ORIGINAL PAPER

A block coordinate descent method for sensor network localization

Mitsuhiro Nishijima[1] · **Kazuhide Nakata[1]**

Abstract

The problem of sensor network localization (SNL) can be formulated as a semidefinite programming problem with a rank constraint. We propose a new method for solving such SNL problems. We factorize a semidefinite matrix with the rank constraint into a product of two matrices via the Burer–Monteiro factorization. Then, we add the difference of the two matrices, with a penalty parameter, to the objective function, thereby reformulating SNL as an unconstrained multiconvex optimization problem, to which we apply the block coordinate descent method. In this paper, we also provide theoretical analyses of the proposed method and show that each subproblem that is solved sequentially by the block coordinate descent method can also be solved analytically, with the sequence generated by our proposed algorithm converging to a stationary point of the objective function. We also give a range of the penalty parameter for which the two matrices used in the factorization agree at any accumulation point. Numerical experiments confirm that the proposed method does inherit the rank constraint and that it estimates sensor positions faster than other methods without sacrificing the estimation accuracy, especially when the measured distances contain errors.

Keywords Sensor network localization · Block coordinate descent · Rank constraint · Burer–Monteiro factorization · Multiconvex optimization

✉ Mitsuhiro Nishijima
nishijima.m.ae@m.titech.ac.jp

Kazuhide Nakata
nakata.k.ac@m.titech.ac.jp

[1] Department of Industrial Engineering and Economics, Tokyo Institute of Technology, Ookayama, Meguro-ku, Tokyo 152-8550, Japan

1 Preliminaries

1.1 Introduction

Sensor network localization (SNL) is the problem of estimating the unknown positions of m sensors from the known positions of n anchors and the measured distances (which may contain measurement errors) between sensor–sensor or sensor–anchor pairs. In this problem, let E_{ss} and E_{sa} be the sets of sensor–sensor and sensor–anchor pairs, respectively, with known measured distances, and let d_{ij} and d_{ik} be the measured distances for each $ij := \{i, j\} \in E_{ss}$ and $ik := \{i, k\} \in E_{sa}$, respectively. Let the anchor coordinates be $\boldsymbol{a}_k = (a_{k1}, \ldots, a_{kd})^\top \in \mathbb{R}^d$ $(k = m + 1, \ldots, m + n)$. Then, the SNL problem is formulated as the following system of equations with variables $\boldsymbol{x}_i \in \mathbb{R}^d$ $(i = 1, \ldots, m)$:

$$\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 = d_{ij} \ (\forall ij \in E_{ss}), \quad \|\boldsymbol{x}_i - \boldsymbol{a}_k\|_2 = d_{ik} \ (\forall ik \in E_{sa}). \tag{1}$$

When a matrix variable Z is introduced, finding the $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ satisfying system (1) is known to be equivalent to finding a solution of the following semidefinite programming (SDP) problem with a rank constraint [4,21]:

$$\begin{vmatrix} \begin{aligned} \min \quad & 0 \\ \text{s.t.} \quad & A_{ij} \cdot Z = d_{ij}^2 \quad (\forall ij \in E_{ss}), \\ & A_{ik} \cdot Z = d_{ik}^2 \quad (\forall ik \in E_{sa}), \\ & Z_{(1:d, 1:d)} = I_d, \\ & \text{rank}(Z) \leq d, \\ & Z \in \mathcal{S}_+^{d+m}. \end{aligned} \end{vmatrix} \tag{2}$$

Here, for each $ij \in E_{ss}$ and $ik \in E_{sa}$,

$$A_{ij} := \begin{pmatrix} \boldsymbol{0}_d \\ \boldsymbol{e}_i - \boldsymbol{e}_j \end{pmatrix} \begin{pmatrix} \boldsymbol{0}_d \\ \boldsymbol{e}_i - \boldsymbol{e}_j \end{pmatrix}^\top, \quad A_{ik} := \begin{pmatrix} \boldsymbol{a}_k \\ -\boldsymbol{e}_i \end{pmatrix} \begin{pmatrix} \boldsymbol{a}_k \\ -\boldsymbol{e}_i \end{pmatrix}^\top,$$

where $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_m$ is the canonical basis of $\mathbb{R}^m$. See Sect. 1.2 for the definitions of the other symbols. When d_{ij} and d_{ik} contain measurement errors, problem (2) generally does not have a solution. Therefore, to account for the case in which d_{ij} and d_{ik} contain errors, researchers have often considered problem (3) defined below [4,8–10,16,19]. In this problem, the distance constraints are incorporated in the objective function in the form of quadratic errors, and a penalty is imposed for violating those constraints. In this paper, we also seek to estimate sensor positions in SNL by solving this problem:

$$\min \quad \frac{1}{2} \sum_{ij \in E_{ss}} (A_{ij} \cdot Z - d_{ij}^2)^2 + \frac{1}{2} \sum_{ik \in E_{sa}} (A_{ik} \cdot Z - d_{ik}^2)^2$$

$$\text{s.t.} \quad Z_{(1:d,1:d)} = I_d, \tag{3}$$

$$\text{rank}(Z) \leq d,$$

$$Z \in \mathcal{S}_+^{d+m}.$$

The SNL problem is generally known to be NP-hard [1], and the formulations of the SNL problem as optimization problems (2) and (3) are also nonconvex. This non-convexity is due to the rank constraint that appears in problems (2) and (3). Therefore, many previous SNL studies removed the rank constraint and relaxed the problem into an SDP problem to estimate approximate sensor positions [2–5,14,22]. Among these methods, sparse full SDP (SFSDP) as proposed by Kim et al. [13] is an especially representative one. However, a solution of the SDP relaxation problem is not always a solution of the original problem. So and Ye [18] referred to problem (1), which has a unique solution and does not have sensor positions satisfying all of the given distances in a higher-dimensional space, as "uniquely localizable". They proved that problem (1) is uniquely localizable if and only if the maximum rank of the solutions of the SDP relaxation problem (2) is d. Therefore, when the SDP problem is solved by the interior-point method, which is a representative method for solving general SDP problems, if the problem is uniquely localizable, then the exact sensor positions can be determined by solving the SDP relaxation problem. Otherwise, an optimal solution of the SDP relaxation problem corresponds to a configuration of the sensors in a higher-dimensional space because of the max-rank property of the interior-point method [11], and it thus might give poorly estimated sensor positions in d-dimensional space.

On the other hand, the exact sensor positions in d-dimensional space can be estimated only if problem (1) has a unique solution (Wan et al. [21] referred to a problem satisfying this condition as "locatable"). Therefore, methods have recently emerged for estimating the sensor positions by solving problem (2) or (3) itself. Wan et al. [21] proposed a method that obtains a solution of problem (2) by solving SDP problems multiple times. This approach is based on the fact that the rank of a matrix being less than or equal to d is equivalent to its $(d + 1)$th and subsequent eigenvalues all being zero. Numerical experiments showed that their method's estimation accuracy was better than that of an SDP relaxation-based method. However, their method took more time than the latter method, because it requires solving the SDP problem via an SDP solver at each iteration. Wan et al. [20] also proposed a method that transforms the SDP problem with the rank constraint into an SDP problem with a complementarity constraint and alternately performs minimization with regard to the two semidefinite matrices that appear in the problem. Numerical experiments also confirmed that this method was more accurate than SDP relaxation methods such as SFSDP. However, as in [21], it took too long to estimate the sensor positions.

Another method for solving general SDP problems is the Burer–Monteiro factorization, in which a semidefinite matrix Z is factorized into the form $VV^\top$ and a nonconvex optimization problem is solved after the factorization [7]. If the number of columns of V is chosen as r in this factorization, then it introduces the constraint that the rank must be less than or equal to r. Therefore, this method is suitable for obtain-

ing low-rank solutions of SDP problems. In a series of studies, Chang and colleagues [8–10] attempted to estimate sensor positions by using the Burer–Monteiro factorization. First, Chang and Xue [9] proposed a method that applies the limited-memory Broyden–Fletcher–Goldfarb–Shanno method to the problem after the Burer–Monteiro factorization. However, they set the number of columns of V used in the factorization to that of the semidefinite matrix before the factorization, so their method does not consider the rank constraint. Second, Chang et al. [10] used the same method as in [9] to estimate sensor positions in three-dimensional space, but unlike in [9], they set the number of columns of V to three, so we can say that this method takes the rank constraint into account. They compared it with SFSDP through numerical experiments and reported that the sensor positions could be estimated more quickly and with the same level of accuracy as SFSDP. However, those experiments involved only small problems with up to 200 sensors. Finally, Chang and Liu [8] proposed a method that they called NLP-FD, which solves the optimization problem obtained from the Burer–Monteiro factorization by the curvilinear search algorithm [23]. Their numerical experiments showed the superiority of NLP-FD over SFSDP when a problem is large in scale and the measured distances include errors.

