SIAM J. Optim.
Vol. 20, No. 5, pp. 2679–2708

EXPLICIT SENSOR NETWORK LOCALIZATION USING SEMIDEFINITE REPRESENTATIONS AND FACIAL REDUCTIONS*

NATHAN KRISLOCK† and HENRY WOLKOWICZ†

Abstract. The sensor network localization (SNL) problem in embedding dimension r consists of locating the positions of wireless sensors, given only the distances between sensors that are within radio range and the positions of a subset of the sensors (called anchors). Current solution techniques relax this problem to a weighted, nearest, (positive) semidefinite programming (SDP) completion problem by using the linear mapping between Euclidean distance matrices (EDM) and semidefinite matrices. The resulting SDP is solved using primal-dual interior point solvers, yielding an expensive and inexact solution.

This relaxation is highly degenerate in the sense that the feasible set is restricted to a low dimensional face of the SDP cone, implying that the Slater constraint qualification fails. Cliques in the graph of the SNL problem give rise to this degeneracy in the SDP relaxation. In this paper, we take advantage of the absence of the Slater constraint qualification and derive a technique for the SNL problem, with exact data, that explicitly solves the corresponding rank restricted SDP problem. No SDP solvers are used. For randomly generated instances, we are able to efficiently solve many huge instances of this NP-hard problem to high accuracy by finding a representation of the minimal face of the SDP cone that contains the SDP matrix representation of the EDM. The main work of our algorithm consists in repeatedly finding the intersection of subspaces that represent the faces of the SDP cone that correspond to cliques of the SNL problem.

Key words. sensor network localization, Euclidean distance matrix completions, semidefinite programming, loss of the Slater constraint qualification

AMS subject classifications. 90C35, 90C22, 90C26, 90C06

DOI. 10.1137/090759392

1. Introduction.

The sensor network localization problem (SNL) consists of locating the positions of n wireless sensors, $p_i \in \mathbb{R}^r$, $i = 1, \ldots, n$, given only the (squared) Euclidean distances $D_{ij} = \|p_i - p_j\|_2^2$ between sensors that are within a given radio range, $R > 0$, and given the positions of a subset of the sensors, p_i, $i = n - m + 1, \ldots, n$ (called anchors); r is the *embedding dimension* of the problem. Currently, many solution techniques for this problem use a relaxation to a nearest, weighted, semidefinite approximation problem

$$\min_{Y \succeq 0,\, Y \in \Omega} \|W \circ (\mathcal{K}(Y) - D)\|, \tag{1.1}$$

where $Y \succeq 0$ denotes positive semidefiniteness, $Y \in \Omega$ denotes additional linear constraints, $\mathcal{K}$ is a specific linear mapping, and $\circ$ denotes the *Hadamard (elementwise) product.* This approach requires semidefinite programming (SDP) primal-dual interior point (p-d i-p) techniques; see, for example, [2, 3, 5, 8, 9, 12, 23]. This yields an expensive and inexact solution.

The SNL problem is a special case of the Euclidean distance matrix (EDM) completion problem (EDMC). If D is a *partial* EDM, then the completion problem consists of finding the missing elements (squared distances) of D. It is shown in

*Received by the editors May 16, 2009; accepted for publication (in revised form) June 2, 2010; published electronically July 20, 2010. This research was supported by the Natural Sciences Engineering Research Council Canada and a grant from AFOSR.
http://www.siam.org/journals/siopt/20-5/75939.html

†Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada (ngbkrisl@math.uwaterloo.ca, hwolkowicz@uwaterloo.ca).

[13] that there are advantages for handling the SNL problem as an EDMC and ignoring the distinction between the anchors and the other sensors until after the EDMC is solved. In this paper we use this framework and derive an algorithm that locates the sensors by exploiting the structure and implicit degeneracy in the SNL problem. In particular, we solve the SDP problems *explicitly* (exactly) without using any p-d i-p techniques. We do so by repeatedly viewing SNL in three equivalent forms: as a graph realization problem, as an EDMC, and as a rank restricted SDP.

A common approach to solving the EDMC problem is to relax the rank constraint and solve a weighted, nearest, positive semidefinite completion problem (like problem (1.1)) using SDP. The resulting SDP is, implicitly, highly degenerate in the sense that the feasible semidefinite matrices have low rank. In particular, cliques in the graph of the SNL problem reduce the ranks of these feasible semidefinite matrices. This means that the Slater constraint qualification (strict feasibility) implicitly fails for the SDP. Our algorithm is based on exploiting this degeneracy. We characterize the face of the SDP cone that corresponds to a given clique in the graph, thus reducing the size of the SDP problem. Then we characterize the intersection of two faces that correspond to overlapping cliques. This allows us to explicitly *grow/increase* the size of the cliques by repeatedly finding the intersection of subspaces that represent the faces of the SDP cone that correspond to these cliques. Equivalently, this corresponds to completing overlapping blocks of the EDM. In this way, we further reduce the dimension of the faces until we get a completion of the entire EDM. The intersection of the subspaces can be found using a singular value decomposition or by exploiting the special structure of the subspaces. No SDP solver is used. Thus we solve the SDP problem in a finite number of steps, where the work of each step is to find the intersection of two subspaces (or, equivalently, each step is to find the intersection of two faces of the SDP cone).

Though our results hold for general embedding dimension r, our preliminary numerical tests involve sensors with embedding dimensions $r = 2$ and $r = 3$. The sensors are in the region $[0, 1]^r$. There are n sensors, m of which are anchors. The radio range is R units.

1.1. Related work/applications.

The number of applications for distance geometry problems is large and increasing in number and importance. The particular case of SNL has applications to environmental monitoring of geographical regions, as well as tracking of animals and machinery; see, for example, [5, 12]. There have been many algorithms published recently that solve the SNL problem. Many of these involve SDP relaxations and use SDP solvers; see, for example, [5, 6, 7, 8, 9, 13] and more recently [20, 28]. Heuristics are presented in, for example, [11]. SNL is closely related to the EDMC problem; see, for example, [3, 12] and the survey [2].

Carter, Jin, Saunders, and Ye [10] and Jin [19] propose the SpaseLoc heuristic. It is limited to $r = 2$ and uses an SDP solver for small localized subproblems. They then *sew* these subproblems together. So and Ye [25] show that the problem of solving a noiseless SNL that is *uniquely localizable*[1] can be phrased as an SDP and thus can be solved in polynomial time. They also give an efficient criterion for checking whether a given instance has a unique solution for $r = 2$.

Two contributions of this paper are as follows: we do not use iterative p-d i-p techniques to solve the SDP but rather we solve it with a finite number of explicit

[1]An SNL problem is uniquely localizable in dimension r if it has a unique solution in $\mathbb{R}^r$ and it does not have any solution whose affine span is $\mathbb{R}^h$, where $h > r$; see [25].

solutions; we start with local cliques and expand the cliques. Our algorithm has four different basic steps. The first basic step takes two cliques for which the intersection contains at least $r + 1$ nodes and implicitly completes the corresponding EDM to form the union of the cliques. The second step does this when one of the cliques is a single element. Therefore, this provides an extension of the algorithm in [14], where Eren et al have shown that the family of *trilateration graphs*[2] admits a polynomial time algorithm for computing a realization in a required dimension. Our algorithm repeatedly finds explicit solutions of an SDP. Other examples of finding explicit solutions of an SDP are given in [27, 29].

The SNL problem with given embedding dimension r is NP-hard [17, 18, 24]. However, from our numerical tests it appears that random problems that have a unique solution can be solved very efficiently. This phenomenon fits into the results in [4, 15].

1.2. Outline.

We continue in section 1.3 to present notation and results that will be used. The facial reduction process is based on the results in section 2. The single clique facial reduction is given in Theorem 2.3; the reduction of two overlapping cliques in the rigid and nonrigid cases is presented in Theorems 2.10 and 2.14, respectively; absorbing nodes into cliques in the rigid and nonrigid cases is given in Corollaries 2.17 and 2.18, respectively. These results are then used in our algorithm in section 3. The numerical tests appear in sections 3.1 and 3.2. Our concluding remarks are given in section 4.

1.3. Preliminaries.

We work in the vector space of *real symmetric $k \times k$ matrices*, $\mathcal{S}^k$, equipped with the *trace inner product*, $\langle A, B\rangle = \operatorname{trace}(AB)$. We let $\mathcal{S}^k_+$ and $\mathcal{S}^k_{++}$ denote the cone of positive semidefinite and positive definite matrices, respectively; $A \succeq B$ and $A \succ B$ denote the Löwner partial order, $A - B \in \mathcal{S}^k_+$ and $A - B \in \mathcal{S}^k_{++}$, respectively; e denotes the vector of ones of appropriate dimension; $\mathcal{R}(\mathcal{L})$ and $\mathcal{N}(\mathcal{L})$ denote the range space and null space of the linear transformation $\mathcal{L}$, respectively; $\operatorname{cone}(S)$ denotes the convex cone generated by the set S. We use the MATLAB notation $1{:}n = \{1, \ldots, n\}$.

A subset $F \subseteq K$ is a *face of the cone* K, denoted $F \trianglelefteq K$, if

$$\left(x, y \in K,\ \frac{1}{2}(x + y) \in F\right) \implies (\operatorname{cone}\{x, y\} \subseteq F).$$

If $F \trianglelefteq K$ but is not equal to K, we write $F \triangleleft K$. If $\{0\} \neq F \triangleleft K$, then F is a *proper face* of K. For $S \subseteq K$, we let $\operatorname{face}(S)$ denote the smallest face of K that contains S. A face $F \trianglelefteq K$ is an *exposed face* if it is the intersection of K with a hyperplane. The cone K is *facially exposed* if every face $F \trianglelefteq K$ is exposed.

The cone $\mathcal{S}^n_+$ is facially exposed. Moreover, each face $F \trianglelefteq \mathcal{S}^n_+$ is determined by the range of any matrix S in the relative interior of the face, $S \in \operatorname{relint} F$: if $S = U\Gamma U^T$ is the compact spectral decomposition of S with the diagonal matrix of eigenvalues $\Gamma \in \mathcal{S}^t_{++}$, then (e.g., [22])

$$F = U\mathcal{S}^t_+ U^T. \tag{1.2}$$

[2]A graph is a trilateration graph in dimension r if there exists an ordering of the nodes $1, \ldots, r + 1, r + 2, \ldots, n$ such that the first $r + 1$ nodes form a clique and each node $j > r + 1$ has at least $r + 1$ edges to nodes earlier in the sequence.

A matrix $D = (D_{ij}) \in \mathcal{S}^n$ with nonnegative elements and zero diagonal is called a *predistance matrix*. In addition, if there exist points $p_1, \ldots, p_n \in \mathbb{R}^r$ such that

$$D_{ij} = \|p_i - p_j\|_2^2, \quad i, j = 1, \ldots, n, \tag{1.3}$$

then D is called a *Euclidean distance matrix*, denoted EDM. Note that we work with *squared distances*. The smallest value of r such that (1.3) holds is called the *embedding dimension* of D. Throughout this paper, we assume that r is given and *fixed*. The set of EDM matrices forms a closed convex cone in $\mathcal{S}^n$, denoted $\mathcal{E}^n$. If we are given an $n \times n$ partial EDM D_p, let $\mathcal{G} = (N, E, \omega)$ be the corresponding simple graph on the nodes $N = 1\!:\!n$ whose edges E correspond to the known entries of D_p with $(D_p)_{ij} = \omega_{ij}^2$ for all $(i, j) \in E$.

Definition 1.1. *For $Y \in \mathcal{S}^n$ and $\alpha \subseteq 1\!:\!n$, we let $Y[\alpha]$ denote the corresponding* principal submatrix *formed from the rows and columns with indices α. If, in addition, $|\alpha| = k$ and $\bar{Y} \in \mathcal{S}^k$ is given, then we define*

$$\mathcal{S}^n(\alpha, \bar{Y}) := \left\{ Y \in \mathcal{S}^n : Y[\alpha] = \bar{Y} \right\}, \quad \mathcal{S}_+^n(\alpha, \bar{Y}) := \left\{ Y \in \mathcal{S}_+^n : Y[\alpha] = \bar{Y} \right\},$$

that is, the subset of matrices $Y \in \mathcal{S}^n$ ($Y \in \mathcal{S}_+^n$) with principal submatrix $Y[\alpha]$ fixed to $\bar{Y}$. For example, the subset of matrices in $\mathcal{S}^n$ with the top left $k \times k$ block fixed is

$$\mathcal{S}^n(1\!:\!k, \bar{Y}) = \left\{ Y \in \mathcal{S}^n : Y = \left[\begin{array}{c|c} \bar{Y} & \cdot \\ \hline \cdot & \cdot \end{array}\right] \right\}. \tag{1.4}$$

A clique $\gamma \subseteq 1\!:\!n$ in the graph $\mathcal{G}$ corresponds to a subset of sensors for which the distances $\omega_{ij} = \|p_i - p_j\|_2$ are known for all $i, j \in \gamma$; equivalently, the clique corresponds to the principal submatrix $D_p[\gamma]$ of the partial EDM matrix D_p, where all the elements of $D_p[\gamma]$ are known.

Suppose that we are given a subset of the (squared) distances from (1.3) in the form of a partial EDM D_p. The *EDM completion problem* consists of finding the missing entries of D_p to complete the EDM; see Figure 1.1. This completion problem can be solved by finding a set of points $p_1, \ldots, p_n \in \mathbb{R}^r$ satisfying (1.3), where r is the embedding dimension of the partial EDM, D_p. This problem corresponds to the graph realizability problem with dimension r, which is the problem of finding positions in $\mathbb{R}^r$ for the vertices of a graph such that the interdistances of these positions satisfy the given edge lengths of the graph.

