.6	(2.8e-07)	198.4	(3.0e-08)	282.4
SFSDP(4)	rand100	2.9e-04 (1.2e-07)	30.7	1.3e-06 (3.0e-11)	24.2	(3.2e-11)	24.1

(h) The difference in cpu time between ESDP(5) and SFSDP(4) is larger for the problems of 2000 sensors shown in Table 7 than those of Table 6.

From the numerical results in sections 5.1 and 5.2, we confirm that the technique described in section 3.3 for exploiting sparsity in the SDP relaxation problems (13) and (14) is very effective; in particular, we observe the computational advantage of SFSDP(4) in getting accurate solutions more quickly than ESDP(5) or ESDP(10) for the problems with exact distances. In addition, for the problems with noisy distances, the computational performance of SFSDP(4) is superior to that of the other methods.

TABLE 6
Numerical results on 2*-dimensional problems with randomly generated* 1000 *sensors in* $[0,1] \times [0,1]$ *and noisy distances* ($\sigma = 0.1$).

SDP($\lambda\|\kappa$)	Anchor location	ρ=0.1 rmsd	ρ=0.1 cpu	ρ=0.2 rmsd	ρ=0.2 cpu	ρ=0.3 rmsd	ρ=0.3 cpu
ESDP(5)	bd3	(6.3e-01)	90.9	(6.3e-01)	65.3	(6.3e-01)	52.2
ESDP(10)	bd3	(6.3e-01)	308.6	(6.6e-01)	297.2	(6.0e-01)	181.7
SFSDP(4)	bd3	4.7e-01 (4.4e-01)	39.2	4.1e-01 (3.5e-01)	22.7	(2.7e-01)	24.0
ESDP(5)	corner4	(2.2e-02)	118.1	(2.3e-02)	79.2	(2.9e-02)	59.9
ESDP(10)	corner4	(1.4e-02)	347.1	(1.3e-02)	381.4	(1.5e-02)	192.7
SFSDP(4)	corner4	5.4e-02 (1.1e-02)	38.1	5.4e-02 (1.6e-02)	24.8	(2.4e-02)	23.4
ESDP(5)	5×5	(1.0e-02)	49.1	(1.1e-02)	40.1	(1.3e-02)	40.1
ESDP(10)	5×5	(6.3e-03)	295.5	(8.9e-03)	337.1	(1.1e-02)	192.0
SFSDP(4)	5×5	1.8e-02 (8.2e-03)	29.0	2.4e-02 (1.5e-02)	16.0	(2.0e-02)	16.0
ESDP(5)	rand100	(3.1e-02)	53.0	(1.1e-02)	49.2	(1.1e-02)	51.8
ESDP(10)	rand100	(8.2e-03)	200.6	(6.3e-03)	406.2	(7.6e-03)	277.1
SFSDP(4)	rand100	2.2e-02 (1.4e-02)	28.2	3.2e-02 (1.4e-02)	15.7	(2.4e-02)	15.7

TABLE 7
Numerical results on 2*-dimensional problems with randomly generated* 2000 *sensors in* $[0,1] \times [0,1]$ *and noisy distances* ($\sigma = 0.1$).

SDP($\lambda\|\kappa$)	Anchor location	ρ=0.1 rmsd	ρ=0.1 cpu	ρ=0.2 rmsd	ρ=0.2 cpu	ρ=0.3 rmsd	ρ=0.3 cpu
ESDP(5)	bd3	(6.3e-01)	373.1	(6.3e-01)	184.2	(6.3e-01)	154.2
SFSDP(4)	bd3	4.5e-01 (4.3e-01)	87.6	4.2e-01 (3.3e-01)	58.8	(2.4e-01)	58.2
ESDP(5)	corner4	(1.6e-02)	511.1	(2.2e-02)	209.1	(3.0e-02)	167.1
SFSDP(4)	corner4	4.9e-02 (8.0e-03)	86.9	5.4e-02 (1.6e-02)	60.5	(2.2e-02)	60.5
ESDP(5)	5×5	(9.4e-03)	322.9	(1.1e-02)	176.8	(1.3e-02)	173.3
SFSDP(4)	5×5	1.8e-02 (7.6e-03)	65.9	2.4e-02 (1.4e-02)	41.6	(2.0e-02)	39.6
ESDP(5)	rand100	(1.1e-02)	351.4	(7.6e-03)	270.7	(8.0e-03)	357.6
SFSDP(4)	rand100	1.9e-02 (1.2e-02)	41.6	2.8e-02 (1.4e-02)	38.2	(2.2e-02)	40.5

TABLE 8
Numerical results on SFSDP(4) *applied to* 2*-dimensional problems with randomly generated* 4000 *sensors in* $[0,1] \times [0,1]$.