In this paper, we propose a new method for SNL that accounts for the rank constraint. First, we factorize Z in problem (3) into a product of two matrices through the Burer–Monteiro factorization:

$$Z = \begin{pmatrix} I_d \\ U^\top \end{pmatrix} \begin{pmatrix} I_d \\ V^\top \end{pmatrix}^\top .$$

This factorization is equivalent under the constraint $U - V = \boldsymbol{O}$. Therefore, problem (3) can be transformed into an unconstrained multiconvex optimization problem by adding the difference between the two matrices, with a penalty parameter γ, to the objective function. Then, the block coordinate descent method can be applied to the new objective function, and optimization can be performed sequentially for each column of U and V. We formalize this procedure as Algorithm 1.

We also analyze the proposed method theoretically. First, we show that each subproblem in Algorithm 1 is an unconstrained convex quadratic optimization problem and can be solved analytically (Theorem 2). Second, we show that any accumulation point of the sequence generated by Algorithm 1 is a stationary point of the objective function (Theorem 3). Third, we give a range of γ for which the two matrices U and V used in the factorization coincide at any accumulation point (Theorem 4). Finally, we explain the relationship between the objective function in the reformulated problem and the augmented Lagrangian. Numerical experiments confirm that the proposed method does inherit the rank constraint; furthermore, the results demonstrate not only that our method estimates sensor positions faster than SFSDP and NLP-FD without sacrificing estimation accuracy, especially when the measured distances include errors, but also that our method does not run out of memory even for large-scale SNL problems.

The rest of this paper is organized as follows. In Sect. 2, we present the proposed method and analyze it theoretically. In Sect. 3, we compare it with the other methods to confirm its effectiveness. Finally, in Sect. 4, we present our conclusions and suggest possible future work.

1.2 Notation

- $\mathbb{N}$ denotes the set of natural numbers without zero. $\mathbb{R}^p$ denotes the set of p-dimensional real vectors, and $\mathbf{0}_p$ denotes the zero vector of $\mathbb{R}^p$. When the size is clear from the context, we omit the size subscript at the lower right. $\mathbb{R}^{p \times q}$ denotes the set of $p \times q$ real matrices. Let I_p be the identity matrix of $\mathbb{R}^{p \times p}$ and $\mathbf{O}$ be the zero matrix of an appropriate size. $\mathcal{S}_+^p$ denotes the set of $p \times p$ symmetric positive semidefinite matrices.
- For $\boldsymbol{x} \in \mathbb{R}^p$, $\|\boldsymbol{x}\|_2$ denotes the 2-norm of $\boldsymbol{x}$. For $A, B \in \mathbb{R}^{p \times q}$, $A \cdot B$ means the inner product between A and B, denoted by $\operatorname{tr}(A^\top B)$; $\|A\|_F$ denotes the Frobenius norm of A; $A_{(i:j,k:l)}$ denotes the submatrix of A obtained by choosing the $\{i, \ldots, j\}$th rows of A and the $\{k, \ldots, l\}$th columns of A; and $\operatorname{rank}(A)$ denotes the rank of A. For any symmetric matrix A, $\lambda_{\max}(A)$ denotes its maximum eigenvalue.
- For $i = 1, \ldots, m$, $E_{ss}[i]$ and $E_{sa}[i]$ denote the sets of sensors and anchors, respectively, that are connected directly to sensor i.

2 Proposed method and analyses

2.1 Proposed method

Problem (3) is an SDP problem with a rank constraint and is difficult to solve directly. In this subsection, we propose a new method that transforms problem (3) into an unconstrained multiconvex optimization problem and solves the latter problem sequentially to estimate sensor positions. First, a matrix Z satisfies the three constraints in problem (3) if and only if it can be factorized into the product of two matrices as follows:

$$Z = \begin{pmatrix} I_d & U \\ U^\top & U^\top U \end{pmatrix} = \begin{pmatrix} I_d \\ U^\top \end{pmatrix} \begin{pmatrix} I_d \\ V^\top \end{pmatrix}^\top, \quad U - V = \mathbf{O}.$$

Thus, problem (3) is equivalent to

$$
\begin{aligned}
\min \quad & f(U, V) := \frac{1}{2} \sum_{ij \in E_{ss}} \left(A_{ij} \cdot \begin{pmatrix} I_d \\ U^\top \end{pmatrix} \begin{pmatrix} I_d \\ V^\top \end{pmatrix}^\top - d_{ij}^2 \right)^2 \\
& + \frac{1}{2} \sum_{ik \in E_{sa}} \left(A_{ik} \cdot \begin{pmatrix} I_d \\ U^\top \end{pmatrix} \begin{pmatrix} I_d \\ V^\top \end{pmatrix}^\top - d_{ik}^2 \right)^2 \\
\text{s.t.} \quad & U - V = \mathbf{O}.
\end{aligned}
\tag{4}
$$

To make problem (4) easier to solve, we remove the constraint $U - V = \mathbf{O}$ and add a quadratic penalty term $\gamma/2\|U - V\|_F$ with a penalty parameter γ (> 0) to the objective function; as a result, the objective function takes larger values as the constraint $U - V = \mathbf{O}$ is more strongly violated. In other words, we adopt the

following unconstrained optimization problem:

$$\min \quad F(U, V; \gamma) := \frac{\gamma}{2} \|U - V\|_F^2 + f(U, V). \tag{5}$$

In the proposed algorithm, we let $U = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m)$ and $V = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m)$ and then perform minimization with regard to $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$, i.e., each column of U and V sequentially. Specifically, the procedure is as listed in Algorithm 1.

Algorithm 1 Proposed algorithm for problem (5)

Require: an initial point $U^{(0)} = (\boldsymbol{u}_1^{(0)}, \ldots, \boldsymbol{u}_m^{(0)})$, $V^{(0)} = (\boldsymbol{v}_1^{(0)}, \ldots, \boldsymbol{v}_m^{(0)}) \in \mathbb{R}^{d \times m}$, a penalty parameter
γ, a parameter ϵ
Ensure: a generated sequence $\{(U^{(p)}, V^{(p)})\}$
1: **while** 1 **do**
2:    **for** $i = 1, \ldots, m$ **do**
3:       $\boldsymbol{u}_i^{(p)} = \operatorname{argmin}_{\boldsymbol{u}_i} F(\boldsymbol{u}_1^{(p)}, \ldots, \boldsymbol{u}_{i-1}^{(p)}, \boldsymbol{u}_i, \boldsymbol{u}_{i+1}^{(p-1)}, \ldots, \boldsymbol{u}_m^{(p-1)}, V^{(p-1)}; \gamma)$.
4:    **end for**
5:    **for** $i = 1, \ldots, m$ **do**
6:       $\boldsymbol{v}_i^{(p)} = \operatorname{argmin}_{\boldsymbol{v}_i} F(U^{(p)}, \boldsymbol{v}_1^{(p)}, \ldots, \boldsymbol{v}_{i-1}^{(p)}, \boldsymbol{v}_i, \boldsymbol{v}_{i+1}^{(p-1)}, \ldots, \boldsymbol{v}_m^{(p-1)}; \gamma)$.
7:    **end for**
8:    **if** $\max \left\{ \frac{2\|U^{(p)} - V^{(p)}\|_F}{\|U^{(p)}\|_F + \|V^{(p)}\|_F}, \frac{\|U^{(p)} - U^{(p-1)}\|_F}{\|U^{(p-1)}\|_F}, \frac{\|V^{(p)} - V^{(p-1)}\|_F}{\|V^{(p-1)}\|_F} \right\} < \epsilon$ **then**
9:       stop algorithm.
10:   **else**
11:      $p = p + 1$
12:   **end if**
13: **end while**

The proposed method has the following advantages over other methods:

(i) The SDP problem with the rank constrained (3) is equivalent to problem (4), from which we obtained problem (5) by incorporating $U - V = O$ in the objective function as a quadratic penalty term with a penalty parameter γ. As we will see from Theorem 4, if γ is larger than a real-valued threshold, then the $U^{(p)}$ and $V^{(p)}$ generated by Algorithm 1 coincide with each other at any accumulation point, thereby satisfying the constraint of problem (4). Therefore, the proposed method inherits the rank constraint in problem (3) and retains the potential capability to estimate sensor positions accurately for problems that are not uniquely localizable. This advantage will be verified in Sect. 3.1.