Let $Y \in \mathcal{M}^n$ be an $n \times n$ real matrix and $y \in \mathbb{R}^n$ be a vector. We let $\operatorname{diag}(Y)$ denote the vector in $\mathbb{R}^n$ formed from the diagonal of Y, and we let $\operatorname{Diag}(y)$ denote the diagonal matrix in $\mathcal{M}^n$ with the vector y along its diagonal. Note that diag and Diag are the adjoint linear transformations of each other: $\operatorname{Diag} = \operatorname{diag}^*$. The operator offDiag can then be defined as $\operatorname{offDiag}(Y) := Y - \operatorname{Diag}(\operatorname{diag} Y)$. For

$$P = \begin{bmatrix} p_1^T \\ p_2^T \\ \vdots \\ p_n^T \end{bmatrix} \in \mathcal{M}^{n \times r},$$

where p_j, $j = 1, \ldots, n$, are the points used in (1.3), let $Y := PP^T$, and let D be the corresponding EDM satisfying (1.3). Defining the linear operators $\mathcal{K}$ and $\mathcal{D}_e$ on $\mathcal{S}^n$

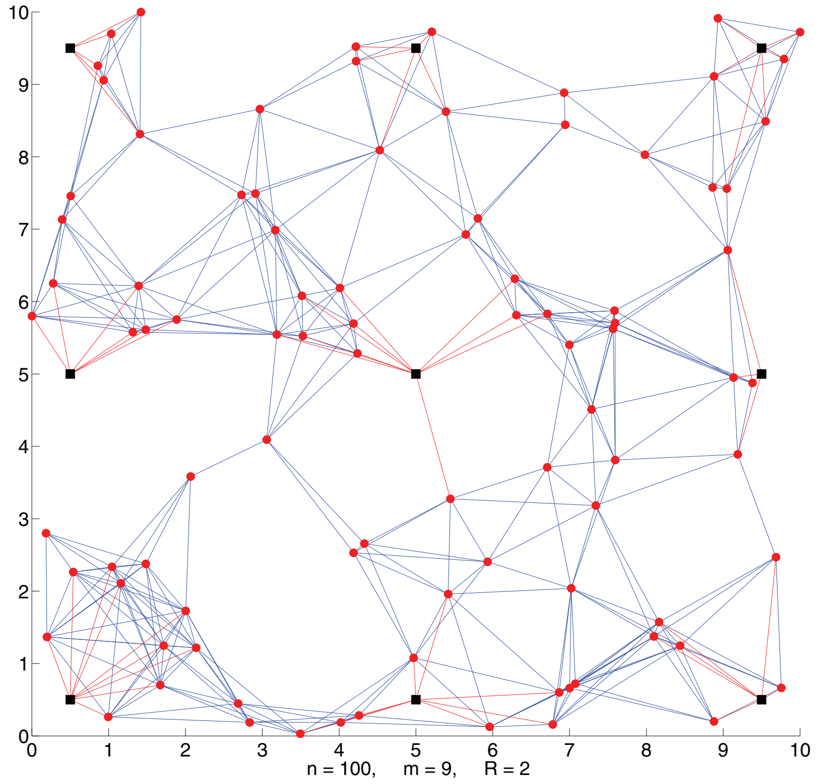

FIG. 1.1. *Graph of partial EDM with sensors* $\circ$ *and anchors* $\blacksquare$.

as follows, we see that

$$\begin{array}{rcl} \mathcal{K}(Y) & := & \mathcal{D}_e(Y) - 2Y \\ & := & \operatorname{diag}(Y)\, e^T + e \operatorname{diag}(Y)^T - 2Y \\ & = & \left(p_i^T p_i + p_j^T p_j - 2p_i^T p_j\right)_{i,j=1}^n \\ & = & \left(\|p_i - p_j\|_2^2\right)_{i,j=1}^n \\ & = & D. \end{array} \tag{1.5}$$

That is, $\mathcal{K}$ maps the positive semidefinite matrix Y onto the EDM D. More generally, we can allow for a general vector v to replace e and define $\mathcal{D}_v(Y) := \operatorname{diag}(Y)\, v^T + v \operatorname{diag}(Y)^T$. By abuse of notation, we also allow $\mathcal{D}_v$ to act on a vector; that is, $\mathcal{D}_v(y) := yv^T + vy^T$. The adjoint of $\mathcal{K}$ is

$$\mathcal{K}^*(D) \quad = \quad 2(\operatorname{Diag}(De) - D). \tag{1.6}$$

The linear operator $\mathcal{K}$ is a one-to-one and onto mapping between the *centered* and *hollow* subspaces of $\mathcal{S}^n$, which are defined as

$$\begin{array}{rcll} \mathcal{S}_C & := & \{Y \in \mathcal{S}^n : Ye = 0\} & \text{(zero row sums)}, \\ \mathcal{S}_H & := & \{D \in \mathcal{S}^n : \operatorname{diag}(D) = 0\} & = \mathcal{R}(\operatorname{offDiag}). \end{array} \tag{1.7}$$

Let $J := I - \frac{1}{n}ee^T$ denote the orthogonal projection onto the subspace $\{e\}^\perp$ and define the linear operator $\mathcal{T}(D) := -\frac{1}{2}J \operatorname{offDiag}(D)J$. Then we have the following relationships.

PROPOSITION 1.2 (see [1]). *The linear operator $\mathcal{T}$ is the generalized inverse of the linear operator $\mathcal{K}$; that is, $\mathcal{K}^\dagger = \mathcal{T}$. Moreover:*

$$\begin{aligned} \mathcal{R}(\mathcal{K}) &= \mathcal{S}_H; & \mathcal{N}(\mathcal{K}) &= \mathcal{R}(\mathcal{D}_e); \\ \mathcal{R}(\mathcal{K}^*) = \mathcal{R}(\mathcal{T}) &= \mathcal{S}_C; & \mathcal{N}(\mathcal{K}^*) = \mathcal{N}(\mathcal{T}) &= \mathcal{R}(\mathrm{Diag}); \end{aligned} \tag{1.8}$$

$$\mathcal{S}^n = \mathcal{S}_H \oplus \mathcal{R}(\mathrm{Diag}) = \mathcal{S}_C \oplus \mathcal{R}(\mathcal{D}_e). \tag{1.9}$$

THEOREM 1.3 (see [1]). *The linear operators $\mathcal{T}$ and $\mathcal{K}$ are one-to-one and onto mappings between the cone $\mathcal{E}^n \subset \mathcal{S}_H$ and the face of the semidefinite cone $\mathcal{S}^n_+ \cap \mathcal{S}_C$. That is,*

$$\mathcal{T}(\mathcal{E}^n) = \mathcal{S}^n_+ \cap \mathcal{S}_C \quad \text{and} \quad \mathcal{K}(\mathcal{S}^n_+ \cap \mathcal{S}_C) = \mathcal{E}^n.$$

Remark 1.4. $D \in \mathcal{E}^n$ has embedding dimension r if and only if $\mathcal{K}^\dagger(D) \succeq 0$ and $\operatorname{rank}(\mathcal{K}^\dagger(D)) = r$. In addition, we get $\mathcal{K}^\dagger(D)e = 0$. Therefore, we can factor $\mathcal{K}^\dagger(D) = PP^T$, for some $P \in \mathcal{M}^{n\times r}$, to recover the (centered) sensors in $\mathbb{R}^r$ from the rows in P. Note that rotations of the points in the rows of P do not change the value $Y = PP^T$ since $PP^T = PQ^TQP$ if Q is orthogonal. However, the nullspace of $\mathcal{K}$ is related to translations of the points in P. Let $D \in \mathcal{E}^n$ with embedding dimension r, and let $Y := \mathcal{K}^\dagger(D)$ have full rank factorization $Y = PP^T$ with $P \in \mathcal{M}^{n\times r}$. Then the translation of points in the rows of P to $\bar{P} := P + ew^T$, for some $w \in \mathbb{R}^r$, results in $\bar{Y} := \bar{P}\bar{P}^T = Y + \mathcal{D}_e(y)$ with $y := Pw + \frac{w^Tw}{2}e$ and $\mathcal{K}(\bar{Y}) = \mathcal{K}(Y) = D$ since $\mathcal{D}_e(y) \in \mathcal{N}(\mathcal{K})$. Note that $\mathcal{R}(Y) = \mathcal{R}(P)$; therefore, $y = Pw + \frac{w^Tw}{2}e \in \mathcal{R}(Y) + \operatorname{cone}\{e\}$, as we will also see in more generality in Lemma 2.1 below.

Let $D_p \in \mathcal{S}^n$ be a *partial* EDM with embedding dimension r, and let $W \in \mathcal{S}^n$ be the 0–1 matrix corresponding to the known entries of D_p. One can use the substitution $D = \mathcal{K}(Y)$, where $Y \in \mathcal{S}^n_+ \cap \mathcal{S}_C$, in the EDM completion problem

$$\begin{array}{cc} \text{Find} & D \in \mathcal{E}^n \\ \text{such that} & W \circ D = W \circ D_p \end{array}$$

to obtain the SDP relaxation

$$\begin{array}{cc} \text{Find} & Y \in \mathcal{S}^n_+ \cap \mathcal{S}_C \\ \text{such that} & W \circ \mathcal{K}(Y) = W \circ D_p \end{array}.$$

This relaxation does not restrict the rank of Y and may yield a solution with an embedding dimension that is too large if $\operatorname{rank}(Y) > r$. Moreover, solving SDP problems with rank restrictions is NP-hard. However, we work on faces of $\mathcal{S}^n_+$ described by $U\mathcal{S}^t_+U^T$ with $t \le n$. In order to find the face with the smallest dimension t, we must have the correct knowledge of the matrix U. In this paper, we obtain information on U using the cliques in the graph of the partial EDM.

2. Semidefinite facial reduction.

We now present several techniques for reducing an EDM completion problem when one or more (possibly intersecting) cliques are known. This extends the reduction using disjoint cliques presented in [13]. In each case, we take advantage of the loss of the Slater constraint qualification and project the problem to a lower dimensional SDP cone.

We first need the following two technical lemmas that exploit the structure of the SDP cone.

LEMMA 2.1. *Let $B \in \mathcal{S}^n$, $Bv = 0$, $v \neq 0$, $y \in \mathbb{R}^n$, and $\bar{Y} := B + \mathcal{D}_v(y)$. If $\bar{Y} \succeq 0$, then*

$$y \in \mathcal{R}(B) + \operatorname{cone}\{v\}.$$

Proof. First we will show that $y \in \mathcal{R}(B) + \operatorname{span}\{v\} = \mathcal{R}([B \quad v])$. If this is not the case, then y can be written as the orthogonal decomposition

$$y = Bu + \beta v + \bar{y},$$

where $0 \neq \bar{y} \in \mathcal{R}([B \quad v])^\perp = \mathcal{N}([B \quad v]^T)$. Note that $\bar{y}$ satisfies $B\bar{y} = 0$ and $v^T\bar{y} = 0$. To get a contradiction with the assumption that $\bar{Y} \succeq 0$, we let

$$z := \frac{1}{2}\frac{v}{\|v\|^2} - (1 + |\beta|)\frac{\bar{y}}{\|\bar{y}\|^2}$$

and observe that $Bz = 0$ and $v^T z = 1/2$. Then

$$\begin{array}{rcl} z^T\bar{Y}z & = & z^T\mathcal{D}_v(y)z \\ & = & z^T\left(yv^T + vy^T\right)z \\ & = & y^T z \\ & = & \frac{1}{2}\beta + \bar{y}^T z \\ & < & \frac{1}{2}(1+|\beta|) + \bar{y}^T z \\ & = & -\frac{1}{2}(1+|\beta|) \\ & < & 0, \end{array}$$

which gives us the desired contradiction. Therefore, $y \in \mathcal{R}(B) + \operatorname{span}\{v\}$, so to show that $y \in \mathcal{R}(B) + \operatorname{cone}\{v\}$, we need to show only that if $y = Bu + \beta v$, then $\beta \geq 0$. First note that $v^T y = \beta v^T v$. Then

$$\begin{array}{rcl} v^T\bar{Y}v & = & v^T\left(yv^T + vy^T\right)v \\ & = & 2v^T y v^T v \\ & = & 2\beta(v^T v)^2. \end{array}$$

Since $\bar{Y} \succeq 0$, we have $2\beta(v^Tv)^2 \geq 0$. This implies that $\beta \geq 0$ since $v \neq 0$. □

If $\bar{Y} \in \mathcal{S}^k_+$, then we can use the minimal face of $\mathcal{S}^k_+$ containing $\bar{Y}$ to find an expression for the minimal face of $\mathcal{S}^n_+$ that contains $\mathcal{S}^n_+(1\!:\!k, \bar{Y})$.

LEMMA 2.2. *Let $\bar{U} \in \mathcal{M}^{k\times t}$ with $\bar{U}^T\bar{U} = I_t$. If $\operatorname{face}\{\bar{Y}\} \trianglelefteq \bar{U}\mathcal{S}^t_+\bar{U}^T$, then*

$$\operatorname{face}\mathcal{S}^n_+(1\!:\!k,\bar{Y}) \trianglelefteq \begin{bmatrix}\bar{U} & 0\\ 0 & I_{n-k}\end{bmatrix}\mathcal{S}^{n-k+t}_+\begin{bmatrix}\bar{U} & 0\\ 0 & I_{n-k}\end{bmatrix}^T. \tag{2.1}$$

Furthermore, if $\operatorname{face}\{\bar{Y}\} = \bar{U}\mathcal{S}^t_+\bar{U}^T$, then

$$\operatorname{face}\mathcal{S}^n_+(1\!:\!k,\bar{Y}) = \begin{bmatrix}\bar{U} & 0\\ 0 & I_{n-k}\end{bmatrix}\mathcal{S}^{n-k+t}_+\begin{bmatrix}\bar{U} & 0\\ 0 & I_{n-k}\end{bmatrix}^T. \tag{2.2}$$

Proof. Since $\bar{Y} \in \bar{U}\mathcal{S}^t_+\bar{U}^T$, then $\bar{Y} = \bar{U}S\bar{U}^T$ for some $S \in \mathcal{S}^t_+$. Let $Y \in \mathcal{S}^n_+(1\!:\!k,\bar{Y})$, and choose $\bar{V}$ so that $\begin{bmatrix}\bar{U} & \bar{V}\end{bmatrix}$ is an orthogonal matrix. Then, with Y blocked appropriately, we evaluate the congruence

$$0 \preceq \begin{bmatrix}\bar{V} & 0\\ 0 & I_{n-k}\end{bmatrix}^T Y \begin{bmatrix}\bar{V} & 0\\ 0 & I_{n-k}\end{bmatrix} = \begin{bmatrix}0 & \bar{V}^TY_{21}^T\\ Y_{21}\bar{V} & Y_{22}\end{bmatrix} = \begin{bmatrix}0 & 0\\ 0 & Y_{22}\end{bmatrix}.$$

Therefore, $Y \succeq 0$ implies that $\bar{V}^T Y_{21}^T = 0$. Since $\mathcal{N}(\bar{V}^T) = \mathcal{R}(\bar{U})$, we get $Y_{21}^T = \bar{U}X$ for some X. Therefore, we can write

$$Y = \begin{bmatrix} \bar{U} & 0 \\ 0 & I_{n-k} \end{bmatrix} \begin{bmatrix} S & X \\ X^T & Y_{22} \end{bmatrix} \begin{bmatrix} \bar{U} & 0 \\ 0 & I_{n-k} \end{bmatrix}^T .$$

This implies that face $\mathcal{S}_+^n(1\!:\!k, \bar{Y}) \trianglelefteq U\mathcal{S}_+^{n-k+t}U^T$, where

$$U := \begin{bmatrix} \bar{U} & 0 \\ 0 & I_{n-k} \end{bmatrix} .$$

This proves (2.1). To prove (2.2), note that if face $\{\bar{Y}\} = \bar{U}\mathcal{S}_+^t \bar{U}^T$, then $\bar{Y} \in \operatorname{relint}\left(\bar{U}\mathcal{S}_+^t \bar{U}^T\right)$, so $\bar{Y} = \bar{U}S\bar{U}^T$ for some $S \in \mathcal{S}_{++}^t$. Letting

$$\hat{Y} := \begin{bmatrix} \bar{U} & 0 \\ 0 & I_{n-k} \end{bmatrix} \begin{bmatrix} S & 0 \\ 0 & I_{n-k} \end{bmatrix} \begin{bmatrix} \bar{U} & 0 \\ 0 & I_{n-k} \end{bmatrix}^T ,$$

we see that $\hat{Y} \in \mathcal{S}_+^n(1 : k, \bar{Y}) \cap \operatorname{relint}\left(U\mathcal{S}_+^{n-k+t}U^T\right)$. This implies that there is no smaller face of $\mathcal{S}_+^n$ containing $\mathcal{S}_+^n(1\!:\!k, \bar{Y})$, completing the proof. $\square$

2.1. Single clique facial reduction. If the principal submatrix $\bar{D} \in \mathcal{E}^k$ is given for index set $\alpha \subseteq 1\!:\!n$ with $|\alpha| = k$, we define

$$\mathcal{E}^n(\alpha, \bar{D}) := \left\{ D \in \mathcal{E}^n : D[\alpha] = \bar{D} \right\}. \tag{2.3}$$

Similarly, the subset of matrices in $\mathcal{E}^n$ with the top left $k \times k$ block fixed is

$$\mathcal{E}^n(1\!:\!k, \bar{D}) = \left\{ D \in \mathcal{E}^n : D = \left[\begin{array}{c|c} \bar{D} & \cdot \\ \hline \cdot & \cdot \end{array}\right] \right\}. \tag{2.4}$$

A fixed principal submatrix $\bar{D}$ in a partial EDM D corresponds to a clique α in the graph $\mathcal{G}$ of the partial EDM D. Given such a fixed clique defined by the submatrix $\bar{D}$, the following theorem shows that the following set, containing the feasible set of the corresponding SDP relaxation,

$$\left\{ Y \in \mathcal{S}_+^n \cap \mathcal{S}_C : \mathcal{K}(Y[\alpha]) = \bar{D} \right\} = \mathcal{K}^\dagger \left(\mathcal{E}^n(\alpha, \bar{D}) \right),$$

is contained in a proper face of $\mathcal{S}_+^n$. This means that the Slater constraint qualification (strict feasibility) fails, and we can reduce the size of the SDP problem; see [13]. We expand on this and find an explicit expression for face $\mathcal{K}^\dagger(\mathcal{E}^n(\alpha, \bar{D}))$ in Theorem 2.3. For simplicity, here and below, we often work with ordered sets of integers for the two cliques. This simplification can always be obtained by a permutation of the indices of the sensors.

