Noisy sensor network localization: robust facial reduction and the Pareto frontier*

Vris Yuen-Lam Cheung †‡ Dmitriy Drusvyatskiy *§ Nathan Krislock ¶
Henry Wolkowicz*

October 23, 2014

Key Words: Sensor network localization, Euclidean distance matrix completions, convex optimization, semidefinite programming

Abstract

We consider the localization problem in sensor networks where the inter-sensor distance measurements are inaccurate and incomplete. In this paper, we present two novel algorithms for large-scale sensor localization based on semidefinite programming relaxations. Emerging exoscale networks lead to semidefinite relaxations that are prohibitively large for interior-point methods to handle. Both of our methods are able to efficiently solve the semidefinite programming relaxations without the use of interior-point methods. Our first method works by relating cliques in the graph of the problem to faces of the positive semidefinite cone, allowing us to devise a combinatorial algorithm for the localization problem that is provably robust and parallelizable. Our second algorithm is a first order method for maximizing the trace—a popular low-rank inducing regularizer—in a robust formulation of the localization problem. Namely, we consider the related much easier problem where the trace objective and the robust constraint are interchanged. By solving a sequence of these easier problems, we are able to obtain a solution to the original max-trace problem. Both of our algorithms output a configuration of sensors that can serve as a high-quality initialization for local optimization techniques. We provide numerical experiments on large-scale sensor network localization problems illustrating the development.

1 Introduction

Given a network of sensors physically located in two or three dimensional space, the *sensor network localization* (or SNL for short) problem is to reconstruct the location of the sensors from incomplete and inexact pairwise distances. Typically, a pairwise distance between two sensors is approximately known when they are within a certain physical distance, called the radio range. Moreover, the exact location of some sensors (called anchors) is a priori known.

The SNL problem can be modeled as a nonconvex optimization problem. Semidefinite programming techniques have proven to be extremely useful for this problem; see for example [4–8, 16, 20, 23, 31]. It can be shown that under reasonable conditions [22], the SNL problem is indeed equivalent to a semidefinite program. For large networks, however, the semidefinite programs become intractable for off-the-shelf methods.

In the current work, we attempt to close the computational gap by focusing on combinatorial algorithms and efficient first-order methods. The starting point is the observation that the standard convex relaxation for the SNL problem becomes ill-conditioned as the noise in the distance measurements tends to zero [16],

*Research supported by Natural Sciences Engineering Research Council Canada and a grant from AFOSR.
†Department of Combinatorics and Optimization, Waterloo, Ontario N2L 3G1, Canada.
‡Department of Computer Science, University of Colorado, Boulder, CO 80309-0430, USA.
§Department of Mathematics, University of Washington, Seattle, WA 98195-4350, USA.
¶Department of Mathematical Sciences, Northern Illinois University, DeKalb, IL 60115, USA.

i.e., the strict feasibility fails. Krislock and Wolkowicz [16] effectively used this observation in a combinatorial algorithm that was able to solve SNL instances with *exact* distance measurements on hundreds of thousands of sensors on a conventional laptop in minutes.

An important caveat of this geometric approach is that near-exactness of the distance measurements is essential for their algorithm to work, both in theory and in practice. Indeed, the basic idea of the algorithm is that each clique in the graph describing the problem certifies that the entire feasible region lies in a certain face of the positive semidefinite cone. The method proceeds by exploring cliques and intersecting the corresponding faces, yielding in many cases a unique solution to the SNL problem. An immediate extension of this idea to the inexact setting does not work for the simple reason that randomly perturbed faces of the positive semidefinite cone typically intersect only at the origin. Remarkably, using dual information, we are able to design a method complementary to [16] for the SNL problem with noisy distance measurements. Under reasonable conditions, the algorithm is provably robust to noise in the sense that the output error is linearly proportional to the noise level in the distance measurements. Moreover, in contrast to the algorithm [16], the new method is in large part parallelizable even in the exact setting.

Next we explore the *Pareto search* strategy for the SNL problem, which targets sparse networks and therefore complements the robust facial reduction technique (which relies on the amount of cliques available in the networks). We consider maximizing the trace—a popular low-rank inducing regularizer [3,32]—in the robust formulation of the localization problem. Following the root finding strategy of [2,28,29], originating much earlier in portfolio optimization, we exchange the trace objective function and the robust constraint. We observe that this flipped problem is readily amenable to first-order methods since the nearest-point mapping to the feasible region is easy to compute. Applying a Newton type method to the value function (as was pioneered in the SPGL1 code [28–30] for basis pursuit), we use this easier flipped problem to solve the original.