(ii) As we will see from Theorem 2, each subproblem appearing inside a **for** statement in Algorithm 1 is an unconstrained convex quadratic optimization problem. The solution of each subproblem can be obtained analytically, because the solution process can be reduced to solving a system of linear equations with an invertible coefficient matrix of size d. Because d is at most three in real situations, the system can be solved rapidly and without running out of memory, regardless of the number of sensors m. Moreover, the subproblems only need to be solved $2m$ times (i.e., a number proportional to m) for each outer loop. Therefore, especially in the case of large-scale SNL problems, we expect faster estimates of the sensor positions as compared with other methods. This advantage will be verified in Sect. 3.2.

2.2 Analyses of the proposed method

In this subsection, we present theoretical analyses of problem (5) and Algorithm 1. First, we impose an assumption about the problem that we are examining.

Assumption 1 All sensors are connected to an anchor either directly or indirectly.

The same assumption was also made in [8,18,22] and is very natural when estimating sensor positions: if a sensor is not connected to any anchors, then its absolute position cannot be determined uniquely.

First, we prove that the optimal solution of each subproblem in Algorithm 1 can be obtained uniquely as an analytical solution.

Theorem 2 *Fix $U' = (u'_1, \ldots, u'_m)$ and $V' = (v'_1, \ldots, v'_m)$ arbitrarily. Then, for each $i = 1, \ldots, m$, the solutions u^*_i, v^*_i of the following two optimization problems*

$$\min_{u_i \in \mathbb{R}^d} F(u'_1, \ldots, u'_{i-1}, u_i, u'_{i+1}, \ldots, u'_m, V'; \gamma),$$

$$\min_{v_i \in \mathbb{R}^d} F(U', v'_1, \ldots, v'_{i-1}, v_i, v'_{i+1}, \ldots, v'_m; \gamma)$$

are respectively $u^*_i = A_{u_i}^{-1} b_{u_i}$, $v^*_i = A_{v_i}^{-1} b_{v_i}$, where

$$A_{u_i} := \gamma I_d + \sum_{j \in E_{ss}[i]} (v'_i - v'_j)(v'_i - v'_j)^\top + \sum_{k \in E_{sa}[i]} (v'_i - a_k)(v'_i - a_k)^\top,$$

$$b_{u_i} := \gamma v'_i + \sum_{j \in E_{ss}[i]} ((u'_j)^\top v'_i - (u'_j)^\top v'_j + d_{ij}^2)(v'_i - v'_j)$$

$$+ \sum_{k \in E_{sa}[i]} (a_k^\top v'_i - a_k^\top a_k + d_{ik}^2)(v'_i - a_k),$$

$$A_{v_i} := \gamma I_d + \sum_{j \in E_{ss}[i]} (u'_i - u'_j)(u'_i - u'_j)^\top + \sum_{k \in E_{sa}[i]} (u'_i - a_k)(u'_i - a_k)^\top,$$

$$b_{v_i} := \gamma u'_i + \sum_{j \in E_{ss}[i]} ((v'_j)^\top u'_i - (v'_j)^\top u'_j + d_{ij}^2)(u'_i - u'_j)$$

$$+ \sum_{k \in E_{sa}[i]} (a_k^\top u'_i - a_k^\top a_k + d_{ik}^2)(u'_i - a_k).$$

Proof If we focus on only u_i in $F(U, V; \gamma)$ in particular, then we can represent $F(u'_1, \ldots, u'_{i-1}, u_i, u'_{i+1}, \ldots, u'_m, V'; \gamma)$ as

$$F(u'_1, \ldots, u'_{i-1}, u_i, u'_{i+1}, \ldots, u'_m, V'; \gamma)$$
$$= \frac{1}{2} u_i^\top A_{u_i} u_i - b_{u_i}^\top u_i + [\text{a constant unrelated to } u_i]. \tag{6}$$

From Eq. (6), we can see that $F(\boldsymbol{u}_1', \ldots, \boldsymbol{u}_{i-1}', \boldsymbol{u}_i, \boldsymbol{u}_{i+1}', \ldots, \boldsymbol{u}_m', V'; \gamma)$ is γ-strongly convex. Thus, the optimal solution of

$$\min_{\boldsymbol{u}_i \in \mathbb{R}^d} F(\boldsymbol{u}_1', \ldots, \boldsymbol{u}_{i-1}', \boldsymbol{u}_i, \boldsymbol{u}_{i+1}', \ldots, \boldsymbol{u}_m', V'; \gamma)$$

is the stationary point of $F(\boldsymbol{u}_1', \ldots, \boldsymbol{u}_{i-1}', \boldsymbol{u}_i, \boldsymbol{u}_{i+1}', \ldots, \boldsymbol{u}_m', V'; \gamma)$. Because

$$\nabla_{\boldsymbol{u}_i} F(\boldsymbol{u}_1', \ldots, \boldsymbol{u}_{i-1}', \boldsymbol{u}_i, \boldsymbol{u}_{i+1}', \ldots, \boldsymbol{u}_m', V'; \gamma) = A_{\boldsymbol{u}_i} \boldsymbol{u}_i - \boldsymbol{b}_{\boldsymbol{u}_i}$$

and $A_{\boldsymbol{u}_i}$ is positive definite and invertible in particular, we can obtain $\boldsymbol{u}_i^*$. We can also obtain $\boldsymbol{v}_i^*$ from the same calculation. $\qquad\square$

Next, we show that the sequence generated by Algorithm 1 converges to a stationary point of the objective function F.

Theorem 3 *Fix the penalty parameter γ in problem (5) arbitrarily. Let $\mathcal{N}$ be the set of stationary points of F. Then, the sequence $\{(U^{(p)}, V^{(p)})\}_{p=1}^{\infty}$ generated by Algorithm 1 satisfies*

$$\lim_{p \to \infty} \inf_{(U,V) \in \mathcal{N}} \|(U^{(p)}, V^{(p)}) - (U, V)\|_F = 0. \tag{7}$$

In particular, any accumulation point (U^*, V^*) of the generated sequence is a stationary point of F.

The consequence of Theorem 3 is based on the result of [25]. By Corollary 2.4 in [25], if all three of the following conditions are satisfied, then Eq. (7) holds when the *stationary point* of F in the definition of $\mathcal{N}$ is replaced by the *Nash equilibrium* of F (see Definition 2 below).

Condition (a): F is continuous, bounded below, and has a Nash equilibrium.
Condition (b): The objective function of each subproblem is strongly convex.[1]
Condition (c): The sequence generated by Algorithm 1 is bounded.

In the following, we prove Theorem 3 by showing the equivalence between the Nash equilibrium and the stationary point of F and then verifying that all three conditions are satisfied.

We can see from Eq. (6) that the function $F(U, V; \gamma)$ is convex on $\mathbb{R}^d$ if we focus only on each column of U and V. Such a function F with this property is called multiconvex [25]. For simplicity, let $\mathcal{X} := \mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_s}$ (where $n_1, \ldots, n_s \in \mathbb{N}$),[2] and when $\boldsymbol{x} \in \mathcal{X}$ is represented as $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_s)$, then $\boldsymbol{x}_i \in \mathbb{R}^{n_i}$ $(i = 1, \ldots, s)$.

Definition 1 A function $g : \mathcal{X} \to \mathbb{R}$ is called multiconvex on $\mathcal{X}$ (with respect to the block division $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_s) \in \mathcal{X}$) if for all $i = 1, \ldots, s$ and all $\boldsymbol{x}_j \in \mathbb{R}^{n_j}$ $(j = 1, \ldots, i-1, i+1, \ldots, m)$, the function

$$g(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{i-1}, \cdot, \boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_s) : \mathbb{R}^{n_i} \to \mathbb{R}$$

---

[1] To be more precise, a stronger assumption about the parameter of the strongly convex function is needed. However, in the present problem, the stronger assumption is satisfied automatically because the parameter is a constant γ. See Assumption 2 in [25] for details.

[2] In the problem, we are considering the case in which $s = 2m$ and $n_i \equiv d$, so the simplification of $\mathcal{X}$ does not affect the discussion in this paper.

is convex.

One of the concepts of minimality for a multiconvex function is the Nash equilibrium [25], which appears in Condition (a).

Definition 2 For a function $g : \mathcal{X} \to \mathbb{R}$, $(x_1^*, \ldots, x_s^*) \in \mathcal{X}$ is called a Nash equilibrium of g (with respect to the block division as in Definition 1) if

$$g(x_1^*, \ldots, x_{i-1}^*, x_i^*, x_{i+1}^*, \ldots, x_s^*) \le g(x_1^*, \ldots, x_{i-1}^*, x_i, x_{i+1}^*, \ldots, x_s^*)$$

holds for all $i = 1, \ldots, s$ and all $x_i \in \mathbb{R}^{n_i}$.