THEOREM 2.3. *Let $D \in \mathcal{E}^n$ with embedding dimension r. Let $\bar{D} := D[1\!:\!k] \in \mathcal{E}^k$ with embedding dimension t, and let $B := \mathcal{K}^\dagger(\bar{D}) = \bar{U}_B S \bar{U}_B^T$, where $\bar{U}_B \in \mathcal{M}^{k \times t}$, $\bar{U}_B^T \bar{U}_B = I_t$, and $S \in \mathcal{S}_{++}^t$. Furthermore, let $U_B := [\bar{U}_B \quad \frac{1}{\sqrt{k}}e] \in \mathcal{M}^{k \times (t+1)}$ and $U := \left[\begin{smallmatrix} U_B & 0 \\ 0 & I_{n-k} \end{smallmatrix}\right]$, and let $[V \quad \frac{U^T e}{\|U^T e\|}] \in \mathcal{M}^{n-k+t+1}$ be orthogonal. Then*

$$\operatorname{face} \mathcal{K}^\dagger \left(\mathcal{E}^n(1\!:\!k, \bar{D}) \right) = \left(U\mathcal{S}_+^{n-k+t+1}U^T \right) \cap \mathcal{S}_C = (UV)\mathcal{S}_+^{n-k+t}(UV)^T. \tag{2.5}$$

Proof. Let $Y \in \mathcal{K}^\dagger(\mathcal{E}^n(1 : k, \bar{D}))$ and $\bar{Y} := Y[1 : k]$. Then there exists $D \in \mathcal{E}^n(1 : k, \bar{D})$ such that $Y = \mathcal{K}^\dagger(D)$, implying that $\mathcal{K}(Y) = D$ and that $\mathcal{K}(\bar{Y}) =$

$\bar{D} = \mathcal{K}(B)$. Thus, $\bar{Y} \in B + \mathcal{N}(\mathcal{K}) = B + \mathcal{R}(\mathcal{D}_e)$, where the last equality follows from Proposition 1.2. This implies that $\bar{Y} = B + \mathcal{D}_e(y)$ for some $y \in \mathbb{R}^k$. From Theorem 1.3, we get $\bar{Y} \succeq 0$ and $Be = 0$. Therefore, Lemma 2.1 implies that $y = Bu + \beta e$ for some $u \in \mathbb{R}^k$ and $\beta \geq 0$. This further implies

$$\bar{Y} = B + Bue^T + eu^T B + 2\beta ee^T.$$

From this expression for $\bar{Y}$, we can see that $\mathcal{R}(\bar{Y}) \subseteq \mathcal{R}([B \quad e]) = \mathcal{R}(U_B)$, where the last equality follows from the fact that $Be = 0$. Therefore, $\bar{Y} \in U_B \mathcal{S}_+^{t+1} U_B^T$, implying, by Lemma 2.2, that face $\mathcal{S}_+^n(1:k, \bar{Y}) \trianglelefteq U\mathcal{S}_+^{n-k+t+1}U^T$. Since $Y \in \mathcal{S}_+^n(1:k, \bar{Y})$ and $Ye = 0$, we have that $Y \in \left(U\mathcal{S}_+^{n-k+t+1}U^T\right) \cap \mathcal{S}_C$. Therefore, face $\mathcal{K}^\dagger(\mathcal{E}^n(1:k, \bar{D})) \trianglelefteq \left(U\mathcal{S}_+^{n-k+t+1}U^T\right) \cap \mathcal{S}_C$. Since $V^T U^T e = 0$, we have that

$$\left(U\mathcal{S}_+^{n-k+t+1}U^T\right) \cap \mathcal{S}_C = UV\mathcal{S}_+^{n-k+t}V^T U^T. \tag{2.6}$$

To show that face $\mathcal{K}^\dagger(\mathcal{E}^n(1:k, \bar{D})) = \left(U\mathcal{S}_+^{n-k+t+1}U^T\right) \cap \mathcal{S}_C$, we need to find

$$\hat{Y} = UZU^T \in \mathcal{K}^\dagger\left(\mathcal{E}^n(1:k, \bar{D})\right) \quad \text{with } \operatorname{rank}(\hat{Y}) = n-k+t,\ \hat{Y}e = 0,\ Z \in \mathcal{S}_+^{n-k+t+1}. \tag{2.7}$$

To accomplish this, we let $T_1 = \left[\begin{smallmatrix} S & 0 \\ 0 & 1 \end{smallmatrix}\right]$. Then $T_1 \succ 0$, and

$$B + \frac{1}{k}ee^T = U_B T_1 U_B^T = \bar{P}\bar{P}^T, \quad \text{where } \bar{P} := U_B T_1^{1/2} \in \mathcal{M}^{k \times (t+1)}.$$

Let

$$P := \left[\begin{array}{c|c} \bar{P} & 0 \\ \hline 0 & I_{n-k-1} \\ -e^T\bar{P} & -e^T \end{array}\right] \in \mathcal{M}^{n \times (n-k+t)}.$$

Since $\bar{P}$ has full-column rank, we see that P also has full-column rank. Moreover, $P^T e = 0$. Therefore,

$$\hat{Y} := PP^T = \left[\begin{array}{c|cc} \bar{P}\bar{P}^T & 0 & -e \\ \hline 0 & I_{n-k-1} & -e \\ -e^T & -e^T & n-1 \end{array}\right] \in \mathcal{S}_+^n$$

satisfies $\hat{Y}e = 0$ and $\operatorname{rank}(\hat{Y}) = n-k+t$. Furthermore, we have that $\hat{Y} = UZU^T$, where

$$Z = \left[\begin{array}{cc|cc} S & 0 & 0 & 0 \\ 0 & 1 & 0 & -\sqrt{k} \\ \hline 0 & 0 & I_{n-k-1} & -e \\ 0 & -\sqrt{k} & -e^T & n-1 \end{array}\right] \in \mathcal{S}^{n-k+t+1}.$$

Note that we can also write Z as

$$Z = \begin{bmatrix} S & 0 \\ 0 & T \end{bmatrix} \in \mathcal{S}^{n-k+t+1},$$

where

$$T := \begin{bmatrix} 1 & 0 & -\sqrt{k} \\ 0 & I_{n-k-1} & -e \\ -\sqrt{k} & -e^T & n-1 \end{bmatrix} \in \mathcal{S}^{n-k+1}.$$

The eigenvalues of T are 0, 1, and n, with multiplicities 1, $n-k-1$, and 1, respectively. Therefore, $\operatorname{rank}(T) = n-k$, which implies that $\operatorname{rank}(Z) = n-k+t$ and $Z \succeq 0$.

Letting $\hat{D} := \mathcal{K}(\hat{Y})$, we have that $\hat{D} \in \mathcal{E}^n(1\!:\!k, \bar{D})$ since

$$\hat{D}[1\!:\!k] = \mathcal{K}(\hat{Y}[1\!:\!k]) = \mathcal{K}(\bar{P}\bar{P}^T) = \mathcal{K}\left(B + \frac{1}{k}ee^T\right) = \mathcal{K}(B) = \bar{D}.$$

Therefore, $\hat{Y}$ satisfies (2.7), completing the proof. $\square$

Remark 2.4. Theorem 2.3 provides a reduction in the dimension of the EDM completion problem. Initially, our problem consists of finding $Y \in \mathcal{S}^n_+ \cap \mathcal{S}_C$ such that the constraint

$$\mathcal{K}(Y[\alpha]) = D[\alpha], \quad \alpha = 1\!:\!k$$

holds. After the reduction, we have the smaller dimensional variable $Z \in \mathcal{S}^{n-k+t}_+$; by construction $Y := (UV)Z(UV)^T$ will automatically satisfy the above constraints. This is a reduction of $k-t-1 = (n-1)-(n-k+t)$ in the dimension of the matrix variable. The addition of the vector e to the range of B, $U_B := [\bar{U}_B \quad \frac{1}{\sqrt{k}}e]$, has a geometric interpretation. If $B = PP^T$, $P \in \mathcal{M}^{k\times t}$, then the rows of P provide *centered* positions for the k sensors in the clique α. However, these sensors are not necessarily centered once they are combined with the remaining $n-k$ sensors. Therefore, we have to allow for translations, e.g., to $P + ev^T$ for some v. The multiplication $(P+ev^T)(P+ev^T)^T = PP^T + Pve^T + ev^TP^T + ev^Tve^T$ is included in the set of matrices that we get after adding e to the range of B. Note that $Pve^T + ev^TP^T + ev^Tve^T = \mathcal{D}_e(y)$ for $y = Pv + \frac{1}{2}ev^Tv$.

The special case $k=1$ is of interest.

COROLLARY 2.5. *Suppose that the hypotheses of Theorem* 2.3 *hold but that* $k=1$ *and* $\bar{D} = 0$. *Then* $U_B = 1$, $U = I_n$, *and*

$$\text{face}\,\mathcal{K}^\dagger\left(\mathcal{E}^n(1\!:\!k, \bar{D})\right) = \text{face}\,\mathcal{K}^\dagger\left(\mathcal{E}^n\right) = \mathcal{S}^n_+ \cap \mathcal{S}_C = V\mathcal{S}^{n-1}_+V^T, \tag{2.8}$$

where $[V \quad \frac{1}{\sqrt{n}}e] \in \mathcal{M}^n$ *is orthogonal.*

Proof. Since $k=1$, necessarily we get $t=0$ and we can set $U_B = 1$. $\square$

2.1.1. Disjoint cliques facial reduction. Theorem 2.3 can be easily extended to two or more disjoint cliques; see also [13].

COROLLARY 2.6. *Let* $D \in \mathcal{E}^n$ *with embedding dimension* r. *Let* $k_0 := 1 < k_1 < \ldots < k_l \le n$. *For* $i = 1, \ldots, l$, *let* $\bar{D}_i := D[k_{i-1}\!:\!k_i] \in \mathcal{E}^{k_i-k_{i-1}+1}$ *with embedding dimension* t_i, $B_i := \mathcal{K}^\dagger(\bar{D}_i) = \bar{U}_{B_i}S\bar{U}_{B_i}^T$, *where* $\bar{U}_{B_i} \in \mathcal{M}^{k\times t_i}$, $\bar{U}_{B_i}^T\bar{U}_{B_i} = I_{t_i}$, $S_i \in \mathcal{S}^{t_i}_{++}$, *and* $U_{B_i} := [\bar{U}_{B_i} \quad \frac{1}{\sqrt{k_i}}e] \in \mathcal{M}^{k\times(t_i+1)}$. *Let*

$$U := \begin{bmatrix} U_{B_1} & \ldots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \ldots & U_{B_l} & 0 \\ 0 & \ldots & 0 & I_{n-k_l} \end{bmatrix}$$

and $[V \quad \frac{U^Te}{\|U^Te\|}] \in \mathcal{M}^{n-k_l+\sum_{i=1}^l t_i+l}$ *be orthogonal. Then*

$$\begin{aligned} \textstyle\bigcap_{i=1}^l \text{face}\,\mathcal{K}^\dagger\left(\mathcal{E}^n(k_{i-1}\!:\!k_i, \bar{D}_i)\right) &= \left(U\mathcal{S}^{n-k_l+\sum_{i=1}^l t_i+l}_+U^T\right) \cap \mathcal{S}_C \\ &= (UV)\mathcal{S}^{n-k_l+\sum_{i=1}^l t_i+l-1}_+(UV)^T. \end{aligned} \tag{2.9}$$

Proof. The result follows from noting that the range of U is the intersection of the ranges of the matrices U_{B_i} with appropriate identity blocks added. $\square$

2.2. Two (intersecting) clique facial reduction. The construction (2.6) illustrates how we can find the intersection of two faces. Using this approach, we now extend Theorem 2.3 to two cliques that (possibly) intersect; see the ordered indices in (2.10) and the corresponding Venn diagram in Figure 2.1. We also find expressions for the intersection of the corresponding faces in $\mathcal{S}^n_+$; see (2.12). The key is to find the intersection of the subspaces that represent the faces, as in condition (2.11).