σ	Anchor location	ρ=0.06 rmsd	ρ=0.06 cpu	ρ=0.08 rmsd	ρ=0.08 cpu	ρ=0.1 rmsd	ρ=0.1 cpu
0.0	corner4	5.4e-04 (2.5e-06)	948.7	7.6e-04 (4.4e-07)	368.3	(7.6e-07)	205.7
0.0	5×5	5.4e-04 (1.7e-07)	530.5	1.7e-04 (8.9e-08)	200.4	(1.0e-07)	134.1
0.0	rand100	1.9e-03 (1.4e-07)	204.1	6.1e-04 (1.0e-07)	113.1	(8.2e-08)	84.5
0.1	corner4	5.1e-02 (5.2e-03)	607.4	5.0e-02 (7.0e-03)	337.1	(8.2e-03)	214.2
0.1	5×5	1.5e-02 (4.6e-03)	523.0	1.6e-02 (6.0e-03)	241.9	(7.6e-03)	182.6
0.1	rand100	2.0e-02 (1.0e-02)	240.4	2.1e-02 (1.0e-02)	150.5	(1.2e-02)	123.9

5.3. Problems with 4000 sensors. Table 8 shows numerical results on SFSDP(4) applied to problems with 4000 sensors and $\rho = 0.06,\ 0.08,\ 0.1$. We observe that highly accurate rmsd is obtained in all cases and that SeDuMi cpu time varies from 80 to 1000 seconds.

5.4. 3-dimensional problems with 1000 sensors. Solving 3-dimensional problems is far more difficult than solving 2-dimensional problems. Thus, we show only numerical results on problems with 1000 sensors in Table 9. From Table 9, we notice the following:

TABLE 9
Numerical results on SFSDP(4) applied to 3-dimensional problems with randomly generated 1000 sensors in $[0,1]^3$.

σ	Anchor location	ρ=0.3		ρ=0.4		ρ=0.5	
		rmsd	cpu	rmsd	cpu	rmsd	cpu
0.0	corner8	1.2e-03 (5.7e-09)	200.0	5.1e-04 (7.3e-09)	44.3	(4.4e-09)	28.8
0.0	$3 \times 3 \times 3$	3.5e-04 (8.5e-10)	66.6	2.8e-05 (3.0e-11)	20.2	(3.8e-14)	13.8
0.0	rand100	5.7e-05 (7.3e-12)	16.5	1.8e-06 (1.3e-12)	14.2	(2.2e-14)	14.1
0.1	corner8	1.0e-01 (3.5e-02)	172.8	1.1e-01 (4.4e-02)	53.0	(5.4e-02)	38.9
0.1	$3 \times 3 \times 3$	6.6e-02 (3.1e-02)	77.1	6.6e-02 (4.1e-02)	30.9	(4.7e-02)	24.6
0.1	rand100	6.6e-02 (3.1e-02)	28.6	9.5e-02 (4.1e-02)	25.3	(6.0e-02)	15.7

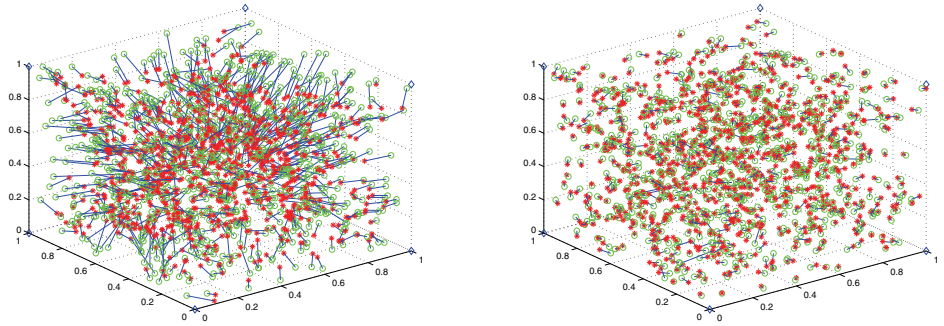

FIG. 3. *SFSDP(4) before and after applying lsqnonlin for* 1000 *sensors,* 8 *anchors at the corner,* $\sigma = 0.1$, *and* $\rho = 0.3$.