Gorski et al. [12] proved the equivalence between the stationary point and the Nash equilibrium in the case of $s = 2$.[3] Herein, we extend the equivalence to the case of arbitrary s.

Lemma 1 *Let* $g : \mathcal{X} \to \mathbb{R}$ *be once differentiable and multiconvex. Then,* $x^* = (x_1^*, \ldots, x_s^*) \in \mathcal{X}$ *is a stationary point of g if and only if* x^* *is a Nash equilibrium of g.*

Proof We begin by proving the "if" part. If we assume that x^* is a stationary point of g, then because
$$g_i(x_i) := g(x_1^*, \ldots, x_{i-1}^*, x_i, x_{i+1}^*, \ldots, x_s^*)$$
is convex on $\mathbb{R}^{n_i}$ for all $i = 1, \ldots, m$,

$$g_i(x_i) \ge g_i(x_i^*) + \nabla_{x_i} g_i(x_i^*)^\top (x_i - x_i^*) \tag{8}$$

holds for all $x_i \in \mathbb{R}^{n_i}$. Because $\nabla_{x_i} g_i(x_i^*) = 0$ follows from the assumption of x^* being a stationary point of g, we can say from inequality (8) that $g_i(x_i) \ge g_i(x_i^*)$ for all $x_i \in \mathbb{R}^{n_i}$. Because i is arbitrary, we can conclude that x^* is a Nash equilibrium of g.

Next, we prove the "only if" part. If we assume that x^* is a Nash equilibrium of g, then for each $i = 1, \ldots, s$, $g_i(x_i)$ attains its minimum value at $x_i = x_i^*$, from which we obtain $\nabla_{x_i} g_i(x_i^*) = 0$. Thus, x^* is a stationary point of g. □

Lemma 2 *Suppose that Assumption 1 holds. Then, for all α, the level set $S_F(\alpha) := \{(U, V) \mid F(U, V; \gamma) \le \alpha\}$ is bounded and closed.*

Because similar (but not the same) results were already pointed out in [10,18], we omit the proof of Lemma 2 because of the page limit. Note that if Assumption 1 does not hold, then $S_F(\alpha)$ is always not bounded.

Corollary 1 *For any initial point $(U^{(0)}, V^{(0)})$, the sequence $\{(U^{(p)}, V^{(p)})\}_{p=1}^\infty$ generated by Algorithm 1 is bounded.*

Proof Let $\alpha := F(U^{(0)}, V^{(0)}; \gamma)$. Because each minimization subproblem in Algorithm 1 is strictly optimized, we can say that $(U^{(p)}, V^{(p)}) \in S_F(\alpha)$ for all $p \in \mathbb{N}$, i.e., $\{(U^{(p)}, V^{(p)})\}_{p=1}^\infty \subseteq S_F(\alpha)$, which is bounded from Lemma 2. □

---

[3] In [12], the term "partial optimum" is used instead of "Nash equilibrium."

We can now prove Theorem 3 on the basis of the above claims.

Proof The continuity and below-boundedness of F are evident from its definition. When combined with Lemma 2, the global optimal solution of problem (5) is guaranteed to exist, from which the existence of a Nash equilibrium can be proved. Therefore, we can say that Condition (a) holds. In addition, from Eq. (6), the objective function of each subproblem is γ-strongly convex, and thus Condition (b) holds. Finally, Condition (c) is also satisfied from Corollary 1. Therefore, the conditions of Corollary 2.4 in [25] are all satisfied, from which we can show Eq. (7) by using the equivalence between the stationary point and the Nash equilibrium that was shown in Lemma 1. Because F is of class C^1, $\mathcal{N}$ is closed, from which we can easily show the last part of Theorem 3. $\qquad\square$

Finally, we show that the $U^{(p)}$ and $V^{(p)}$ generated by Algorithm 1 coincide with each other at any accumulation point if γ is larger than a real-valued threshold, which does not generally hold in the quadratic penalty method.

Theorem 4 *For any initial point $(U^{(0)}, V^{(0)})$ such that $U^{(0)} = V^{(0)}$, if*

$$\gamma > \frac{1}{2}\sqrt{2f(U^{(0)}, V^{(0)})} \max_{1 \le i \le m} \sqrt{4|E_{ss}[i]| + |E_{sa}[i]|}, \tag{9}$$

then any accumulation point (U^*, V^*) of the sequence $\{(U^{(p)}, V^{(p)})\}_{p=1}^{\infty}$ generated by Algorithm 1 satisfies $U^* = V^*$.

Proof For each $p \in \mathbb{N}$, $ij \in E_{ss}$, and $ik \in E_{sa}$, let

$$\alpha_{ij}^{(p)} := A_{ij} \cdot \begin{pmatrix} I_d \\ (U^{(p)})^\top \end{pmatrix} \begin{pmatrix} I_d \\ (V^{(p)})^\top \end{pmatrix}^\top - d_{ij}^2,$$

$$\alpha_{ik}^{(p)} := A_{ik} \cdot \begin{pmatrix} I_d \\ (U^{(p)})^\top \end{pmatrix} \begin{pmatrix} I_d \\ (V^{(p)})^\top \end{pmatrix}^\top - d_{ik}^2.$$

Then, because the initial point satisfies $U^{(0)} = V^{(0)}$ and the value of the objective function F decreases monotonically by Algorithm 1, we can conclude that for all $p \in \mathbb{N}$,

$$f(U^{(0)}, V^{(0)}) = F(U^{(0)}, V^{(0)}; \gamma) \ge F(U^{(p)}, V^{(p)}; \gamma)$$
$$\ge \frac{1}{2} \sum_{ij \in E_{ss}} (\alpha_{ij}^{(p)})^2 + \frac{1}{2} \sum_{ik \in E_{sa}} (\alpha_{ik}^{(p)})^2. \tag{10}$$

By taking a subsequence, without loss of generality, we can assume that $\{(U^{(p)}, V^{(p)})\}_{p=1}^{\infty}$ itself converges to (U^*, V^*). Then, it follows from inequality (10) that

$$\frac{1}{2} \sum_{ij \in E_{ss}} (\alpha_{ij}^*)^2 + \frac{1}{2} \sum_{ik \in E_{sa}} (\alpha_{ik}^*)^2 \le f(U^{(0)}, V^{(0)}). \tag{11}$$

By Theorem 3, (U^*, V^*) is a stationary point of F, from which we have

$$\nabla_{(U,V)} F(U^*, V^*; \gamma) = \mathbf{O}.$$

Thus, let U_l, $V_l \in \mathbb{R}^m$ be the respective lth-column vectors of $U^\top$, $V^\top$ for each $l = 1, \ldots, d$; then, $\nabla_{U_l} F(U^*, V^*; \gamma) = \nabla_{V_l} F(U^*, V^*; \gamma) = \mathbf{0}$ holds. For each $ik \in E_{\mathrm{sa}}$ and $l = 1, \ldots, d$, let $\mathbf{b}_{ik}^l$ be an m-dimensional vector such that its ith component is $-a_{kl}$ and all other components are zeros. Furthermore, for each $ij \in E_{\mathrm{ss}}$ and $ik \in E_{\mathrm{sa}}$, let $\bar{A}_{ij}$ and $\bar{A}_{ik}$ respectively be

$$\bar{A}_{ij} := (A_{ij})_{(d+1:d+m, d+1:d+m)}, \quad \bar{A}_{ik} := (A_{ik})_{(d+1:d+m, d+1:d+m)}.$$

Using these symbols, we can represent $F(U, V; \gamma)$ as

$$F(U, V; \gamma) = \frac{\gamma}{2} \sum_{l=1}^{d} \|U_l - V_l\|_2^2 + \frac{1}{2} \sum_{ij \in E_{\mathrm{ss}}} \left(\sum_{l=1}^{d} U_l^\top \bar{A}_{ij} V_l - d_{ij}^2 \right)^2$$

$$+ \frac{1}{2} \sum_{ik \in E_{\mathrm{sa}}} \left(\sum_{l=1}^{d} U_l^\top \bar{A}_{ik} V_l + \sum_{l=1}^{d} (\mathbf{b}_{ik}^l)^\top (U_l + V_l) + \mathbf{a}_k^\top \mathbf{a}_k - d_{ik}^2 \right)^2.$$