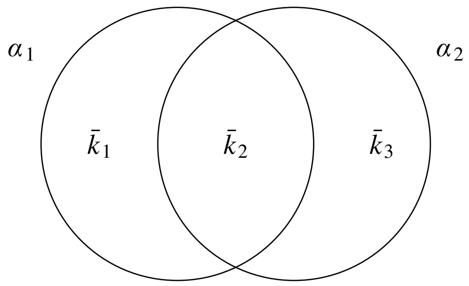

FIG. 2.1. *Venn diagram of the sets of ordered indices, α_1 and α_2, in Theorem 2.7.*

THEOREM 2.7. *Let $D \in \mathcal{E}^n$ with embedding dimension r, and, as in Figure 2.1, define the sets of positive integers*

$$\begin{aligned} \alpha_1 &:= 1\!:\!(\bar{k}_1+\bar{k}_2), \quad \alpha_2 := (\bar{k}_1+1)\!:\!(\bar{k}_1+\bar{k}_2+\bar{k}_3) \subseteq 1\!:\!n, \\ k_1 &:= |\alpha_1| = \bar{k}_1+\bar{k}_2, \quad k_2 := |\alpha_2| = \bar{k}_2+\bar{k}_3, \\ k &:= \bar{k}_1+\bar{k}_2+\bar{k}_3. \end{aligned} \tag{2.10}$$

For $i = 1,2$, let $\bar{D}_i := D[\alpha_i] \in \mathcal{E}^{k_i}$ with embedding dimension t_i and $B_i := \mathcal{K}^\dagger(\bar{D}_i) = \bar{U}_i S_i \bar{U}_i^T$, where $\bar{U}_i \in \mathcal{M}^{k_i \times t_i}$, $\bar{U}_i^T \bar{U}_i = I_{t_i}$, $S_i \in \mathcal{S}^{t_i}_{++}$, and $U_i := [\bar{U}_i \quad \frac{1}{\sqrt{k_i}}e] \in \mathcal{M}^{k_i \times (t_i+1)}$. Let t and $\bar{U} \in \mathcal{M}^{k \times (t+1)}$ satisfy

$$\mathcal{R}(\bar{U}) = \mathcal{R}\left(\begin{bmatrix} U_1 & 0 \\ 0 & I_{\bar{k}_3}\end{bmatrix}\right) \cap \mathcal{R}\left(\begin{bmatrix} I_{\bar{k}_1} & 0 \\ 0 & U_2\end{bmatrix}\right) \text{ with } \bar{U}^T\bar{U} = I_{t+1}. \tag{2.11}$$

Let $U := \left[\begin{smallmatrix} \bar{U} & 0 \\ 0 & I_{n-k}\end{smallmatrix}\right] \in \mathcal{M}^{n \times (n-k+t+1)}$ and $[V \quad \frac{U^Te}{\|U^Te\|}] \in \mathcal{M}^{n-k+t+1}$ be orthogonal. Then

$$\bigcap_{i=1}^{2} \operatorname{face} \mathcal{K}^\dagger\left(\mathcal{E}^n(\alpha_i, \bar{D}_i)\right) = \left(U\mathcal{S}^{n-k+t+1}_+ U^T\right) \cap \mathcal{S}_C = (UV)\mathcal{S}^{n-k+t}_+(UV)^T. \tag{2.12}$$

Proof. From Theorem 2.3, we have that

$$\operatorname{face} \mathcal{K}^\dagger\left(\mathcal{E}^n(\alpha_1, \bar{D}_1)\right) = \left(\left[\begin{array}{cc|c} U_1 & 0 & 0 \\ 0 & I_{\bar{k}_3} & 0 \\ \hline 0 & 0 & I_{n-k}\end{array}\right] \mathcal{S}^{n-k_1+t_1+1}_+ \left[\begin{array}{cc|c} U_1 & 0 & 0 \\ 0 & I_{\bar{k}_3} & 0 \\ \hline 0 & 0 & I_{n-k}\end{array}\right]^T\right) \cap \mathcal{S}_C$$

and, after a permutation of rows and columns in Theorem 2.3,

$$\operatorname{face} \mathcal{K}^\dagger\left(\mathcal{E}^n(\alpha_2, \bar{D}_2)\right) = \left(\left[\begin{array}{cc|c} I_{\bar{k}_1} & 0 & 0 \\ 0 & U_2 & 0 \\ \hline 0 & 0 & I_{n-k}\end{array}\right] \mathcal{S}^{n-k_2+t_2+1}_+ \left[\begin{array}{cc|c} I_{\bar{k}_1} & 0 & 0 \\ 0 & U_2 & 0 \\ \hline 0 & 0 & I_{n-k}\end{array}\right]^T\right) \cap \mathcal{S}_C.$$

The range space condition (2.11) then implies that

$$\mathcal{R}(U) = \mathcal{R}\left(\left[\begin{array}{cc|c} U_1 & 0 & 0 \\ 0 & I_{\bar{k}_3} & 0 \\ \hline 0 & 0 & I_{n-k} \end{array}\right]\right) \cap \mathcal{R}\left(\left[\begin{array}{cc|c} I_{\bar{k}_1} & 0 & 0 \\ 0 & U_2 & 0 \\ \hline 0 & 0 & I_{n-k} \end{array}\right]\right),$$

giving us the result (2.12). □

Remark 2.8. Theorem 2.7 provides a reduction in the dimension of the EDM completion problem. Initially, our problem consists in finding $Y \in \mathcal{S}^n_+ \cap \mathcal{S}_C$ such that the two constraints

$$\mathcal{K}(Y[\alpha_i]) = D[\alpha_i], \quad i = 1, 2$$

hold. After the reduction, we want to find the smaller dimensional $Z \in \mathcal{S}^{n-k+t}_+$; by construction $Y := (UV)Z(UV)^T$ will automatically satisfy the above constraints.

The explicit expression for the intersection of the two faces is given in (2.12) and uses the matrix $\bar{U}$ obtained from the intersection of the two ranges in condition (2.11). Finding a matrix whose range is the intersection of two subspaces can be done using [16, Algorithm 12.4.3]. However, our subspaces have special structure. We can exploit this structure to find the intersection; see Lemmas (2.9) and (2.13) below.

The dimension of the face in (2.12) is reduced to $n - k + t$. However, we can get a dramatic reduction if we have a common block with embedding dimension r and a reduction in the case where the common block has embedding dimension $r - 1$ as well. This provides an algebraic proof using semidefinite programming of the rigidity of the union of the two cliques under this intersection assumption.

2.2.1. Nonsingular facial reduction with intersection embedding dimension *r*. We now consider the case when the intersection of the two cliques results in $D[\alpha_1 \cap \alpha_2]$ having embedding dimension r; see Figure 2.2. We see that we can explicitly find the completion of the EDM $D[\alpha_1 \cup \alpha_2]$. We first need the following result on the intersection of two structured subspaces.

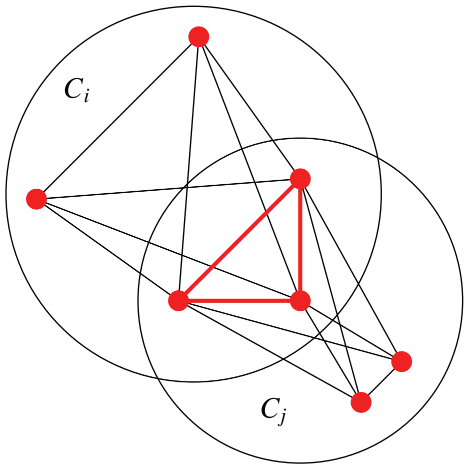

FIG. 2.2. *Two clique reduction with intersection with embedding dimension r.*

LEMMA 2.9. *Let*

$$U_1 := \begin{array}{c c} & {\scriptstyle r+1} \\ {\scriptstyle s_1} & \left[\begin{array}{c} U_1' \\ U_1'' \end{array}\right] \\ {\scriptstyle k} & \end{array}, \quad U_2 := \begin{array}{c c} & {\scriptstyle r+1} \\ {\scriptstyle k} & \left[\begin{array}{c} U_2'' \\ U_2' \end{array}\right] \\ {\scriptstyle s_2} & \end{array}, \quad \hat{U}_1 := \begin{array}{c c c} & {\scriptstyle r+1} & {\scriptstyle s_2} \\ {\scriptstyle s_1} & U_1' & 0 \\ {\scriptstyle k} & U_1'' & 0 \\ {\scriptstyle s_2} & 0 & I \end{array},$$

$$\hat{U}_2 := \begin{array}{c c c} & {\scriptstyle s_1} & {\scriptstyle r+1} \\ {\scriptstyle s_1} & I & 0 \\ {\scriptstyle k} & 0 & U_2'' \\ {\scriptstyle s_2} & 0 & U_2' \end{array}$$

be appropriately blocked with $U_1'', U_2'' \in \mathcal{M}^{k\times(r+1)}$ *full-column rank and* $\mathcal{R}(U_1'') = \mathcal{R}(U_2'')$. *Furthermore, let*

$$\bar{U}_1 := \begin{array}{c c} & {\scriptstyle r+1} \\ {\scriptstyle s_1} & U_1' \\ {\scriptstyle k} & U_1'' \\ {\scriptstyle s_2} & U_2'(U_2'')^\dagger U_1'' \end{array}, \quad \bar{U}_2 := \begin{array}{c c} & {\scriptstyle r+1} \\ {\scriptstyle s_1} & U_1'(U_1'')^\dagger U_2'' \\ {\scriptstyle k} & U_2'' \\ {\scriptstyle s_2} & U_2' \end{array}. \tag{2.13}$$

Then $\bar{U}_1$ *and* $\bar{U}_2$ *are full-column rank and satisfy*

$$\mathcal{R}(\hat{U}_1) \cap \mathcal{R}(\hat{U}_2) = \mathcal{R}\left(\bar{U}_1\right) = \mathcal{R}\left(\bar{U}_2\right).$$

Moreover, if $e_{r+1} \in \mathbb{R}^{r+1}$ *is the* $(r+1)$st *standard unit vector and* $U_i e_{r+1} = \alpha_i e$ *for some* $\alpha_i \neq 0$ *for* $i = 1, 2$, *then* $\bar{U}_i e_{r+1} = \alpha_i e$ *for* $i = 1, 2$.

Proof. From the definitions, $x \in \mathcal{R}(\hat{U}_1) \cap \mathcal{R}(\hat{U}_2)$ if and only if

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} U_1' v_1 \\ U_1'' v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} w_1 \\ U_2'' w_2 \\ U_2' w_2 \end{bmatrix} \quad \text{for some } v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}, \ w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}.$$

Note that $U_1'' v_1 = U_2'' w_2$ if and only if $w_2 = (U_2'')^\dagger U_1'' v_1$; this follows from the facts that U_2'' full-column rank implies $(U_2'')^\dagger U_2'' = I$ and $\mathcal{R}(U_1'') = \mathcal{R}(U_2'')$ implies $U_2''(U_2'')^\dagger U_1'' = U_1''$. Therefore, $x \in \mathcal{R}(\hat{U}_1) \cap \mathcal{R}(\hat{U}_2)$ if and only if

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} U_1' v_1 \\ U_1'' v_1 \\ U_2'(U_2'')^\dagger U_1'' v_1 \end{bmatrix} = \bar{U}_1 v_1 \quad \text{for some } v_1$$

with $v_2 := U_2'(U_2'')^\dagger U_1'' v_1$, $w_1 := U_1' v_1$, and $w_2 := (U_2'')^\dagger U_1'' v_1$, implying that $\mathcal{R}(\hat{U}_1) \cap \mathcal{R}(\hat{U}_2) = \mathcal{R}(\bar{U}_1)$; a similar argument shows that $\mathcal{R}(\hat{U}_1) \cap \mathcal{R}(\hat{U}_2) = \mathcal{R}(\bar{U}_2)$.

Now suppose, for $i = 1, 2$, that $U_i e_{r+1} = \alpha_i e$ for some $\alpha_i \neq 0$. Then $e \in \mathcal{R}(\hat{U}_1) \cap \mathcal{R}(\hat{U}_2)$, so $e \in \mathcal{R}(\bar{U}_1)$, implying that $\bar{U}_1 v = e$ for some vector v. Since $\bar{U}_1 = \left[\begin{smallmatrix} U_1 \\ U_2'(U_2'')^\dagger U_1'' \end{smallmatrix}\right]$, we have $U_1 v = e$. Furthermore, since U_1 has full-column rank, we conclude that $v = \frac{1}{\alpha_1} e_{r+1}$, implying that $\bar{U}_1 e_{r+1} = \alpha_1 e$. Similarly, we can show that $\bar{U}_2 e_{r+1} = \alpha_2 e$. ☐

We now state and prove a key result that shows we can complete the distances in the union of two cliques, provided that their intersection has embedding dimension equal to r.

THEOREM 2.10. *Let the hypotheses of Theorem* 2.7 *hold. Let*

$$\beta \subseteq \alpha_1 \cap \alpha_2, \quad \bar{D} := D[\beta], \quad B := \mathcal{K}^\dagger(\bar{D}), \quad \bar{U}_\beta := \bar{U}[\beta, :],$$

where $\bar{U} \in \mathcal{M}^{k\times(t+1)}$ *satisfies* (2.11). *Let* $[\bar{V} \quad \frac{\bar{U}^T e}{\|\bar{U}^T e\|}] \in \mathcal{M}^{t+1}$ *be orthogonal. Let*

$$Z := (J\bar{U}_\beta \bar{V})^\dagger B((J\bar{U}_\beta \bar{V})^\dagger)^T. \tag{2.14}$$

If the embedding dimension for $\bar{D}$ *is* r, *then* $t = r$, $Z \in \mathcal{S}^r_{++}$ *is the unique solution of*

$$(J\bar{U}_\beta \bar{V})Z(J\bar{U}_\beta \bar{V})^T = B, \tag{2.15}$$

and

$$D[\alpha_1 \cup \alpha_2] = \mathcal{K}\left((\bar{U}\bar{V})Z(\bar{U}\bar{V})^T\right). \tag{2.16}$$

Proof. Since the embedding dimension of $\bar{D}$ is r, we have $\operatorname{rank}(B) = r$. Furthermore, we have $Be = 0$ and $B \in \mathcal{S}^{|\beta|}_+$, implying that $|\beta| \geq r+1$. In addition, since the embedding dimension of D is also r, we conclude that the embedding dimension of $\bar{D}_i$ is r for $i = 1, 2$. Similarly, the embedding dimension of $D[\alpha_1 \cap \alpha_2]$ is also r.

Since $\bar{U} \in \mathcal{M}^{k\times(t+1)}$ satisfies (2.11), we have that

$$\mathcal{R}(\bar{U}) = \mathcal{R}\left(\begin{bmatrix} U_1' & 0 \\ U_1'' & 0 \\ 0 & I_{\bar{k}_3} \end{bmatrix}\right) \cap \mathcal{R}\left(\begin{bmatrix} I_{\bar{k}_1} & 0 \\ 0 & U_2'' \\ 0 & U_2' \end{bmatrix}\right).$$

Note that we have partitioned $U_i = [\bar{U}_i \quad \frac{1}{\sqrt{k_i}}e] \in \mathcal{M}^{k_i\times(r+1)}$ so that $U_i'' = [\bar{U}_i'' \quad \frac{1}{\sqrt{k_i}}e] \in \mathcal{M}^{|\alpha_1\cap\alpha_2|\times(r+1)}$ for $i = 1, 2$. Moreover, we have used the fact that the embedding dimension of $\bar{D}_i$ is r so that $t_i = r$ for $i = 1, 2$.