We first formalize the SNL problem in the noiseless and noisy settings in Section 2, which also reviews the notion of *unique localizability*. In Section 3 we introduce the robust facial reduction technique for solving the noisy SNL problem, provide the framework (in Algorithm 1), discuss the implementation and provide some numerical results. Section 4 outlines the Pareto search technique, which primarily deals with sparse graphs and uses first-order optimization techniques.

2 Sensor network localization: problem statement

In this section, we formally state the SNL problem in the noiseless and in the noisy settings. A *noiseless* SNL problem boils down to a feasibility problem (as in (1) below), which is typically NP-hard because of the dimension constraint. A *noisy* SNL can be phrased as a rank- and cone-constrained least squares problem (as in (5)).

To model the SNL problem, throughout we fix an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ on a node set $\mathcal{V} = \{1, \dots, n\}$ and an edge set $\mathcal{E} \subset \{ij : 1 \le i < j \le n\}$. The vertices represent sensors in an r-dimensional space $\mathbb{R}^r$, while the presence of an edge ij joining the vertices i an j signifies that the physical distance between the sensors i and j is available.

2.1 Noiseless SNL problem

Given a set of squared distances d_{ij} indexed by the edges $ij \in \mathcal{E}$, the *noiseless SNL* problem asks to recover the configuration of sensors in space of a given dimension r, that is to find a collection of points $x_1, \dots, x_n$ satisfying

$$\|x_i - x_j\|^2 = d_{ij} \text{ for all } ij \in \mathcal{E}, \quad x_1, \dots, x_n \in \mathbb{R}^r. \tag{1}$$

The collection of points $x_1, \dots, x_n$ is then called a *localization* of the network. Usually there is in addition a distinguished subset of sensors, called anchors, whose location in space is a priori known. For ease of exposition, we will treat such anchors as regular sensors, throughout. This is fairly innocuous, since if the solution to the anchorless problem is unique up to orthogonal transformations, which is the usual setting, then we may first localize the anchorless network, and then easily translate and rotate the configuration of

sensors to align the anchors with their known positions. We comment further on incorporating anchors into our framework in the conclusion.

The main tool we use in the current work (even if indirectly) is semidefinite programming (SDP). To this end, let $\mathcal{S}^n$ denote the Euclidean space of $n \times n$ real symmetric matrices endowed with the trace inner product $\langle A, B \rangle = \operatorname{tr} AB$ and the Frobenius norm $\|A\|_F = \sqrt{\operatorname{tr} A^2}$. The convex cone of $n \times n$ positive semidefinite matrices will be denoted by $\mathcal{S}^n_+$. This cone defines a partial ordering: for any $A, B \in \mathcal{S}^n$ the binary relation $A \succeq B$ means $A - B \in \mathcal{S}^n_+$. We let $e \in \mathbb{R}^n$ denote the vector of all ones.

The noiseless SNL problem is equivalent to finding a matrix $X \in \mathcal{S}^n$ satisfying the system:

$$\left\{ \begin{array}{ll} X_{ii} + X_{jj} - 2X_{ij} = d_{ij} & \text{for all } ij \in \mathcal{E} \\ Xe = 0 & \\ \operatorname{rank} X \leq r & \\ X \succeq 0. & \end{array} \right\} \tag{2}$$

Indeed, suppose that X is feasible for (2). Then since X is positive semidefinite and has rank at most r, we may form a factorization $X = BB^T$ for some $n \times r$ matrix B. It is easy to verify that the rows of B yield a solution to the noiseless SNL problem (1). Conversely, if some points $x_1, \ldots, x_n \in \mathbb{R}^r$ localize the network, then we may center them around the origin and assemble them into the matrix $B = [x_1; \ldots; x_n]^T \in \mathbb{R}^{n \times r}$. The resulting Gram matrix $X := BB^T$ is feasible for the above system. For more details, see for example [16].