Because

$$\nabla_{U_l} F(U^*, V^*; \gamma) = \gamma (U_l^* - V_l^*) + \sum_{ij \in E_{\mathrm{ss}}} \alpha_{ij}^* \bar{A}_{ij} V_l^* + \sum_{ik \in E_{\mathrm{sa}}} \alpha_{ik}^* (\bar{A}_{ik} V_l^* + \mathbf{b}_{ik}^l)$$

$$= \mathbf{0},$$

$$\nabla_{V_l} F(U^*, V^*; \gamma) = \gamma (V_l^* - U_l^*) + \sum_{ij \in E_{\mathrm{ss}}} \alpha_{ij}^* \bar{A}_{ij} U_l^* + \sum_{ik \in E_{\mathrm{sa}}} \alpha_{ik}^* (\bar{A}_{ik} U_l^* + \mathbf{b}_{ik}^l)$$

$$= \mathbf{0}$$

for all $l = 1, \ldots, d$, we obtain

$$(U_l^* - V_l^*)^\top \nabla_{U_l} F(U^*, V^*; \gamma) + (V_l^* - U_l^*)^\top \nabla_{V_l} F(U^*, V^*; \gamma)$$

$$= (U_l^* - V_l^*)^\top \left\{ 2\gamma I_m - \left(\sum_{ij \in E_{\mathrm{ss}}} \alpha_{ij}^* \bar{A}_{ij} + \sum_{ik \in E_{\mathrm{sa}}} \alpha_{ik}^* \bar{A}_{ik} \right) \right\} (U_l^* - V_l^*) = 0.$$

$$(12)$$

For convenience, let

$$\bar{A} := \sum_{ij \in E_{\mathrm{ss}}} \alpha_{ij}^* \bar{A}_{ij} + \sum_{ik \in E_{\mathrm{sa}}} \alpha_{ik}^* \bar{A}_{ik}.$$

Next, we seek to prove the following inequality:

$$\lambda_{\max}(\bar{A}) \leq \sqrt{2 f(U^{(0)}, V^{(0)})} \max_{1 \leq i \leq m} \sqrt{4 |E_{\mathrm{ss}}[i]| + |E_{\mathrm{sa}}[i]|}.$$

$$(13)$$

In fact, if inequality (13) can be shown, then because γ satisfies inequality (9), $2\gamma I_m - \bar{A}$ is a positive definite matrix, and thus, $U_l^* = V_l^*$ from equality (12). Because of the arbitrariness of l, we can eventually conclude that $U^* = V^*$. Therefore, we need only prove inequality (13).

It follows from the Gershgorin circle theorem that

$$\lambda_{\max}(\bar{A}) \leq \max_{1 \leq i \leq m} \left\{ \sum_{j \in E_{ss}[i]} \alpha_{ij}^* + \sum_{k \in E_{sa}[i]} \alpha_{ik}^* + \sum_{j \in E_{ss}[i]} |\alpha_{ij}^*| \right\}. \tag{14}$$

For each $i = 1, \ldots, m$, let $v(i)$ be the optimal value of the following optimization problem:[4]

$$\left| \begin{array}{ll} \max & \sum_{j \in E_{ss}[i]} \alpha_{ij} + \sum_{k \in E_{sa}[i]} \alpha_{ik} + \sum_{j \in E_{ss}[i]} |\alpha_{ij}| \\ \text{s.t.} & \frac{1}{2} \sum_{ij \in E_{ss}} \alpha_{ij}^2 + \frac{1}{2} \sum_{ik \in E_{sa}} \alpha_{ik}^2 \leq f(U^{(0)}, V^{(0)}), \end{array} \right.$$

where the right side of inequality (14) does not exceed $\max_{1 \leq i \leq m} v(i)$ because of inequality (11). We can easily check that $v(i)$ is equal to the optimal value of the following optimization problem for each $i = 1, \ldots, m$:

$$\left| \begin{array}{ll} \max & 2 \sum_{j \in E_{ss}[i]} \alpha_{ij} + \sum_{k \in E_{sa}[i]} \alpha_{ik} \\ \text{s.t.} & \frac{1}{2} \sum_{j \in E_{ss}[i]} \alpha_{ij}^2 + \frac{1}{2} \sum_{k \in E_{sa}[i]} \alpha_{ik}^2 = f(U^{(0)}, V^{(0)}). \end{array} \right. \tag{15}$$

Using the method of Lagrange multipliers, we can see that the optimal value of problem (15) is $\sqrt{2f(U^{(0)}, V^{(0)})}\sqrt{4|E_{ss}[i]| + |E_{sa}[i]|}$. Therefore,

$$\max_{1 \leq i \leq m} \left\{ \sum_{j \in E_{ss}[i]} \alpha_{ij}^* + \sum_{k \in E_{sa}[i]} \alpha_{ik}^* + \sum_{j \in E_{ss}[i]} |\alpha_{ij}^*| \right\}$$
$$\leq \max_{1 \leq i \leq m} v(i) = \sqrt{2f(U^{(0)}, V^{(0)})} \max_{1 \leq i \leq m} \sqrt{4|E_{ss}[i]| + |E_{sa}[i]|},$$

which implies inequality (13). □

2.3 Relationship to the augmented Lagrangian

In this paper, we adopt the quadratic-penalty-based method, in which the equality constraint $U - V = O$ is incorporated in the objective function as a quadratic penalty term and the resulting new objective function is minimized. On the other hand, there are also

---

[4] Although we use the notations "ij" and "ik" to denote the indices of the variables α in the sums in the constraint of this optimization problem, they are not related to i ($= 1, \ldots, m$), which is fixed here.

methods such as the augmented Lagrangian method [17] and the alternating direction method of multipliers [6] that minimize the augmented Lagrangian, which contains not only the quadratic penalty term but also the Lagrange multiplier term. It is known that the augmented Lagrangian method is more efficient than the quadratic penalty method. For example, while the quadratic penalty method requires the penalty parameter to diverge to positive infinity, the augmented Lagrangian method does not require it to diverge, and the sequence obtained by the augmented Lagrangian method converges faster than that obtained by the quadratic penalty method [17, Example 17.4]. Hence, we explain that problem (5) can be regarded as a minimization problem of the augmented Lagrangian with an exact Lagrangian multiplier.

$\Lambda = \boldsymbol{O}$ is the exact Lagrange multiplier of problem (4). In fact, for all local optimum solutions (U^*, V^*) of problem (4), because problem (4) satisfies the linear independence constraint qualification, there exists a Lagrange multiplier $\Lambda^* \in \mathbb{R}^{d \times m}$ satisfying the Karush–Kuhn–Tucker condition. In other words, if we let the Lagrangian for problem (4) be

$$\mathcal{L}(U, V, \Lambda) := f(U, V) - \Lambda \cdot (U - V),$$

then

$$\nabla_U \mathcal{L}(U^*, V^*, \Lambda^*) = \nabla_U f(U^*, V^*) - \Lambda^* = \boldsymbol{O}, \tag{16a}$$

$$\nabla_V \mathcal{L}(U^*, V^*, \Lambda^*) = \nabla_V f(U^*, V^*) + \Lambda^* = \boldsymbol{O}, \tag{16b}$$

$$\nabla_\Lambda \mathcal{L}(U^*, V^*, \Lambda^*) = -(U^* - V^*) = \boldsymbol{O} \tag{16c}$$

hold. We get $U^* = V^*$ from Eq. (16c) and denote both of them as W^*. Because $f(U, V) = f(V, U)$ ($\forall U, V \in \mathbb{R}^{d \times m}$), $\nabla_U f(W^*, W^*) = \nabla_V f(W^*, W^*)$. Using this equation and Eqs. (16a) and (16b), we obtain $\Lambda^* = \boldsymbol{O}$. Therefore, the augmented Lagrangian with the Lagrange multiplier $\Lambda = \boldsymbol{O}$ and penalty parameter γ is

$$f(U, V) - \boldsymbol{O} \cdot (U - V) + \frac{\gamma}{2} \|U - V\|_F^2 = f(U, V) + \frac{\gamma}{2} \|U - V\|_F^2,$$

which is the definition of $F(U, V; \gamma)$ itself. Hence, problem (5) can be regarded as a minimization problem of the augmented Lagrangian with the exact Lagrange multiplier $\Lambda = \boldsymbol{O}$ for problem (4).

3 Numerical experiments

In this section, we use numerical simulation to verify the advantages (i) and (ii) described in Sect. 2.1 for the proposed method. We begin by confirming that our method does inherit the rank constraint; to confirm this, we compare it with an SDP relaxation-based method for a problem that is locatable but not uniquely localizable. Next, to confirm the effectiveness of the proposed method, we compare its estimation time and estimation accuracy with those of other methods by using artificial data under various conditions. All experiments were conducted on a computer with the macOS

Catalina operating system, an Intel Core i5-8279U 2.40 GHz CPU, and 16 GB of memory. All the algorithms were implemented using MATLAB (R2020a). The parameter ϵ in Algorithm 1 was set to 10^{-5} throughout the experiments.