We claim that U_1'' and U_2'' have full-column rank and that $\mathcal{R}(U_1'') = \mathcal{R}(U_2'')$. First we let $Y := \mathcal{K}^\dagger(D[\alpha_1 \cup \alpha_2])$. Then $Y \in \mathcal{K}^\dagger\left(\mathcal{E}^k(\alpha_1, \bar{D}_1)\right)$. By Theorem 2.3, there exists $Z_1 \in \mathcal{S}^{\bar{k}_3+r+1}_+$ such that

$$Y = \begin{bmatrix} U_1' & 0 \\ U_1'' & 0 \\ 0 & I_{\bar{k}_3} \end{bmatrix} Z_1 \begin{bmatrix} U_1' & 0 \\ U_1'' & 0 \\ 0 & I_{\bar{k}_3} \end{bmatrix}^T.$$

Therefore, $Y[\alpha_1 \cap \alpha_2] = [U_1'' \quad 0]Z_1[U_1'' \quad 0]^T \in U_1''\mathcal{S}^{r+1}_+(U_1'')^T$, so

$$\mathcal{R}(Y[\alpha_1 \cap \alpha_2]) \subseteq \mathcal{R}(U_1'').$$

Furthermore, since $\mathcal{K}(Y) = D[\alpha_1 \cup \alpha_2]$, we have that $\mathcal{K}(Y[\alpha_1 \cap \alpha_2]) = D[\alpha_1 \cap \alpha_2] = \mathcal{K}(\mathcal{K}^\dagger(D[\alpha_1 \cap \alpha_2]))$, so $Y[\alpha_1 \cap \alpha_2] \in \mathcal{K}^\dagger(D[\alpha_1 \cap \alpha_2]) + \mathcal{N}(\mathcal{K})$. Since $\mathcal{N}(\mathcal{K}) = \mathcal{R}(\mathcal{D}_e)$, there exists a vector y such that

$$Y[\alpha_1 \cap \alpha_2] = \mathcal{K}^\dagger(D[\alpha_1 \cap \alpha_2]) + \mathcal{D}_e(y) = \mathcal{K}^\dagger(D[\alpha_1 \cap \alpha_2]) + ye^T + ey^T.$$

By Lemma 2.1, $y \in \mathcal{R}([\mathcal{K}^\dagger(D[\alpha_1 \cap \alpha_2]) \quad e])$. Therefore,

$$\mathcal{R}(Y[\alpha_1 \cap \alpha_2]) = \mathcal{R}\left(\left[\mathcal{K}^\dagger(D[\alpha_1 \cap \alpha_2]) \quad e\right]\right).$$

Moreover, $\operatorname{rank} \mathcal{K}^\dagger(D[\alpha_1 \cap \alpha_2]) = r$ and $\mathcal{K}^\dagger(D[\alpha_1 \cap \alpha_2])e = 0$, so

$$r + 1 = \dim \mathcal{R}(Y[\alpha_1 \cap \alpha_2]) \leq \dim \mathcal{R}(U_1'') \leq r + 1.$$

Therefore, U_1'' has full-column rank, and $\mathcal{R}(U_1'') = \mathcal{R}(Y[\alpha_1 \cap \alpha_2])$. Similarly, we can show that U_2'' has full-column rank and that $\mathcal{R}(U_2'') = \mathcal{R}(Y[\alpha_1 \cap \alpha_2])$, so we conclude that $\mathcal{R}(U_1'') = \mathcal{R}(U_2'')$.

We now claim that $t = r$, where $\bar{U} \in \mathcal{M}^{k\times(t+1)}$ satisfies (2.11). Since $U_1'', U_2'' \in \mathcal{M}^{|\alpha_1\cap\alpha_2|\times(r+1)}$ have full-column rank and $\mathcal{R}(U_1'') = \mathcal{R}(U_2'')$, we have by Lemma 2.9 that $\mathcal{R}(\bar{U}) = \mathcal{R}(\bar{U}_1) = \mathcal{R}(\bar{U}_2)$, where

$$\bar{U}_1 := \begin{bmatrix} U_1' \\ U_1'' \\ U_2'(U_2'')^\dagger U_1'' \end{bmatrix} \quad \text{and} \quad \bar{U}_2 := \begin{bmatrix} U_1'(U_1'')^\dagger U_2'' \\ U_2'' \\ U_2' \end{bmatrix}.$$

Therefore,

$$t + 1 = \dim \mathcal{R}(\bar{U}) = \dim \mathcal{R}(\bar{U}_1) = \dim \mathcal{R}(\bar{U}_2) = r + 1,$$

so we have $t = r$ as claimed.

Recall that $Y = \mathcal{K}^\dagger(D[\alpha_1 \cup \alpha_2])$, so $Y \in \cap_{i=1,2}\mathcal{K}^\dagger(\mathcal{E}^k(\alpha_i, \bar{D}_i))$. Thus, Theorem 2.7 implies that there exists $\bar{Z} \in \mathcal{S}_+^r$ such that $Y = (\bar{U}\bar{V})\bar{Z}(\bar{U}\bar{V})^T$. Observe that $\mathcal{K}(Y[\beta]) = D[\beta] = \bar{D}$. Thus,

$$\mathcal{K}\left((\bar{U}_\beta \bar{V})\bar{Z}(\bar{U}_\beta \bar{V})^T\right) = \bar{D},$$

implying that

$$\mathcal{K}^\dagger\,\mathcal{K}\left((\bar{U}_\beta \bar{V})\bar{Z}(\bar{U}_\beta \bar{V})^T\right) = B.$$

Since $\mathcal{K}^\dagger\,\mathcal{K}$ is the projection onto $\mathcal{R}(\mathcal{K}^*) = \mathcal{S}_C$, we have that $\mathcal{K}^\dagger\,\mathcal{K}(\cdot) = J(\cdot)J$. Therefore, we have that $\bar{Z}$ satisfies (2.15). It remains to show that (2.15) has a unique solution. Let $A := J\bar{U}_\beta\bar{V} \in \mathcal{M}^{|\beta|\times r}$. Then $A\bar{Z}A^T = B$ and $\operatorname{rank}(B) = r$ implies that $\operatorname{rank}(A) \geq r$, so A has full-column rank. This implies that (2.15) has a unique solution and that $\bar{Z} = A^\dagger B(A^\dagger)^T = Z$. Finally, since $Y = (\bar{U}\bar{V})Z(\bar{U}\bar{V})^T$ and $D[\alpha_1 \cup \alpha_2] = \mathcal{K}(Y)$, we get (2.16). ☐

The following result shows that if we know the minimal face of $\mathcal{S}_+^n$ containing $\mathcal{K}^\dagger(D)$ and we know a small submatrix of D, then we can compute a set of points in $\mathbb{R}^r$ that generate D by solving a small equation.

COROLLARY 2.11. *Let $D \in \mathcal{E}^n$ with embedding dimension r, and let $\beta \subseteq 1\!:\!n$. Let $U \in \mathcal{M}^{n\times(r+1)}$ satisfy*

$$\operatorname{face}\mathcal{K}^\dagger(D) = \left(U\mathcal{S}_+^{r+1}U^T\right) \cap \mathcal{S}_C,$$

let $U_\beta := U[\beta,:]$, and let $[V \;\; \frac{U^Te}{\|U^Te\|}] \in \mathcal{M}^{r+1}$ be orthogonal. If $D[\beta]$ has embedding dimension r, then

$$(JU_\beta V)Z(JU_\beta V)^T = \mathcal{K}^\dagger(D[\beta])$$

has a unique solution $Z \in \mathcal{S}_{++}^r$ and $D = \mathcal{K}(PP^T)$, where $P := UVZ^{1/2} \in \mathbb{R}^{n\times r}$.

Proof. Apply Theorem 2.10 with $\alpha_1 = \alpha_2 = 1\!:\!n$. ☐

Remark 2.12. A more efficient way to calculate Z uses the full rank factorization

$$B = QD^{1/2}\left(QD^{1/2}\right)^T, \quad Q^TQ = I_r, \quad D \in \mathcal{S}_{++}^r.$$

Let $C = (J\bar{U}_\beta\bar{V})^\dagger(QD^{1/2})$. Then Z in (2.14) can be found from $Z = CC^T$. Note that our algorithm postpones finding Z until the end, where we can no longer perform any clique reductions. At each iteration, we compute the matrix $\bar{U}$ that represents the face corresponding to the union of two cliques; $\bar{U}$ is chosen from one of $\bar{U}_i$ for $i = 1, 2$ in (2.13). Moreover, for stability, we maintain $\bar{U}^T\bar{U} = I$, $\bar{U}e_{r+1} = \alpha e$.

For many of our test problems, we can repeatedly apply Theorem 2.10 until there is only one clique left. Since each repetition reduces the number of cliques by one, this means that there are at most n such steps.

2.2.2. Singular facial reduction with intersection embedding dimension $\boldsymbol{r-1}$. We now show that if the embedding dimension of the intersection is $r-1$ (i.e., deficient), then we can find at most two completions. If exactly one of these two completions is feasible in the sense that it satisfies the related distance equality constraints and, if included, the related lower bound inequality constraints obtained from the radio range R, then we have identified the unique completion; see Figure 2.3. We first need the following extension of Lemma 2.9 on the intersection of two structured subspaces for the case where the common middle blocks are not full rank.

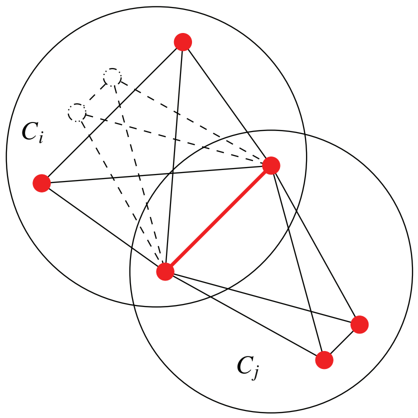

FIG. 2.3. *Two clique reduction with intersection having embedding dimension $< r$.*

LEMMA 2.13. *Let $U_i, \hat{U}_i, \bar{U}_i$ for $i = 1, 2$ be defined and appropriately blocked as in Lemma 2.9, with $U_i'' \in \mathcal{M}^{k\times(r+1)}$ having rank r for $i = 1, 2$ and $\mathcal{R}(U_1'') = \mathcal{R}(U_2'')$. Let $0 \neq u_i \in \mathcal{N}(U_i'')$ for $i = 1, 2$. If $\bar{U} \in \mathcal{M}^{k\times(t+1)}$ satisfies $\mathcal{R}(\bar{U}) = \mathcal{R}(\hat{U}_1) \cap \mathcal{R}(\hat{U}_2)$, then $t = r + 1$ and*

$$\begin{aligned}
\mathcal{R}(\bar{U}) &= \mathcal{R}\left(\begin{bmatrix} U_1' & 0 \\ U_1'' & 0 \\ U_2'(U_2'')^\dagger U_1'' & U_2'u_2 \end{bmatrix}\right) = \mathcal{R}\left(\begin{bmatrix} \bar{U}_1 & \begin{bmatrix} 0 \\ 0 \\ U_2'u_2 \end{bmatrix} \end{bmatrix}\right) \\
&= \mathcal{R}\left(\begin{bmatrix} U_1'(U_1'')^\dagger U_2'' & U_1'u_1 \\ U_2'' & 0 \\ U_2' & 0 \end{bmatrix}\right) = \mathcal{R}\left(\begin{bmatrix} \bar{U}_2 & \begin{bmatrix} U_1'u_1 \\ 0 \\ 0 \end{bmatrix} \end{bmatrix}\right).
\end{aligned} \tag{2.17}$$

Moreover, if $e_{r+1} \in \mathbb{R}^{r+1}$ is the $(r + 1)$st *standard unit vector and $U_ie_{r+1} = \alpha_ie$ for some $\alpha_i \neq 0$ for $i = 1, 2$, then $\bar{U}_ie_{r+1} = \alpha_ie$ for $i = 1, 2$.*

Proof. From the definitions, $x \in \mathcal{R}(\bar{U})$ if and only if

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} U_1' v_1 \\ U_1'' v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} w_1 \\ U_2'' w_2 \\ U_2' w_2 \end{bmatrix} \text{ for some } v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}, w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}. \tag{2.18}$$

Since $\mathcal{R}(U_1'') = \mathcal{R}(U_2'')$ and $U_i'', i = 1, 2$ are both rank r, we conclude that $x_2 = U_1'' v_1 = U_2'' w_2$ for some v_1, w_2 if and only if $x_2 \in \mathcal{R}(U_1'')$ with v_1, w_2 determined by

$$v_1 = (U_1'')^\dagger x_2 + \alpha_1 u_1 \text{ for some } \alpha_1 \in \mathbb{R}, \qquad w_2 = (U_2'')^\dagger U_1'' v_1 + \alpha_2 u_2 \text{ for some } \alpha_2 \in \mathbb{R}.$$

In other words, we get

$$\begin{gathered} x_2 = U_1'' v_1 = U_2'' w_2 \text{ for some } v_1, w_2 \\ \text{if and only if} \\ x_2 = U_1'' v_1 \text{ for some } v_1 \text{ with } w_2 = (U_2'')^\dagger U_1'' v_1 + \alpha_2 u_2 \text{ for some } \alpha_2 \in \mathbb{R}. \end{gathered} \tag{2.19}$$

After substituting for v_2 with $v_2 = U_2' w_2 = U_2'((U_2'')^\dagger U_1'' v_1 + \alpha_2 u_2)$, we conclude that (2.18) holds if and only if the first equality in (2.17) holds; i.e., (2.18) holds if and only if

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} U_1' v_1 \\ U_1'' v_1 \\ U_2'(U_2'')^\dagger U_1'' v_1 + \alpha_2 U_2' u_2 \end{bmatrix} \text{ for some } v_1, \alpha_2,$$

where

$$v_2 = U_2'(U_2'')^\dagger U_1'' v_1 + \alpha_2 U_2' u_2, \quad w_1 = U_1' v_1, \quad w_2 = (U_2'')^\dagger U_1'' v_1 + \alpha_2 u_2.$$

The second equality in (2.17) follows similarly. The last statements about $\bar{U}_i e_{r+1}$ follow as in the proof of Lemma 2.9. ∎

In the rigid case in Theorem 2.10, we use the expression for $\bar{U}$ from Lemma 2.9 to obtain a unique Z in order to get the completion of $D[\alpha_1 \cup \alpha_2]$. The Z is unique because the $r + 1$ columns of $\bar{U}$ that represent the new clique $\alpha_1 \cup \alpha_2$ are linearly independent, $e \in \mathcal{R}(\bar{U})$, $\operatorname{rank}(B) = r$, and $Be = 0$. This means that the solution C of $(J\bar{U}_\beta \bar{V})C = QD^{1/2}$ in Remark 2.12 exists and is unique. (Recall that $J\bar{U}_\beta \bar{V}$ is full-column rank.) This also means that the two matrices, U_1 and U_2, that represent the cliques, α_1 and α_2, respectively, can be replaced by the single matrix $\bar{U}$ without actually calculating C; we can use $\bar{U}$ to represent the clique $\alpha_1 \cup \alpha_2$ and complete all or part of the partial EDM $D[\alpha_1 \cup \alpha_2]$ only when needed.