(i) When the number of anchors is small (corner8) and/or the radio range ρ is small ($\rho = 0.3$), it takes longer cpu time for SeDuMi to solve SDP relaxation problems.

(j) For problems with no noise ($\sigma = 0.0$), SFSDP(4) attains very small rmsd values; however, as the noisy factor σ grows, the values of rmsd increase. See Figure 3. In Figure 3, we also observe the effectiveness of lsqnonlin in improving rmsd.

6. Concluding remarks. We have formulated the sensor network localization problem with exact distances as a QOP (4) and proposed applying the sparse SDP relaxation [31] with the relaxation order 1 to the QOPs. We have shown that the sparse SDP relaxation is equivalent to the dense relaxation [19] with the same relaxation order 1 and that it is at least as strong as the Biswas–Ye SDP relaxation [2] theoretically. We have also derived a sparse variant of the Biswas–Ye SDP relaxation. For the solutions of the QOPs derived from the problems with exact distances, a MATLAB package SparsePOP is applied and a MATLAB function "lsqnonlin" is used for refining the obtained solution. Numerical results demonstrate that exploiting sparsity in the methods is very effective in computing accurate solutions in less cpu time. In particular, SFSDP provides comparable or better accuracy in less cpu time than all the other SDP relaxations for most of the tested problems.

We have applied SparsePOP to the QOPs (4) to construct their sparse SDP relaxation problems with the relaxation order 1. We should, however, note that SparsePOP is designed for solving general sparse POPs. Thus, its application to the QOPs (4) is not very efficient. For better computational efficiency, it is necessary

to develop codes specialized for generating sparse SDP relaxation problems with the relaxation order 1 from the QOP (4).

REFERENCES

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

[2] P. Biswas and Y. Ye, *Semidefinite programming for ad hoc wireless sensor network localization*, in Proceedings of the Third International Symposium on Information Processing in Sensor Networks, ACM, New York, 2004, pp. 46–54.

[3] P. Biswas and Y. Ye, *A distributed method for solving semidefinite programs arising from ad hoc wireless sensor network localization*, in Multiscale Optimization Methods and Applications, Springer-Varlag, New York, 2006, pp. 69–84.

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

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

[6] J. R. S. Blair and B. Peyton, *An introduction to chordal graphs and clique trees*, in Graph Theory and Sparse Matrix Computation, A. George, J. R. Gilbert, and J. W. H. Liu, eds., Springer-Verlag, New York, 1993, pp. 1–29.

[7] T. F. Coleman and Y. Li, *On the convergence of reflective Newton methods for large-scale nonlinear minimization subject to bounds*, Math. Programming, 67 (1994), pp. 189–224.

[8] T. F. Coleman and Y. Li, *An interior trust region approach for nonlinear minimization subject to bounds*, SIAM J. Optim., 6 (1996), pp. 418–445.

[9] L. Doherty, K. S. J. Pister, and L. El Ghaoui, *Convex position estimation in wireless sensor networks*, in Proceedings of the 20th INFOCOM, IEEE Computer Society, Washington D.C., 2001, pp. 1655–1663.

[10] T. Eren, D. K. Goldenberg, W. Whiteley, Y. R. Wang, A. S. Morse, B. D. O. Anderson, and P. N. Belhumeur, *Rigidity, computation, and randomization in network localization*, in Proceedings of IEEE INFOCOM, IEEE Computer Society, Washington, D.C., 2004, pp. 2673–2684.

[11] T. Fujie and M. Kojima, *Semidefinite relaxation for nonconvex programs*, J. Global Optim., 10 (1997), pp. 367–380.