3.1 Comparison with SFSDP for a problem that is not uniquely localizable

In this subsection, we demonstrate that the proposed method has the capability to estimate sensor positions accurately for a problem that is locatable but not uniquely localizable. Specifically, we examine a problem from [18]:

$$\|x_1 - x_2\|_2 = \sqrt{10}/5, \ \|x_1 - a_4\|_2 = \sqrt{5}/2, \ \|x_1 - a_5\|_2 = \sqrt{5}/2,$$
$$\|x_2 - a_3\|_2 = \sqrt{85}/10, \ \|x_2 - a_5\|_2 = \sqrt{65}/10,$$

where $a_3 = (0, 1.4)^\top$, $a_4 = (-1, 0)^\top$, and $a_5 = (1, 0)^\top$, and the true positions of the two sensors are $x_1^{\text{true}} = (0, 0.5)^\top$, $x_2^{\text{true}} = (0.6, 0.7)^\top$. For this problem, Algorithm 1 was executed after fixing the penalty parameter γ as $\sqrt{2 f(U^{(0)}, V^{(0)})} \max_{1 \le i \le m} \sqrt{4|E_{\text{ss}}| + |E_{\text{sa}}|}/2$ according to Theorem 4. We examined the two cases of whether the initial points $u_i^{(0)} (= v_i^{(0)}) \in \mathbb{R}^2$ $(i = 1, 2)$ in Algorithm 1 are in the interior or the exterior of the convex hull of the three anchors. Figure 1 shows the sensor positions estimated by SFSDP, the SDP relaxation-based method described in Sect. 1, and the proposed method. Note that when we estimated the sensor positions with the proposed method, the randomness of the initial points was varied 10 times. The results were similar to those of Fig. 1b in all cases in which the initial points were in the interior of the convex hull of the anchors. On the other hand, the results were similar to those of either Fig. 1c or d in all cases in which the initial points were in the exterior of the convex hull. Accordingly, only these three cases are included in Fig. 1.

When we used SFSDP, the sensor positions were not estimated correctly (Fig. 1a). For the proposed method, the estimation accuracy depended on the initial points. When the initial points were in the interior of the convex hull of the anchors, the sensor positions were estimated accurately (Fig. 1b). On the other hand, when the initial points were in the exterior of the convex hull, the sensor positions were estimated accurately in some cases (Fig. 1c) but not in others (Fig. 1d).

Of course, because problem (5) examined in this paper is a nonconvex optimization problem, whether the sensor positions can be estimated accurately depends on the initial points. However, as shown in Fig. 1b and c, the proposed method still has the capability to estimate sensor positions accurately even for problems that are not uniquely localizable, although the example here is quite simple. On the other hand, when we use SDP relaxation-based methods, if a given problem is not uniquely localizable, there is no capability for accurate estimation because of the max-rank property of the interior-point method, as described in Sect. 1. Therefore, we can say that the proposed method does inherit the rank constraint.
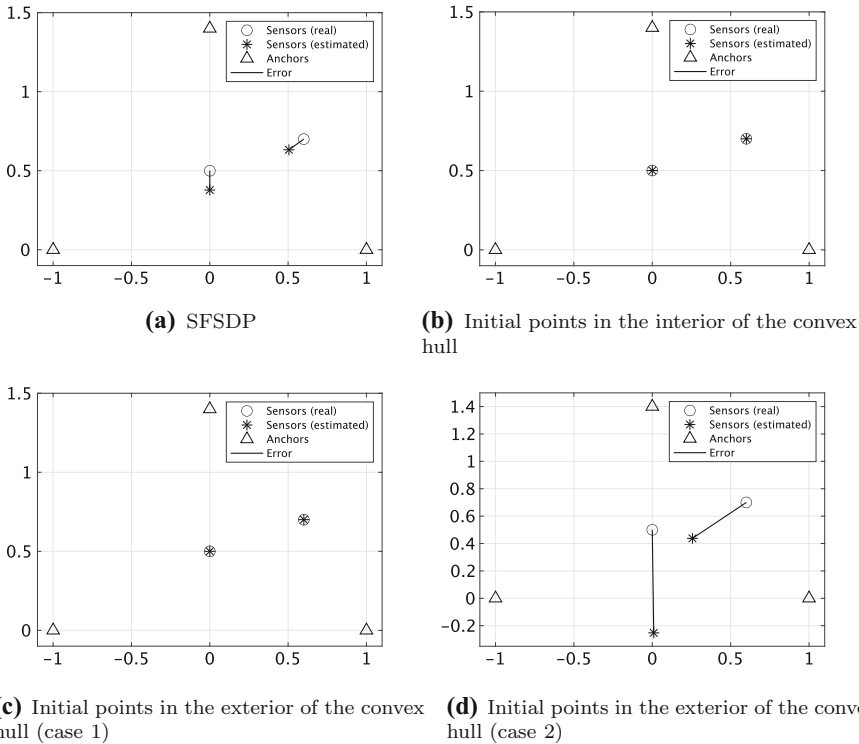
(a) SFSDP

(b) Initial points in the interior of the convex hull

(c) Initial points in the exterior of the convex hull (case 1)

(d) Initial points in the exterior of the convex hull (case 2)

Fig. 1 Estimated sensor positions when the locatable but non-uniquely localizable problem was solved by **a** SFSDP and **b–d** the proposed method

3.2 Comparison of estimation time and accuracy

In this subsection, we quantitatively compare the estimation time and the estimation accuracy of the proposed method with those of existing methods for sensors located in two- or three- dimensional space. The compared methods are SFSDP, which was also used in Sect. 3.1, and NLP-FD, which takes the rank constraint into account as our proposed method does. Although we introduced the methods proposed by Wan et al. [20,21] in Sect. 1, which also account for the rank constraint, we do not compare them here because of their extremely low scalability. In these experiments, $m = 1000, 3000, 5000$, and 20,000 sensors and $n = 0.1m$ anchors were placed randomly in $[0, 1]^d$. E_{ss} and E_{sa} were defined as

$$E_{ss} := \{ij \mid 1 \le i < j \le m, \ \|x_i^{true} - x_j^{true}\|_2 < \rho\},$$
$$E_{sa} := \{ik \mid 1 \le i \le m, \ m + 1 \le k \le m + n, \ \|x_i^{true} - a_k\|_2 < \rho\},$$

where the x_i^{true} $(i = 1, ..., m)$ are the sensors' true positions. In other words, we considered a model in which the distance between two sensors or between a sensor and an anchor is observed if and only if it is less than a radio range ρ (> 0). We set ρ

to 0.1 and $\sqrt{10/m}$ in the case of $d = 2$ and to 0.25 and $\sqrt[3]{15/m}$ in the case of $d = 3$. The measured distances d_{ij} ($ij \in E_{ss}$) and d_{ik} ($ik \in E_{sa}$) were given by

$$d_{ij} = \max\{(1 + \sigma\epsilon_{ij}), 0.1\}\|x_i^{\text{true}} - x_j^{\text{true}}\|_2,$$
$$d_{ik} = \max\{(1 + \sigma\epsilon_{ik}), 0.1\}\|x_i^{\text{true}} - a_k\|_2,$$

where ϵ_{ij}, ϵ_{ik} were selected independently from the standard normal distribution, and σ is a noise factor determining the influence of the error. σ was set to 0, 0.1, and 0.2. As an indicator to measure the estimation accuracy, we used the root-mean-square distance (RMSD), which has been used in many other papers on SNL [8,13,20,22] and is defined as

$$\text{RMSD} := \sqrt{\frac{1}{m}\sum_{i=1}^{m}\|\hat{x}_i - x_i^{\text{true}}\|_2^2},$$

where $\hat{x}_i$ is the estimated position of sensor i. For each set of (m, n, ρ, σ), five different problems of varying randomness were created, and the final results were the averages of five measurements of the estimation time (CPU time) and the estimation accuracy (RMSD).

The initial point $(U^{(0)}, V^{(0)}) \in \mathbb{R}^{d \times m} \times \mathbb{R}^{d \times m}$ in Algorithm 1 was decided similarly to the method in [8]. That is, for each sensor i ($i = 1, \ldots, m$), if it was connected directly to an anchor, then $u_i^{(0)}$ and $v_i^{(0)}$ were set to the coordinates of the anchor nearest to sensor i; otherwise, $u_i^{(0)}$ and $v_i^{(0)}$ were set to

$$\frac{1}{2}\left(\begin{pmatrix}\max_k a_{k1} \\ \vdots \\ \max_k a_{kd}\end{pmatrix} + \begin{pmatrix}\min_k a_{k1} \\ \vdots \\ \min_k a_{kd}\end{pmatrix}\right).$$

The penalty parameter γ was updated dynamically according to Theorem 4 by the following procedure.