We have a similar situation for the singular intersection case following Lemma 2.13. We have the matrix $\bar{U}$ to represent the intersection of the two subspaces, where each subspace represents one of the cliques, α_1 or α_2. However, this is not equivalent to *uniquely* representing the union of the two cliques, α_1 or α_2, since there is an extra column in $\bar{U}$ compared to the nonsingular case. In addition, since $\operatorname{rank}(B) = r-1$, then $J\bar{U}_\beta \bar{V}$ is not necessarily full-column rank. Therefore, there may be infinite solutions for C in Remark 2.12; any $C \in (J\bar{U}_\beta \bar{V})^\dagger \left(QD^{1/2}\right) + \mathcal{N}(J\bar{U}_\beta \bar{V})$ will give us a solution. Moreover, these solutions will not necessarily satisfy $\mathcal{K}((\bar{U}C)(\bar{U}C)^T) = D[\alpha_1 \cup \alpha_2]$. We now see that we can continue and use the $\bar{U}$ to represent a set of cliques rather than just $\alpha_1 \cup \alpha_2$. Alternatively, we can use other relevant distance equality constraints or lower bound constraints from the radio range R to determine the correct C in order to get the correct number of columns for $\bar{U}$; we can then get the correct completion of $D[\alpha_1 \cup \alpha_2]$ if exactly one of the two possible completions with embedding dimension r is feasible.

THEOREM 2.14. *Let the hypotheses of Theorem* 2.10 *hold with the special case that $U_i^T U_i = I$, $U_i e_{r+1} = \alpha_i e$ for $i = 1, 2$. In addition, let $\bar{U}$ be defined by one of the expressions in* (2.17) *in Lemma* 2.13. *For $i = 1, 2$, let $\beta \subset \delta_i \subseteq \alpha_i$ and $A_i := J\bar{U}_{\delta_i}\bar{V}$, where $\bar{U}_{\delta_i} := \bar{U}(\delta_i, :)$. Furthermore, let $B_i := \mathcal{K}^\dagger(D[\delta_i])$, define the linear system*

$$\begin{array}{rcl} A_1 Z A_1^T & = & B_1, \\ A_2 Z A_2^T & = & B_2, \end{array} \tag{2.20}$$

and let $\bar{Z} \in \mathcal{S}^t$ be a particular solution of this system (2.20). *If the embedding dimensions of $D[\delta_1]$ and $D[\delta_2]$ are both r but the embedding dimension of $\bar{D} := D[\beta]$ is $r - 1$, then the following holds:*

1. $\dim \mathcal{N}(A_i) = 1$ *for $i = 1, 2$.*
2. *For $i = 1, 2$, let $n_i \in \mathcal{N}(A_i)$, $\|n_i\|_2 = 1$, and $\Delta Z := n_1 n_2^T + n_2 n_1^T$. Then Z is a solution of the linear system* (2.20) *if and only if*
$$Z = \bar{Z} + \tau \Delta Z \quad \textit{for some } \tau \in \mathbb{R}. \tag{2.21}$$
3. *There are at most two nonzero solutions, τ_1 and τ_2, for the generalized eigenvalue problem $-\Delta Z v = \tau \bar{Z} v$, $v \neq 0$. Set $Z_i := \bar{Z} + \frac{1}{\tau_i}\Delta Z$ for $i = 1, 2$. Then*
$$D[\alpha_1 \cup \alpha_2] \in \left\{ \mathcal{K}(\bar{U}\bar{V} Z_i \bar{V}^T \bar{U}^T) : i = 1, 2 \right\}.$$

Proof. We follow a similar proof as in the nonsingular case. For simplicity, we assume that $\delta_i = \alpha_i$ for $i = 1, 2$ (choosing smaller δ_i can reduce the cost of solving the linear systems).

That a particular solution $\bar{Z}$ exists for the system (2.20) follows from the fact that $\bar{U}$ provides a representation for the intersection of the two faces (or the union of the two cliques).

Since the embedding dimension of $\bar{D}$ is $r - 1$, we have $\operatorname{rank}(B) = r - 1$. Furthermore, we have $Be = 0$ and $B \in \mathcal{S}_+^{|\beta|}$, implying that $|\beta| \geq r$. Without loss of generality and for simplicity, we assume that $|\beta| = r$. Therefore, there exists $0 \neq u_i \in \mathcal{N}(U_i'')$ for $i = 1, 2$. From Lemma 2.13, we can assume that we maintain $\bar{U}_i^T \bar{U}_i = I$, $\bar{U}_i e_{r+1} = \alpha_i e$ for some $\alpha_i \neq 0$ for $i = 1, 2$. Therefore, the action of $\bar{V}$ is equivalent to removing the $r + 1$ column of $\bar{U}_i$. We can then explicitly use u_i to write down $n_i \in \mathcal{N}(A_i)$. By construction, we now have $A_i(n_1 n_2^T + n_2 n_1^T)A_i^T = 0$ for $i = 1, 2$.

From the first expression for $\bar{U}$ in (2.17), we see that the choices for n_1 and n_2 in the first part are in the appropriate nullspaces. The dimensions follow from the assumptions on the embedding dimensions.

The second part now follows from the definition of the general solution of a linear system of equations; i.e., the sum of a particular solution with any solution of the homogeneous equation.

The third part now follows from the role that $\bar{U}$ plays as a representation for the union of the two cliques. $\square$

Remark 2.15. As above in the nonsingular case, a more efficient way to calculate $\bar{Z}$ uses the full rank factorization

$$B_i = QD^{1/2}\left(Q_i D_i^{1/2}\right)^T, \quad Q_i^T Q_i = I_r, \quad D_i \in \mathcal{S}_{++}^r, \quad i = 1, 2.$$

(We have assumed that both have embedding dimension r, although we need only one that does.) We solve the equations $A_i C = (Q_i D_i^{1/2})\bar{Q}_i$, $\bar{Q}_i \bar{Q}_i^T = I$ for $i = 1, 2$ for the unknowns C, and $\bar{Q}_i$ for $i = 1, 2$. Then a particular solution $\bar{Z}$ in (2.20) can be found from $\bar{Z} = CC^T$. Note that the additional orthogonal matrices $\bar{Q}_i$ for $i = 1, 2$ are needed since they still allow $A_i C(A_i C)^T = B_i$ for $i = 1, 2$. Also, without loss of generality, we can assume $\bar{Q}_1 = I$.

2.3. Clique initialization and node absorption. Using the above clique reductions, we now consider techniques that allow one clique to grow/absorb other cliques. This applies Theorem 2.10. We first consider an elementary and fast technique to find some of the existing cliques.

LEMMA 2.16. *For each $i \in \{1, \ldots, n\}$, use half the radio range, and define the set*

$$C_i := \left\{ j \in \{1, \ldots, n\} : D_{ij} \leq (R/2)^2 \right\}.$$

Then each C_i corresponds to a clique of sensors that are within radio range of each other.

Proof. Let $j, k \in C_i$ for a given $i \in \{1, \ldots, n\}$. An elementary application of the triangle inequality shows that $\sqrt{(D_{jk})} \leq \sqrt{(D_{ji})} + \sqrt{(D_{ki})} \leq R$. $\square$

We can now assume that we have a finite set of indices $\mathcal{C} \subseteq \mathbb{Z}_+$ corresponding to a family of cliques, $\{C_i\}_{i \in \mathcal{C}}$. We can combine cliques using the reductions given in Theorems 2.10 and 2.14. We now see how a clique can grow further by absorbing individual sensors; see Figure 2.4.

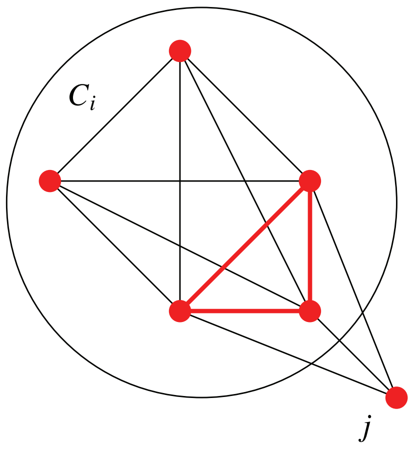

FIG. 2.4. *Absorption with intersection having embedding dimension r.*

COROLLARY 2.17. *Let C_k for $k \in \mathcal{C}$ be a given clique with node $l \notin C_k$, $\beta := \{j_1, \ldots, j_{r+1}\} \subseteq C_k$ such that the distances D_{lj_i}, for $i = 1, \ldots, r+1$ are known. If*

$$\operatorname{rank} \mathcal{K}^\dagger(D[\beta]) = r, \tag{2.22}$$

then l can be absorbed by the clique C_k and we can complete the missing elements in column (row) l of $D[C_k \cup \{l\}]$.

Proof. Let $\alpha_1 := C_k$, $\alpha_2 := \{j_1, \ldots, j_{r+1}, l\}$, and $\beta := \alpha_1 \cap \alpha_2 = \{j_1, \ldots, j_{r+1}\}$. Then the conditions in Theorem 2.10 are satisfied, and we can recover all the missing elements in $D[C_k \cup \{l\}]$. $\square$

2.3.1. Node absorption with degenerate intersection. We can apply the same reasoning as for the clique reduction in the nonsingular case, except now we apply Theorem 2.14. To obtain a unique completion, we test the feasibility of the two possible completions against any related distance equality constraints or, if included, any related lower bound inequality constraints. See Figure 2.5.

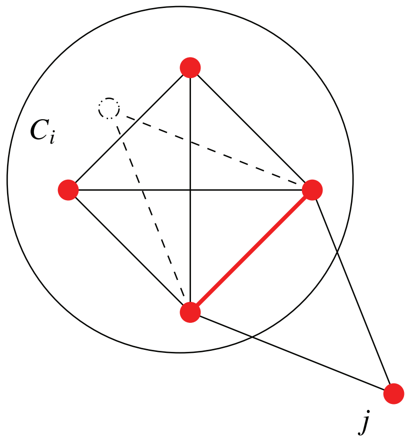

FIG. 2.5. *Degenerate absorption with intersection with embedding dimension less than* r.

COROLLARY 2.18. *Let* C_k *for* $k \in \mathcal{C}$ *be a given clique with node* $l \notin C_k$, $\beta := \{j_1, \dots j_r\} \subseteq C_k$ *such that the distances* D_{lj_i} *for* $i = 1, \dots, r$ *are known. If*

$$\operatorname{rank} \mathcal{K}^\dagger(D[\beta]) = r - 1, \tag{2.23}$$

then we can determine two possible completions of the distances. If exactly one of these two completions is feasible, then l *can be absorbed by the clique* C_k. *We can also complete the missing elements in column (row)* l *of* $D[C_k \cup \{l\}]$.

Proof. Let $\alpha_1 := C_k$, $\alpha_2 := \{j_1, \dots, j_r, l\}$, and $\beta := \alpha_1 \cap \alpha_2 = \{j_1, \dots, j_r\}$. Then the conditions in Theorem 2.14 are satisfied, and we can recover all the missing elements in $D[C_k \cup \{l\}]$. ▯

3. `SNLSDPclique` facial reduction algorithm and numerical results.

Our `SNLSDPclique` algorithm starts by forming a clique C_i around each sensor i. If and when we use this clique, we find a subspace representation from the r eigenvectors corresponding to the r nonzero eigenvalues of $B = \mathcal{K}^\dagger(D[C_i])$.

The algorithm then grows and combines cliques using Theorems 2.10 and 2.14 and Corollaries 2.17 and 2.18. In particular, we do not complete the EDM each time we combine or grow cliques; i.e., we do not evaluate the missing distances. Instead, we use the subspace representations of the corresponding faces of the SDP cone and then find the intersection of the subspaces that represent the faces. This yields a subspace representation of the new smaller face representing the union of two cliques. This is based on Lemmas 2.9 and 2.13 and is therefore inexpensive.

Once we cannot, or need not, grow cliques, we complete the distances using Corollary 2.11. This is also inexpensive. Finally, we rotate and translate the anchors to their original positions using the approach outlined in [13]. We have provided an outline of our facial reduction algorithm `SNLSDPclique` in Algorithm 1.

Algorithm 1: SNLSDPclique A Facial Reduction Algorithm.

```
input  : Partial $n \times n$ EDM $D_p$ and anchors $A \in \mathbb{R}^{m \times r}$;
output : $X \in \mathbb{R}^{|C_i| \times r}$, where $C_i$ is the largest final clique that contains the
         anchors;
1  Let $\mathcal{C} := \{1, \ldots, n+1\}$;
2  Let $\{C_i\}_{i \in \mathcal{C}}$ be a family of cliques satisfying $i \in C_i$ for all $i = 1, \ldots, n$; /* For
   example, by Lemma 2.16, we could choose $C_i := \left\{j : (D_p)_{ij} < (R/2)^2\right\}$
   for $i = 1, \ldots, n$. Alternatively, we could simply choose $C_i := \{i\}$ for
   $i = 1, \ldots, n$.  */
3  Let $C_{n+1} := \{n-m+1, \ldots, n\}$;    /* $C_{n+1}$ is the clique of anchors */
   /* GrowCliques                                                   */
4  Choose MaxCliqueSize $> r+1$;                          /* For example,
   MaxCliqueSize $:= 3(r+1)$ */
5  for $i \in \mathcal{C}$ do
6      while ($|C_i| <$ MaxCliqueSize) and ($\exists$ a node $j$ adjacent to all nodes in
       $C_i$) do
7          $C_i := C_i \cup \{j\}$;
8      end
9  end
   /* ComputeFaces                                                  */
10 for $i \in \mathcal{C}$ do
11     Compute $U_{B_i} \in \mathbb{R}^{|C_i| \times (r+1)}$ to represent face for clique $C_i$;     /* see
       Theorem 2.3 */
       /* Alternatively, wait to compute $U_{B_i}$ when first needed.  This
          can be more efficient since $U_{B_i}$ is not needed for every
          clique.                                                    */
12 end
13 repeat
14     if $|C_i \cap C_j| \geq r+1$ for some $i, j \in \mathcal{C}$, then
15         RigidCliqueUnion($C_i$,$C_j$);                 /* see Algorithm 2 */
16     else if $|C_i \cap \mathcal{N}(j)| \geq r+1$ for some $i \in \mathcal{C}$ and node $j$, then
17         RigidNodeAbsorption($C_i$,$j$);                /* see Algorithm 3 */
18     else if $|C_i \cap C_j| = r$ for some $i, j \in \mathcal{C}$, then
19         NonRigidCliqueUnion($C_i$,$C_j$);              /* see Algorithm 4 */
20     else if $|C_i \cap \mathcal{N}(j)| = r$ for some $i \in \mathcal{C}$ and node $j$, then
21         NonRigidNodeAbsorption($C_i$,$j$);             /* see Algorithm 5 */
22     end
23 until not possible to decrease $|\mathcal{C}|$ or increase $|C_i|$ for some $i \in \mathcal{C}$;
24 Let $C_i$ be the largest clique that contains the anchors;
25 if clique $C_i$ contains some sensors, then
26     Compute a point representation $P \in \mathbb{R}^{|C_i| \times r}$ for the clique $C_i$;    /* see
       Cor. 2.11 */
27     Compute positions of sensors $X \in \mathbb{R}^{(|C_i|-m) \times r}$ in clique $C_i$ by rotating $P$
       to align with anchor positions $A \in \mathbb{R}^{m \times r}$;     /* see Ding et al. [13,
       Method 3.2] */
28     return $X$;
29 else
30     return $X := \emptyset$;
31 end
```