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

[13] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S.Wicker, *An empirical study of epidemic algorithms in large scale multihop wireless network*, 2002.

[14] A. Howard, M. Matarić, and G. Sukhatme, *Relaxation on a mesh: A formalism for generalized localization*, in the IEEE/RSJ International Conference on Intelligent Robots and Systems, Wailea, Hawaii, 2001, pp. 1055–1060.

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

[16] K. Kobayashi, K. Nakata, and M. Kojima, *A conversion of an SDP having free variables into the standard form SDP*, Comput. Optim. Appl., 36 (2007), pp. 289–307.

[17] M. Kojima, *Conversion methods for large scale SDPs to exploit their structured sparsity*, in The MIT Computation for Design and Optimization Program Distinguished Speaker Series, Massachusetts Institute of Technology, Boston, MA, 2008; also available online from http://www.is.titech.ac.jp/∼kojima/articles/MIT2008.pdf.

[18] M. Kojima, S. Kim, and H. Waki, *Elimination of free variables for solving linear optimization problems efficiently*, in SIAM Conference on Optimization, Boston, MA, 2008; also available online from http://www.is.titech.ac.jp/∼kojima/articles/SIOPT2008.pdf.

[19] J. B. Lasserre, *Global optimization with polynomials and the problem of moments*, SIAM J. Optim., 11 (2001), pp. 796–817.

[20] J. B. Lasserre, *Convergent SDP-relaxations in polynomial optimization with sparsity*, SIAM J. Optim., 17 (2006), pp. 822–843.

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

[22] J. J. Moré and Z. Wu, *Global continuation for distance geometry problems*, SIAM J. Optim., 7 (1997), pp. 814–836.
[23] K. Nakata, K. Fujisawa, M. Fukuda, M. Kojima, and K. Murota, *Exploiting sparsity in semidefinite programming via matrix completion* II: *Implementation and numerical results*, Math. Program., 95 (2003), pp. 303–327.
[24] K. Nakata, M. Yamashita, K. Fujisawa, and M. Kojima, *A parallel primal-dual interior-point method for semidefinite programs using positive definite matrix completion*, Parallel Comput., 32 (2006), pp. 24–43.
[25] J. Nie, *Sum of squares method for sensor network localization*, in Comput. Optim. Appl., to appear.
[26] N. Z. Shor, *Quadratic optimization problems*, Soviet J. Comput. Systems Sci., 25 (1987), pp. 1–11.
[27] N. Z. Shor, *Dual quadratic estimates in polynomial and boolean programming*, Ann. Oper. Res., 25 (1990), pp. 163–168.
[28] A. M. So and Y. Ye, *Theory of semidefinite programming for sensor network localization*, Math. Program. Ser. B, 109 (2007), pp. 367–384.
[29] J. F. Strum, *SeDuMi* 1.02*, a MATLAB toolbox for optimization over symmetric cones*, Optim. Methods Softw., 11 & 12 (1999), pp. 625–653.
[30] P. Tseng, *Second-order cone programming relaxation of sensor network localization*, SIAM J. Optim., 18 (2007), pp. 156–185.
[31] H. Waki, S. Kim, M. Kojima, and M. Muramatsu, *Sums of squares and semidefinite program relaxations for polynomial optimization problems with structured sparsity*, SIAM J. Optim., 17 (2006), pp. 218–242.
[32] H. Waki, S. Kim, M. Kojima, M. Muramatsu, and H. Sugimoto, *Algorithm* 883: *SparsePOP: A sparse semidefinite programming relaxation of polynomial optimization problems*, ACM Trans. Math. Software, 35 (2008), pp. 1–13.
[33] Z. Wang, S. Zheng, Y. Ye, and S. Boyd, *Further relaxations of the semidefinite programming approach to sensor network localization*, SIAM J. Optim., 19 (2008), pp. 655–673.
[34] M. Yamashita, K. Fujisawa, and M. Kojima, *Implementation and evaluation of SDPA* 6.0 *(SemiDefinite Programming Algorithm* 6.0*)*, Optim. Methods Softw., 18 (2003), pp. 491–505.
[35] Y. Ye, http://www.stanford.edu/∼yyye.