Step 1. Let

$$\gamma^{(0)} = 5 \times 10^{-3} \times \sqrt{2f(U^{(0)}, V^{(0)})}\max_{1 \leq i \leq m}\sqrt{4|E_{ss}[i]| + |E_{sa}[i]|}/2.$$

By using $\gamma^{(0)}$ as the penalty parameter, $(U^{(1)}, V^{(1)})$ is calculated by the update rule in the **while** loop of Algorithm 1. Let $\gamma^{(1)} = \gamma^{(0)}/2$. Then, by using $\gamma^{(1)}$ as the penalty parameter, $(U^{(2)}, V^{(2)})$ is also calculated by this update rule. Let $p = 2$.

Step 2. If

$$\frac{f(U^{(p-1)}, V^{(p-1)}) - f(U^{(p)}, V^{(p)})}{f(U^{(p-1)}, V^{(p-1)})}$$
$$\geq \frac{f(U^{(p-2)}, V^{(p-2)}) - f(U^{(p-1)}, V^{(p-1)})}{f(U^{(p-2)}, V^{(p-2)})},$$

then $\gamma^{(p)} = (\gamma^{(p-1)}/\gamma^{(p-2)})\gamma^{(p-1)}$; otherwise, $\gamma^{(p)} = \gamma^{(p-2)}$. By using $\gamma^{(p)}$ as the penalty parameter, $(U^{(p+1)}, V^{(p+1)})$ is calculated by the update rule in the **while** loop of Algorithm 1.

Step 3. If $|f(U^{(p)}, V^{(p)}) - f(U^{(p+1)}, V^{(p+1)})|/f(U^{(p)}, V^{(p)}) < 10^{-2}$ or the overall stopping criterion (line 8 of Algorithm 1) is satisfied, then go to Step 4;[5] otherwise, set $p = p + 1$ and go to Step 2.

Step 4. Let $W := (U^{(p)} + V^{(p)})/2$ and

$$\gamma = \sqrt{2f(W, W)} \max_{1 \leq i \leq m} \sqrt{4|E_{ss}[i]| + |E_{sa}[i]|}/2.$$

Restart Algorithm 1 with W as the initial point and γ as the penalty parameter.

The method of updating γ described above consists of two components. First, in Steps 1–3, γ is updated so that the value of f, which represents the squared error of the squared distances, decreases rather than the penalty term $\|U - V\|_F$. However, if we keep reducing the value of f rather than the penalty term, then the penalty term does not decrease much, and U and V may end up taking very different values from each other. Therefore, in Step 4, we try to reduce the difference between U and V by fixing γ according to Theorem 4.

First, the results for $d = 2$ are given in Table 1,[6] wherein the proposed method is referred to as "BCD." We can see that when the measured distances included no errors ($\sigma = 0$), the estimation time of the proposed method was the lowest in most cases; furthermore, even when the proposed method was not the fastest, its estimation time was almost the same as that of the fastest method. In terms of the estimation accuracy, SFSDP estimated the sensor positions with the best accuracy of all the methods, by an order of magnitude. However, comparing the proposed method and NLP-FD shows that there was no appreciable difference between their estimation accuracies. When the measured distances included errors ($\sigma = 0.1$ and 0.2), the proposed method estimated the sensor positions the most rapidly of all the methods, by an order of magnitude in all cases, and the estimation accuracy was about the same as those of the other two methods.

Next, the results for $d = 3$ are given in Table 2.[7] The results for the three-dimensional scenario were similar to those for the two-dimensional scenario; that is, when the measured distances did not include errors, the estimation time of the proposed method was the lowest in each case. In terms of the estimation accuracy, SFSDP estimated the sensor positions with the highest accuracy of all the methods, by an order of magnitude; however, comparing the proposed method and NLP-FD again shows that there was no appreciable difference between their estimation accuracies. When the measured distances included errors, the estimation time of the proposed method was the lowest in all cases except $(m, n, \rho, \sigma) = (1000, 100, 0.1, 0.2)$, and even in

---

[5] Even if the overall stopping criterion is satisfied at this stage, the entire algorithm does not end but always proceeds to Step 4.

[6] For $m = 1000$, only the case of $\rho = 0.1$ is shown because $\sqrt{10/m} = 0.1$.

[7] For $m = 1000$, the case of $\rho = \sqrt[3]{15/m}$ is omitted because $\sqrt[3]{15/m} = 0.24 \approx 0.25$. An entry of "OOM" means that we could not estimate the sensor positions because of insufficient memory (i.e., "out of memory").

Table 1 Results of numerical experiments for sensors and anchors placed randomly in $[0, 1]^2$

	ρ	σ	CPU time			RMSD		
			BCD	SFSDP	NLP-FD	BCD	SFSDP	NLP-FD
$m = 1000$,	0.1	0	**4.1**	4.2	14.6	3.13e–02	**1.18e–05**	1.27e–02
$n = 100$		0.1	**4.2**	13.7	20.1	3.38e–02	1.92e–02	**1.37e–02**
		0.2	**8.8**	14.5	33.6	3.74e–02	2.94e–02	**1.87e–02**
$m = 3000$,	0.1	0	12.8	**12.7**	44.0	4.02e–04	**4.64e–07**	7.87e–04
$n = 300$		0.1	**1.7**	77.7	81.9	**2.78e–03**	8.78e–03	3.18e–03
		0.2	**2.0**	63.8	120.3	6.96e–03	1.72e–02	**6.02e–03**
	$\sqrt{10/m}$	0	**11.2**	15.3	38.0	1.28e–02	**1.51e–05**	4.03e–03
		0.1	**4.9**	85.9	49.6	1.53e–02	8.47e–03	**5.07e–03**
		0.2	**4.1**	91.3	52.5	1.83e–02	1.39e–02	**7.03e–03**
$m = 5000$,	0.1	0	**18.3**	23.2	54.3	**8.22e–08**	2.00e–07	4.75e–04
$n = 500$		0.1	**2.5**	184.8	162.0	**1.90e–03**	8.26e–03	2.17e–03
		0.2	**2.2**	135.6	223.6	**3.74e–03**	1.62e–02	4.19e–03
	$\sqrt{10/m}$	0	**22.0**	30.2	57.6	9.01e–03	**5.26e–05**	5.81e–03
		0.1	**11.4**	198.7	70.4	1.17e–02	6.56e–03	**5.29e–03**
		0.2	**9.4**	160.0	87.8	1.43e–02	1.04e–02	**5.96e–03**
$m = 20,000$,	0.1	0	**82.7**	176.2	718.2	5.81e–06	**6.50e–07**	3.23e–04
$n = 2000$		0.1	**29.2**	643.5	1300.8	**9.14e–04**	8.33e–03	9.54e–04
		0.2	**26.0**	544.9	1450.2	2.22e–03	1.76e–01	**1.79e–03**
	$\sqrt{10/m}$	0	223.0	357.9	**213.8**	3.35e–03	**6.73e–05**	2.25e–03
		0.1	**129.2**	1527.9	215.2	4.16e–03	3.10e–03	**2.30e–03**
		0.2	**110.6**	1558.2	240.5	5.47e–03	4.86e–03	**2.63e–03**

For each parameter combination, the lowest CPU time and shortest root-mean-square distance (RMSD) are in bold

that case, its estimation time was also almost the same as that of the fastest method (SFSDP). The estimation accuracy of the proposed method was also comparable to those of the other two methods. In addition, for $m =$ 20,000, SFSDP could not estimate the sensor positions because of insufficient memory, while the proposed method could estimate the positions without running out of memory.

Overall, these results for the two- and three-dimensional cases show that the proposed method has practical advantages over the other methods: it can estimate sensor positions faster than those methods can without sacrificing the estimation accuracy, especially when measurement errors are included, and it does not run out of memory even for large-scale SNL problems.