Algorithm 2: `RigidCliqueUnion`.

```
input : Cliques $C_i$ and $C_j$ such that $|C_i \cap C_j| \geq r+1$;
1 Load $U_{B_i} \in \mathbb{R}^{|C_i|\times(r+1)}$ and $U_{B_j} \in \mathbb{R}^{|C_j|\times(r+1)}$ representing the faces
  corresponding to the cliques $C_i$ and $C_j$, respectively;
2 Compute $\bar{U} \in \mathbb{R}^{|C_i \cup C_j|\times(r+1)}$ using one of the two formulas in (2.13) from
  Lemma 2.9, where $U_1 = U_{B_i}$, $U_2 = U_{B_j}$, and $k = |C_i \cap C_j|$;    /* see
  Theorem 2.7 */
3 Update $C_i := C_i \cup C_j$;
4 Update $U_{B_i} := \bar{U}$;
5 Update $\mathcal{C} := \mathcal{C} \setminus \{j\}$;
```

Algorithm 3: `RigidNodeAbsorption`.

```
input : Clique $C_i$ and node $j$ such that $|C_i \cap \mathcal{N}(j)| \geq r+1$;
1 Load $U_{B_i} \in \mathbb{R}^{|C_i|\times(r+1)}$ representing the face corresponding to clique $C_i$;
2 if $C_i \cap \mathcal{N}(j)$ is not a clique in the original graph, then
3     Use $U_{B_i}$ to compute a point representation $P_i \in \mathbb{R}^{|C_i|\times r}$ of the sensors in
      $C_i$;
                                                   /* see Cor. 2.11 */
4     Use $P_i$ to compute the distances between the sensors in $C_i \cap \mathcal{N}(j)$;
5 end
6 Use the distances between the sensors in $(C_i \cap \mathcal{N}(j)) \cup \{j\}$ to compute the
  matrix $U_{B_j} \in \mathbb{R}^{(|C_i \cap \mathcal{N}(j)|+1)\times(r+1)}$ representing the face corresponding to the
  clique $(C_i \cap \mathcal{N}(j)) \cup \{j\}$;                /* see Theorem 2.3 */
7 Compute $\bar{U} \in \mathbb{R}^{(|C_i|+1)\times(r+1)}$ using one of the two formulas in (2.13) from
  Lemma 2.9, where $U_1 = U_{B_i}$, $U_2 = U_{B_j}$, and $k = |C_i \cap \mathcal{N}(j)|$;    /* see
  Theorem 2.7 */
8 Update $C_i := C_i \cup \{j\}$;
9 Update $U_{B_i} := \bar{U}$;
```

Algorithm 4: `NonRigidCliqueUnion`.

```
input : Cliques $C_i$ and $C_j$ such that $|C_i \cap C_j| = r$;
1 Load $U_{B_i} \in \mathbb{R}^{|C_i|\times(r+1)}$ and $U_{B_j} \in \mathbb{R}^{|C_j|\times(r+1)}$ representing the faces
  corresponding to the cliques $C_i$ and $C_j$, respectively;
2 Using $U_{B_i}$ and $U_{B_j}$, find the two point representations of the sensors in
  $C_i \cup C_j$;
                                                  /* see Theorem 2.14 */
3 if exactly one of these two point representations is feasible, then
4     Use the feasible point representation to compute $\bar{U} \in \mathbb{R}^{|C_i \cup C_j|\times(r+1)}$
      representing the face corresponding to the clique $C_i \cup C_j$;    /* see
      Theorem 2.3 */
5     Update $C_i := C_i \cup C_j$;
6     Update $U_{B_i} := \bar{U}$;
7     Update $\mathcal{C} := \mathcal{C} \setminus \{j\}$;
8 end
```

ALGORITHM 5: NonRigidNodeAbsorption.

```
input : Clique $C_i$ and node $j$ such that $|C_i \cap \mathcal{N}(j)| = r$;
1  Load $U_{B_i} \in \mathbb{R}^{|C_i| \times (r+1)}$ representing the face corresponding to clique $C_i$;
2  if $C_i \cap \mathcal{N}(j)$ is not a clique in the original graph, then
3      Use $U_{B_i}$ to compute a point representation $P_i \in \mathbb{R}^{|C_i| \times r}$ of the sensors in $C_i$;
                                                                  /* see Cor. 2.11 */
4      Use $P_i$ to compute the distances between the sensors in $C_i \cap \mathcal{N}(j)$;
5  end
6  Use the distances between the sensors in $(C_i \cap \mathcal{N}(j)) \cup \{j\}$ to compute the
   matrix $U_{B_j} \in \mathbb{R}^{(|C_i \cap \mathcal{N}(j)|+1) \times (r+1)}$ representing the face corresponding to the
   clique $(C_i \cap \mathcal{N}(j)) \cup \{j\}$;                    /* see Theorem 2.3 */
7  Using $U_{B_i}$ and $U_{B_j}$, find the two point representations of the sensors in
   $C_i \cup \{j\}$;
                                                                  /* see Theorem 2.14 */
8  if exactly one of these two point representations is feasible, then
9      Use the feasible point representation to compute $\bar{U} \in \mathbb{R}^{|C_i \cup C_j| \times (r+1)}$
       representing the face corresponding to the clique $C_i \cup \{j\}$;      /* see
       Theorem 2.3 */
10     Update $C_i := C_i \cup \{j\}$;
11     Update $U_{B_i} := \bar{U}$;
12 end
```

3.1. Numerical tests. Our tests are on problems with sensors and anchors randomly placed in the region $[0,1]^r$ by means of a uniform random distribution. We vary the number of sensors from 2000 to 10000 in steps of 2000 and the radio range R from .07 to .04 in steps of $-.01$. We also include tests on very large problems with 20000 to 100000 sensors. In our tests, we did not use the lower bound inequality constraints coming from the radio range; we used only the equality constraints coming from the partial EDM. Our tests were done using the 32-bit version of MATLAB R2009b on a laptop running Windows XP with a 2.16 GHz Intel Core 2 Duo processor and with 2 GB of RAM. The source code used for running our tests has been released under a GNU General Public License and has been made available from the authors' websites.

We, in particular, emphasize the low CPU times and the high accuracy of the solutions we obtain. Our algorithm compares well with the recent work in [23, 28], where they use, for example, $R = .06$ for $n = 1000, 2000$, $R = .035$ for $n = 4000$, and $R = .02$ for $n = 10000$, and they also use 10% of the sensors as anchors and limit the degree for each node in order to maintain a low sparsity for the graph.

Tables 3.1, 3.2, and 3.3 shown later in this paper contain the results of our tests on noiseless problems. These tables contain the following information:

1. **# sensors, r, # anchors, and R:** We use $m = (\#anchors)$, $n = (\#sensors) + (\#anchors)$, and r to generate ten random instances of $p_1, \dots, p_n \in \mathbb{R}^r$; the last m points are taken to be the anchors. For each of these ten instances and for each value of the radio range $R > 0$, we generate the $n \times n$ partial EDM D_p according to

$$(D_p)_{ij} = \begin{cases} \|p_i - p_j\|^2 & \text{if } \|p_i - p_j\| < R \text{ or both } p_i \text{ and } p_j \text{ are anchors,} \\ \text{unspecified} & \text{otherwise.} \end{cases}$$

2. **# Successful Instances:** An instance was called *successful* if at least some, if not all, of the sensors could be positioned. If, by the end of the algorithm, the largest clique containing the anchors did not contain any sensors, then none of the sensor positions could be determined, making such an instance unsuccessful.
3. **Average Degree:** We have found that the average degree of the nodes of a graph is a good indicator of the percentage of sensors that can be positioned. In the results reported, we give the average of the average degree over all ten instances.
4. **# Sensors Positioned:** We give the average number of sensors that could be positioned over all ten instances. Note that below we indicate that the error measurements are computed only over the sensors that could be positioned.
5. **CPU Time:** This indicates the average running time of `SNLSDPclique` over all ten instances. This time does not include the time to generate the random problems, but it does include all aspects of Algorithm 1, including the time for `GrowCliques` and `ComputeFaces` at the beginning of the algorithm.
6. **Max Error:** This is the maximum distance between the positions of the sensors found and the true positions of those sensors. This is defined as

$$\text{Max Error} := \max_{i \text{ positioned}} \|p_i - p_i^{true}\|_2.$$

7. **RMSD:** This is the root-mean-square deviation (RMSD) of the positions of the sensors found and the true positions of those sensors. This is defined as

$$\text{RMSD} := \left(\frac{1}{\#\text{ positioned}} \sum_{i \text{ positioned}} \|p_i - p_i^{\text{true}}\|_2^2 \right)^{\frac{1}{2}}.$$

We note that for each set of ten random instances, the Max Error and RMSD values reported are only the average Max Error and average RMSD values over the successful instances; this is due to the fact that an unsuccessful instance will have no computed sensor positions to compare with the true sensor positions.

We have three sets of tests on noiseless problems as follows:

1. In Table 3.1 we report the results of using only the `RigidCliqueUnion` step (see Figure 2.2) to solve our random problems.
2. In Table 3.2 we report the results of increasing the level of our algorithm to use both the `RigidCliqueUnion` and `RigidNodeAbsorb` steps (see Figures 2.2 and 2.4) to solve the random problems. We see that the number of sensors localized has increased and that there has been a small, almost insignificant, increase in the CPU time.
3. In Table 3.3 we report the results of increasing the level of our algorithm to use steps `RigidCliqueUnion`, `RigidNodeAbsorb`, and `NonRigidCliqueUnion` (see Figures 2.2, 2.4, and 2.3, respectively) to solve the random problems, further increasing the class of problems that we can complete.

Testing a version of our algorithm that uses all four steps is still ongoing. From the above results, we can see that our facial reduction technique works very well for solving many instances of the SNL problem. We are confident that the results of our ongoing tests will continue to show that we are able to solve an even larger class of SNL problems.

3.2. Noisy data and higher dimensional problems. The above algorithm was derived based on the fact that the SNL had exact data, i.e., for a given clique

Table 3.1
Results of Algorithm 1 on noiseless problems, using step `RigidCliqueUnion`. *The values for Average Degree, # Sensors Positioned, and CPU Time are averaged over ten random instances. The values for Max Error and RMSD values are averaged over the successful instances.*

# sensors	r	# anchors	R	# Successful Instances	Average Degree	# Sensors Positioned	CPU Time	Max Error	RMSD
2000	2	4	.07	9/10	14.5	1632.3	1 s	6e-13	2e-13
2000	2	4	.06	5/10	10.7	720.0	1 s	1e-12	4e-13
2000	2	4	.05	0/10	7.5	0.0	1 s	-	-
2000	2	4	.04	0/10	4.9	0.0	1 s	-	-
4000	2	4	.07	10/10	29.0	3904.1	2 s	2e-13	6e-14
4000	2	4	.06	10/10	21.5	3922.3	2 s	6e-13	2e-13
4000	2	4	.05	10/10	15.1	3836.2	2 s	4e-13	2e-13
4000	2	4	.04	1/10	9.7	237.8	2 s	1e-13	4e-14
6000	2	4	.07	10/10	43.5	5966.9	4 s	3e-13	8e-14
6000	2	4	.06	10/10	32.3	5964.4	4 s	2e-13	7e-14
6000	2	4	.05	10/10	22.6	5894.8	3 s	3e-13	1e-13
6000	2	4	.04	10/10	14.6	5776.9	3 s	7e-13	2e-13
8000	2	4	.07	10/10	58.1	7969.8	6 s	3e-13	8e-14
8000	2	4	.06	10/10	43.0	7980.9	6 s	2e-13	8e-14
8000	2	4	.05	10/10	30.1	7953.1	5 s	6e-13	2e-13
8000	2	4	.04	10/10	19.5	7891.0	5 s	6e-13	2e-13
10000	2	4	.07	10/10	72.6	9974.6	9 s	3e-13	7e-14
10000	2	4	.06	10/10	53.8	9969.1	8 s	9e-13	1e-13
10000	2	4	.05	10/10	37.7	9925.4	7 s	5e-13	2e-13
10000	2	4	.04	10/10	24.3	9907.2	7 s	3e-13	1e-13
20000	2	4	.030	10/10	27.6	19853.3	17 s	7e-13	2e-13
40000	2	4	.020	10/10	24.7	39725.2	50 s	2e-12	6e-13
60000	2	4	.015	10/10	21.0	59461.1	1 m 52 s	1e-11	8e-13
80000	2	4	.013	10/10	21.0	79314.1	3 m 24 s	4e-12	1e-12
100000	2	4	.011	10/10	18.8	99174.4	5 m 42 s	2e-10	9e-11

α, we had an exact correspondence between the EDM and the corresponding Gram matrix $B = \mathcal{K}^{\dagger}(D[\alpha])$. To extend this to the noisy case, we apply a naive, greedy approach. When the Gram matrix B is needed, then we use the best rank r positive semidefinite approximation to B using the well-known Eckert–Young result; see, e.g., [16, Corollary 2.3.3].

Lemma 3.1. *Suppose that $B \in \mathcal{S}^n$ with spectral decomposition $B = \sum_{i=1}^n \lambda_i u_i u_i^T, \lambda_1 \geq \ldots \geq \lambda_n$. Then the best positive semidefinite approximation with at most rank r is $B_+ = \sum_{i=1}^r (\lambda_i)_+ u_i u_i^T$, where $(\lambda_i)_+ = \max\{0, \lambda_i\}$.*

We follow the multiplicative noise model in, e.g., [6, 8, 20, 23, 26, 28], i.e., the noisy (squared) distances D_{ij} are given by

$$D_{ij} = \left(\|p_i - p_j\|(1 + \sigma\epsilon_{ij})\right)^2,$$

where $\sigma \geq 0$ is the noise factor and ϵ_{ij} is chosen from the standard normal distribution $\mathcal{N}(0,1)$. We include preliminary test results in Table 3.4 for problems with 0–1% noise with embedding dimension $r = 2, 3$. Note that we do not apply the noise to the distances between the anchors.