It is interesting to consider why the proposed method and NLP-FD, both of which account for the rank constraint, could not estimate the sensor positions with as much accuracy as SFSDP when there were no measurement errors. The reason was probably because a formulation that accounts for the rank constraint is a nonconvex optimization problem and thus might converge to a stationary point that is not a global optimal solution. In contrast, if a problem is uniquely localizable, then SFSDP can estimate

Table 2 Results of numerical experiments for sensors and anchors placed randomly in $[0, 1]^3$

	ρ	σ	CPU time			RMSD		
			BCD	SFSDP	NLP-FD	BCD	SFSDP	NLP-FD
$m = 1000$,	0.1	0	**5.3**	11.5	34.6	8.19e–03	**2.79e–05**	7.62e–03
$n = 100$		0.1	**18.5**	25.8	32.5	3.21e–02	5.50e–02	**2.77e–02**
		0.2	29.4	**25.4**	42.4	**4.71e–02**	8.18e–02	5.28e–02
$m = 3000$,	0.1	0	**8.6**	24.7	62.3	3.97e–05	**1.22e–06**	1.06e–03
$n = 300$		0.1	**3.3**	100.6	132.0	7.01e–03	3.11e–02	1.09e–02
		0.2	**3.6**	102.0	194.2	**1.47e–02**	5.79e–02	2.45e–02
	$\sqrt[3]{15/m}$	0	**35.6**	61.6	83.1	2.12e–02	**6.25e–05**	4.26e–03
		0.1	**26.8**	185.7	102.5	3.29e–02	3.22e–02	**1.48e–02**
		0.2	**69.5**	158.6	113.8	4.32e–02	5.12e–02	**2.75e–02**
$m = 5000$,	0.1	0	**17.5**	47.5	126.0	5.11e–05	**4.41e–07**	1.09e–03
$n = 500$		0.1	**5.5**	195.5	291.2	**5.50e–03**	2.94e–02	7.95e–03
		0.2	**24.0**	170.3	284.9	**1.19e–02**	5.35e–02	1.38e–02
	$\sqrt[3]{15/m}$	0	**30.4**	199.9	114.0	1.11e–02	**2.13e–04**	7.15e–03
		0.1	**11.9**	554.8	182.8	2.43e–02	2.51e–02	**1.18e–02**
		0.2	**66.5**	522.3	164.7	3.76e–02	4.04e–02	**2.08e–02**
$m = 20,000$,	0.1	0	**128.9**	288.6	1813.7	5.53e–05	**4.24e–07**	6.26e–04
$n = 2000$		0.1	**56.9**	1271.6	2551.9	**2.81e–03**	2.93e–02	3.37e–03
		0.2	**56.9**	1028.0	2679.0	7.59e–03	2.43e–01	**5.61e–03**
	$\sqrt[3]{15/m}$	0	**304.4**	OOM	341.3	**3.93e–03**	OOM	4.00e–03
		0.1	**149.9**	OOM	606.7	1.18e–02	OOM	**6.08e–03**
		0.2	**107.5**	OOM	628.1	2.01e–02	OOM	**1.06e–02**

For each parameter combination, the lowest CPU time and shortest RMSD are in bold

accurate sensor positions because the convergence to the global optimum of a relaxation problem is guaranteed. On the other hand, when measurement errors are included, even if a global minimum solution of the objective function f is obtained and the optimal value is zero, it does not mean that the true sensor positions are estimated, but rather that the positions are estimated incorrectly. In other words, even if the objective function is strictly minimized, it does not necessarily mean that a good estimate of the sensor positions is obtained; thus, when measurement errors are included, estimation accuracy comparable to that of SFSDP can be obtained even with methods that account for the rank constraint and may cause the generated sequence to fall into a stationary point that is not a global optimal solution.

4 Conclusion

In this paper, we proposed a new method that transforms the formulation of problem (3), which appears in SNL, into an unconstrained multiconvex optimization problem (5), to which the block coordinate descent method is applied. We also presented theoretical analyses of the proposed method. First, we showed that each subproblem

that appears in Algorithm 1 can be solved analytically. In addition, we showed that any accumulation point (U^*, V^*) of the sequence $\{(U^{(p)}, V^{(p)})\}_{p=1}^{\infty}$ generated by the proposed algorithm is a stationary point of the objective function of problem (5), and we gave a range of γ such that (U^*, V^*) satisfies $U^* = V^*$. We also pointed out the relationship between the objective function of problem (5) and the augmented Lagrangian. Numerical experiments showed that our method does inherit the rank constraint and that it can estimate sensor positions faster than other methods without sacrificing the estimation accuracy, especially when the measured distances contain errors, and without running out of memory.

The present study suggests three directions for future work. First, Algorithm 1 uses a cycle rule in which the $2m$ subproblems are solved in the order of $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$. However, in the general coordinate descent method, there are other update rules such as a random rule and a greedy rule [15,24]. In SNL, the strategy of updating from variables corresponding to sensors that are connected directly to anchors is also expected to improve the estimation accuracy and time. Therefore, there is still room to consider how the order of solving the $2m$ subproblems affects the estimation time and accuracy. Second, we performed the minimization sequentially with respect to each column of U and V for computational efficiency, but updating some columns of U and V together is also possible, and the manner of block division in applying the block coordinate descent method should be examined further. Finally, the proposed method could be extended to general quadratic SDP problems with a rank constraint.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Aspnes, J., Goldenberg, D., Yang, Y. R.: On the computational complexity of sensor network localization. In: ALGOSENSORS 2004, Lecture Notes in Computer Science. **3121**, 32–44 (2004)
2. Biswas, P., Lian, T.-C., Wang, T.-C., Ye, Y.: Semidefinite programming based algorithms for sensor network localization. ACM Trans. Sens. Netw. **2**(2), 188–220 (2006)
3. Biswas, P., Liang, T.-C., Toh, K.-C., Ye, Y.: Semidefinite programming approaches for sensor network localization with noisy distance measurements. IEEE Trans. Autom. Sci. Eng. **3**(4), 360–371 (2006)

4. Biswas, P., Ye, Y.: Semidefinite programming for ad hoc wireless sensor network localization. In: Third International Symposium on Information Processing in Sensor Networks, 46–54 (2004)
5. Biswas, P., Ye, Y.: A distributed method for solving semidefinite programs arising from ad hoc wireless sensor network localization. In: Hager, W.W., Huang, S.-J., Pardalos, P.M., Prokopyev, O.A. (eds.) Multiscale Optim. Methods Appl., pp. 69–84. Springer, Boston (2006)
6. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Found. Trends Mach. Learn. **3**(1), 1–122 (2010)
7. Burer, S., Monteiro, R.D.C.: A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. Math. Program. Ser. B **95**, 329–357 (2003)
8. Chang, X., Liu, S.: A feasible method for sensor network localization. Optim. Methods Softw. **32**(5), 1113–1131 (2017)
9. Chang, X., Xue, W.: Parallel implementation of feasible direction algorithm for large-scale sensor network location problems. In: IEEE 12th International Conference on Dependable, Autonomic and Secure Computing. 245–251 (2014)
10. Chang, X., Zhu, W., Li, D.: A feasible direction algorithm for solving 3D sensor network localization. J. Beijing Univ. Posts Telecommun. **39**(2), 98–102 (2016)
11. Goldfarb, D., Scheinberg, K.: Interior point trajectories in semidefinite programming. SIAM J. Optim. **8**(4), 871–886 (1998)
12. Gorski, J., Pfeuffer, F., Klamroth, K.: Biconvex sets and optimization with biconvex functions: a survey and extensions. Math. Methods Oper. Res. **66**, 373–407 (2007)
13. Kim, S., Kojima, M., Waki, H.: Exploiting sparsity in SDP relaxation for sensor network localization. SIAM J. Optim. **20**(1), 192–215 (2009)
14. Liang, T.-C., Wang, T.-C., Ye, Y.: 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 (2004)
15. Luo, Z.Q., Tseng, P.: On the convergence of the coordinate descent method for convex differentiable minimization. J. Optim. Theory Appl. **72**(1), 7–35 (1992)
16. Nie, J.: Sum of squares method for sensor network localization. Comput. Optim. Appl. **43**, 151–179 (2009)
17. Nocedal, J., Wright, S.J.: Numerical Optimization, 2nd edn. Springer, NY (2006)
18. So, A.M.-C., Ye, Y.: Theory of semidefinite programming for sensor network localization. Math. Program. Ser. B **109**, 367–384 (2007)
19. Tseng, P.: Second-order cone programming relaxation of sensor network localization. SIAM J. Optim. **18**(1), 156–185 (2007)
20. Wan, C., Jing, G., You, S., Dai, R.: Sensor network localization via alternating rank minimization algorithms. IEEE Trans. Control Netw. Syst. **7**(2), 1040–1051 (2020)
21. Wan, C., You, S., Jing, G., Dai, R.: A distributed algorithm for sensor network localization with limited measurements of relative distance. In: 2019 American Control Conference, 4677–4682 (2019)
22. Wang, Z., Zheng, S., Ye, Y., Boyd, S.: Further relaxations of the semidefinite programming approach to sensor network localization. SIAM J. Optim. **19**(2), 655–673 (2008)
23. Wen, Z., Yin, W.: A feasible method for optimization with orthogonality constraints. Math. Program. Ser. A **142**, 397–434 (2013)
24. Wright, S.J.: Coordinate descent algorithms. Math. Program. Ser. B **151**, 3–34 (2015)
25. Xu, Y., Yin, W.: A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. SIAM J. Imaging Sci. **6**(3), 1758–1789 (2013)