TABLE 3.2

Results of Algorithm 1 *on noiseless problems, using steps* `RigidCliqueUnion` *and* `RigidNodeAbsorb`*. The values for Average Degree, # Sensors Positioned, and CPU Time are averaged over ten random instances. The values for Max Error and RMSD values are averaged over the successful instances.*

# sensors	r	# anchors	R	# Successful Instances	Average Degree	# Sensors Positioned	CPU Time	Max Error	RMSD
2000	2	4	.07	10/10	14.5	2000.0	1 s	6e-13	2e-13
2000	2	4	.06	10/10	10.7	1999.9	1 s	8e-13	3e-13
2000	2	4	.05	10/10	7.5	1996.7	1 s	9e-13	2e-13
2000	2	4	.04	9/10	4.9	1273.8	3 s	2e-11	4e-12
4000	2	4	.07	10/10	29.0	4000.0	2 s	2e-13	6e-14
4000	2	4	.06	10/10	21.5	4000.0	2 s	6e-13	2e-13
4000	2	4	.05	10/10	15.1	3999.9	2 s	6e-13	3e-13
4000	2	4	.04	10/10	9.7	3998.2	2 s	1e-12	5e-13
6000	2	4	.07	10/10	43.5	6000.0	4 s	3e-13	8e-14
6000	2	4	.06	10/10	32.3	6000.0	4 s	2e-13	7e-14
6000	2	4	.05	10/10	22.6	6000.0	3 s	3e-13	1e-13
6000	2	4	.04	10/10	14.6	5999.4	3 s	8e-13	3e-13
8000	2	4	.07	10/10	58.1	8000.0	6 s	3e-13	7e-14
8000	2	4	.06	10/10	43.0	8000.0	5 s	2e-13	8e-14
8000	2	4	.05	10/10	30.1	8000.0	5 s	6e-13	2e-13
8000	2	4	.04	10/10	19.5	8000.0	4 s	7e-13	2e-13
10000	2	4	.07	10/10	72.6	10000.0	9 s	3e-13	7e-14
10000	2	4	.06	10/10	53.8	10000.0	8 s	3e-13	1e-13
10000	2	4	.05	10/10	37.7	10000.0	7 s	5e-13	2e-13
10000	2	4	.04	10/10	24.3	10000.0	6 s	3e-13	1e-13
20000	2	4	.030	10/10	27.6	20000.0	17 s	7e-13	2e-13
40000	2	4	.020	10/10	24.7	40000.0	51 s	2e-12	6e-13
60000	2	4	.015	10/10	21.0	60000.0	1 m 53 s	2e-12	7e-13
80000	2	4	.013	10/10	21.0	80000.0	3 m 21 s	4e-12	1e-12
100000	2	4	.011	10/10	18.8	100000.0	5 m 46 s	2e-10	9e-11

TABLE 3.3

Results of Algorithm 1 *on noiseless problems, using steps* `RigidCliqueUnion`, `RigidNodeAbsorb`*, and* `NonRigidCliqueUnion`*. The values for Average Degree, # Sensors Positioned, and CPU Time are averaged over ten random instances. The values for Max Error and RMSD values are averaged over the successful instances. The results of the tests with more than* 6000 *sensors remain the same as in Table* 3.2*.*

# sensors	r	# anchors	R	# Successful Instances	Average Degree	# Sensors Positioned	CPU Time	Max Error	RMSD
2000	2	4	.07	10/10	14.5	2000.0	1 s	6e-13	2e-13
2000	2	4	.06	10/10	10.7	1999.9	1 s	8e-13	3e-13
2000	2	4	.05	10/10	7.5	1997.9	1 s	9e-13	2e-13
2000	2	4	.04	10/10	4.9	1590.8	5 s	2e-11	7e-12
4000	2	4	.07	10/10	29.0	4000.0	2 s	2e-13	6e-14
4000	2	4	.06	10/10	21.5	4000.0	2 s	6e-13	2e-13
4000	2	4	.05	10/10	15.1	3999.9	2 s	6e-13	3e-13
4000	2	4	.04	10/10	9.7	3998.2	3 s	1e-12	5e-13
6000	2	4	.07	10/10	43.5	6000.0	4 s	3e-13	8e-14
6000	2	4	.06	10/10	32.3	6000.0	4 s	2e-13	7e-14
6000	2	4	.05	10/10	22.6	6000.0	3 s	3e-13	1e-13
6000	2	4	.04	10/10	14.6	5999.4	3 s	8e-13	3e-13

Table 3.4

Results of Algorithm 1 for problems with noise and $r = 2, 3$, using `RigidCliqueUnion` *and* `RigidNodeAbsorb`. *The values for Average Degree, # Sensors Positioned, CPU Time, Max Error, and RMSD are averaged over ten random instances.*

σ	# sensors	r	# anchors	R	Average Degree	# Sensors Positioned	CPU Time	Max Error	RMSD
0	2000	2	4	.08	18.8	2000.0	1 s	1e-13	3e-14
1e-6	2000	2	4	.08	18.8	2000.0	1 s	2e-04	4e-05
1e-4	2000	2	4	.08	18.8	2000.0	1 s	2e-02	4e-03
1e-2	2000	2	4	.08	18.8	2000.0	1 s	2e+01	3e+00
0	6000	2	4	.06	32.3	6000.0	4 s	2e-13	7e-14
1e-6	6000	2	4	.06	32.3	6000.0	4 s	8e-04	3e-04
1e-4	6000	2	4	.06	32.3	6000.0	4 s	9e-02	3e-02
1e-2	6000	2	4	.06	32.3	6000.0	4 s	2e+01	3e+00
0	10000	2	4	.04	24.3	10000.0	6 s	3e-13	1e-13
1e-6	10000	2	4	.04	24.3	10000.0	6 s	5e-04	2e-04
1e-4	10000	2	4	.04	24.3	10000.0	6 s	5e-02	2e-02
1e-2	10000	2	4	.04	24.3	10000.0	7 s	4e+02	1e+02
0	2000	3	5	.20	26.6	2000.0	1 s	3e-13	8e-14
1e-6	2000	3	5	.20	26.6	2000.0	1 s	7e-04	2e-04
1e-4	2000	3	5	.20	26.6	2000.0	1 s	8e-02	2e-02
1e-2	2000	3	5	.20	26.6	2000.0	1 s	2e+03	4e+02
0	6000	3	5	.15	35.6	6000.0	5 s	3e-13	6e-14
1e-6	6000	3	5	.15	35.6	6000.0	5 s	1e-03	2e-04
1e-4	6000	3	5	.15	35.6	6000.0	5 s	1e-01	2e-02
1e-2	6000	3	5	.15	35.6	6000.0	6 s	9e+01	9e+00
0	10000	3	5	.10	18.7	10000.0	9 s	3e-12	2e-13
1e-6	10000	3	5	.10	18.7	10000.0	10 s	4e-02	2e-03
1e-4	10000	3	5	.10	18.7	10000.0	10 s	2e+00	8e-02
1e-2	10000	3	5	.10	18.7	10000.0	10 s	4e+02	1e+01

3.3. Comparison with existing algorithms. We now compare our running times and accuracy to the existing SDP-based algorithms. Currently, the Sparse Full SDP (SFSDP) method of [20, 21] and the Log-barrier Penalty Coordinate Gradient Descent (LPCGD) method of [23] are the most efficient SDP-based methods available. Before these methods came out, the Edge-based SDP (ESDP) method from [28] was the most efficient SDP-based algorithm for solving the sensor network localization problem.

First we note that most of the tests run in [21] and [23] have 10% of the nodes in the network as anchors and 90% as sensors. The reason for having so many anchors is that SDP-based methods typically require the sensors to be in the convex hull of the anchors in order to return good solutions. If the anchors are chosen randomly, this requirement can be met by having many anchors. However, we found that our `SNLSDPclique` algorithm is capable of returning very good solutions even with only a few randomly placed anchor nodes. Indeed, we ran the SFSDP code on random instances like those in our tests and found that sensors far away from the convex hull of the anchors were very poorly localized. Therefore, a side-by-side comparison is difficult since the SFSDP method requires many anchors, whereas we need only a few anchors.

Another major difference between the SFSDP and LPCGD methods and our `SNLSDPclique` algorithm is that they require very sparse problems in order to run efficiently (for example, edge-sparsification heuristics are used in [21] and [23] to speed up their computations), whereas we are able to handle problems with many distance constraints quite efficiently. In fact, we can see from Table 3.3 that having too few distance constraints can be a problem for us since we may not be able to localize all the sensors in the network. Again, this fact makes a side-by-side comparison of our algorithm with the SFSDP and LPCGD methods difficult.

From the results in [23], we see that the best variant of the LPCGD method requires about twice as much CPU time as the times reported in Table 3.2 for problems with up to $10,000$ nodes; moreover, the LPCGD method can attain RMSD values only on the order of 1e-3 for noiseless problems. However, we see in Table 3.2 that our `SNLSDPclique` algorithm attains RMSD values on the order of 1e-13 for noiseless problems, regardless of problem size. For noisy problems with a 1% noise factor, the LPCGD method again attains RMSD values on the order of 1e-3 (see [23]); however, from Table 3.4 we see that our `SNLSDPclique` algorithm can attain RMSD values only on the order of 1 to 100 for problems with a 1% noise factor.

From the latest results in [21], we see that the SFSDP method is able to attain RMSD values on the order of 1e-3 for two-dimensional problems with noise factors from 1–20%; however, the computation times are roughly six minutes for a problem with 6000 nodes with four anchors at the corners of $[0,1]^2$, three minutes for a problem with 9000 sensors and 1000 randomly placed anchors, and nine minutes for a problem with 18000 sensors and 2000 randomly placed anchors. The CPU times we obtain for the `SNLSDPclique` algorithm in Table 3.2 are orders of magnitude smaller than the running times for the SFSDP method, although the RMSD values we obtain for noisy problems in Table 3.4 are much larger than those reported in [21] for the SFSDP method, unless the noise factor is very small, say, less than 1e-6.

In conclusion, we find that our `SNLSDPclique` algorithm clearly outperforms the existing methods in terms of CPU time and accuracy on problems with very low noise. However, our method currently does not do anything special to handle noisy problems (for example, no refinement technique was used in our tests), and at this time we find that it is not competitive with the existing methods for accurately solving problems with medium to high noise. We feel confident that combining our facial reduction algorithm together with techniques for handling noise will produce a code that will be highly competitive in terms of both CPU time and solution accuracy.

4. Conclusion. The SDP relaxation of SNL is highly (implicitly) degenerate since the feasible set of this SDP is restricted to a low dimensional face of the SDP cone, resulting in the failure of the Slater constraint qualification (strict feasibility). We take advantage of this degeneracy by finding explicit representations of intersections of faces of the SDP cone corresponding to unions of intersecting cliques. In addition, from these representations we force further degeneracy in order to find the minimal face that contains the optimal solution. In many cases, we can efficiently compute the exact solution to the SDP relaxation *without* using any SDP solver.

In some cases it is not possible to reduce the problem down to a single clique. However, in these cases, the intersection of the remaining faces returned by `SNLSDPclique` will produce a face containing the feasible region of the original problem. This face can then be used to reduce the problem before passing the problem to an SDP solver, where, for example, the trace of the semidefinite matrix can be maximized [7] to try to keep the embedding dimension small. As an example, if the problem is composed of disjoint cliques, then Corollary 2.6 can be used to significantly reduce the problem size. This reduction can transform a large intractable problem into a much smaller problem that can be solved efficiently via an SDP solver.

REFERENCES

[1] S. Al-Homidan and H. Wolkowicz, *Approximate and exact completion problems for Euclidean distance matrices using semidefinite programming*, Linear Algebra Appl., 406 (2005), pp. 109–141.

[2] A. Alfakih, M. F. Anjos, V. Piccialli, and H. Wolkowicz, *Euclidean distance matrices, semidefinite programming, and sensor network localization*, Port. Math., to appear.

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

[4] B. Ames and S. A. Vavasis, *Nuclear Norm Minimization for the Planted Clique and Biclique Problems*, available online from http://arxiv.org/abs/0901.3348.

[5] P. Biswas, *Semidefinite Programming Approaches to Distance Geometry Problems*, Ph.D. thesis, Stanford University, Stanford, CA, 2007.

[6] 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. Autom. Sci. Eng., 3 (2006), pp. 360–371.

[7] P. Biswas, K.-C. Toh, and Y. Ye, *A distributed SDP approach for large-scale noisy anchor-free graph reailzation with applications to molecular conformation*, SIAM J. Sci. Comput., 30 (2008), pp. 1251–1277.

[8] P. Biswas, T.-C. Lian, T.-C. Wang, and Y. Ye, *Semidefinite programming for ad hoc wireless sensor network localization*, ACM Trans. Sen. Netw., 2 (2006), pp. 188–220.

[9] 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, Nonconvex Optim. Appl. 82, Springer, New York, 2006, pp. 69–84.

[10] M. W. Carter, H. H. Jin, M. A. Saunders, and Y. Ye, *SpaseLoc: An adaptive subproblem algorithm for scalable wireless sensor network localization*, SIAM J. Optim., 17 (2006), pp. 1102–1128.

[11] A. Cassioli, *Global Optimization of Highly Multimodal Problems*, Ph.D. thesis, Universita di Firenze, Firenze, Italy, 2008.

[12] J. Dattorro, *Convex Optimization & Euclidean Distance Geometry*, Meboo Publishing, Palo Alto, CA, 2005.

[13] Y. Ding, N. Krislock, J. Qian, and H. Wolkowicz, *Sensor network localization, Euclidean distance matrix completions, and graph realization*, Optim. Eng., 11 (2010), pp. 45–66.

[14] T. Eren, D. K. Goldenberg, W. Whiteley, Y. R. Yang, A. S. Morse, B. D. O. Anderson, and P. N. Belhumeur, *Rigidity, computation, and randomization in network localization*, in IEEE INFOCOM 2004—The Conference on Computer Communications, 23rd Annual Joint Conference of the IEEE Computer and Communications Societies, 4 (2004), pp. 2673–2684.

[15] U. Feige and R Krauthgamer, *Finding and certifying a large hidden clique in a semi-random graph*, Random Structures Algorithms, 16 (2000), pp. 195–202.

[16] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, MD, 1996.

[17] B. Hendrickson, *The Molecule Problem: Determining Conformation from Pairwise Distances*, Ph.D. thesis, Cornell University, Ithaca, NY, 1991.

[18] B. Hendrickson, *Conditions for unique graph realizations*, SIAM J. Comput., 21 (1992), pp. 65–84.

[19] H. Jin, *Scalable Sensor Localization Algorithms for Wireless Sensor Networks*, Ph.D. thesis, Toronto University, Toronto, Ontario, Canada, 2005.

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

[21] S. Kim, M. Kojima, H. Waki, and M. Yamashita, *A Sparse Version of Full Semidefinite Programming Relaxation for Sensor Network Localization Problems*, Technical Report B-457, Tokyo Institute of Technology, Oh-Okayama, Meguro, Tokyo, 2009.

[22] G. Pataki, *Geometry of semidefinite programming*, in Handbook of Semidefinite Programming: Theory, Algorithms, and Applications, H. Wolkowicz, R. Saigal, and L. Vandenberghe, eds., Kluwer Academic Publishers, Boston, MA, 2000.

[23] T. Pong and P. Tseng, *(Robust) Edge-based semidefinite programming relaxation of sensor network localization*, Math. Program., published online (2010).

[24] J. B. Saxe, *Embeddability of weighted graphs in k-space is strongly NP-hard*, in Proceedings of the 17th Allerton Conference on Communications, Control, and Computing, University of Illinois at Urbana-Champaign, 1979, pp. 480–489.

[25] A. M.-C. So and Y. Ye, *Theory of semidefinite programming for sensor network localization*, Math. Program., 109 (2007), pp. 367–384.

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

[27] R. J. Vanderbei and Y. Bing, *The simplest semidefinite programs are trivial*, Math. Oper. Res., 20 (1995), pp. 590–596.

[28] 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.

[29] H. Wolkowicz, *Explicit solutions for interval semidefinite linear programs*, Linear Algebra Appl., 236 (1996), pp. 95–104.