The formulation (2) is nonconvex, and indeed the noiseless SNL problem is in general NP-hard [21, 33]. A convex relaxation is obtained simply by ignoring the rank constraint yielding an SDP feasibility problem:

$$\left\{ \begin{array}{ll} X_{ii} + X_{jj} - 2X_{ij} = d_{ij} & \text{for all } ij \in \mathcal{E} \\ Xe = 0 & \\ X \succeq 0. & \end{array} \right\} \tag{3}$$

For many SNL instances, however, this convex relaxation is "exact" [22]. For example the following is immediate.

Observation 1 (Exactness of the relaxation)**.** *If the noiseless SNL problem* (1) *is feasible, then the following are equivalent:*

1. *No localization in $\mathbb{R}^l$, for $l > r$, spans the ambient space $\mathbb{R}^l$.*
2. *Any solution of the relaxation* (3) *has rank at most r and consequently any maximal rank solution of* (3) *yields a localization of the noiseless SNL.*

In theory, the exactness of the relaxation is a great virtue. From a computational perspective, however, the observation above shows that the SDP formulation (3) usually does not admit a positive definite solution, that is strict feasibility fails. Moreover, it is interesting to note that a very minor addition to the assumptions of Observation 1 implies that the SDP admits a unique solution [22]. We provide a quick proof for completeness, though the reader can safely skip it.

Observation 2 (Uniqueness of the solution)**.** *If the noiseless SNL problem* (1) *is feasible, then the following are equivalent:*

1. *The network does not admit a localization in $\mathbb{R}^{r-1}$, and moreover for any $l > r$ no localization in $\mathbb{R}^l$ spans the ambient space $\mathbb{R}^l$.*
2. *The relaxation* (3) *has a unique solution.*

Moreover, if either of the above conditions holds, then the SNL problem has a unique solution, up to a linear isometry.

Proof. The implication $2 \Rightarrow 1$ is immediate. To see the converse implication $1 \Rightarrow 2$, suppose that the SDP (3) admits two solutions X and Y. Combining the assumption 1 and Observation 1, we deduce that any solution of the SDP has rank at most r, and moreover both X and Y have rank exactly equal to r. Consider now the line $L := \{X + \lambda(Y - X) : \lambda \in \mathbb{R}\}$. Since the cone $\mathcal{S}^n_+$ is pointed, the line L is not fully contained in the feasible region. Consequently the line segment $L \cap \mathcal{S}^n_+$ has at least one endpoint, which must have rank at most $r - 1$. This is a contradiction since this endpoint yields a localization of the network in $\mathbb{R}^{r-1}$. □

In principle, one may now apply any off-the-shelf SDP solver to solve the problem (3). The effectiveness of such methods, however, depends heavily on the "conditioning" of the SDP system. In particular, if the system admits no feasible positive definite matrix, as is often the case (Observation 1), then no standard method can be guaranteed to perform very well nor be robust to perturbations in the distance measurements.

The authors of [16] found a way to use the degeneracy of the system (3) explicitly to design a combinatorial algorithm for the noiseless SNL problem. To describe the basic idea behind their procedure, we first recall that a convex subset $\mathcal{F}$ of the positive semidefinite cone $\mathcal{S}^n_+$ is called a *face* if it can be written as

$$\mathcal{F} = \left\{ U \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix} U^T : A \in \mathcal{S}^k_+ \right\}, \tag{4}$$

for some $n \times n$ orthogonal matrix U and some integer $k \in \{0, 1, \ldots, n\}$. In [16], the authors observed that each k-clique in the graph $\mathcal{G}$, with $k > r$, certifies that the entire feasible region (3) lies in a certain face $\mathcal{F}$ of the positive semidefinite cone $\mathcal{S}^n_+$. Therefore, the *facial reduction* technique of replacing $\mathcal{S}^n_+$ by a smaller set $\mathcal{F}$ can be applied on (3) to obtain an equivalent problem involving fewer variables. Their method explores cliques in the graph, while possibly growing them, and intersects pairwise such faces in a computationally effective way. Since typical graphs appearing in the SNL problem are fairly dense, there are many cliques to be found. For reasonable instances the procedure would terminate with a unique solution of the SNL problem. Since the algorithm is entirely combinatorial, noiseless SNL problems with 10^5 nodes could be solved on a conventional laptop in a few minutes. In contrast, no standard method for SDP can solve problems with $10^5 \times 10^5$ matrices, especially when strict feasibility fails, as is the case here.

2.2 Noisy SNL problem

In most applications, the inter-sensor distance measurements d_{ij} incorporate a certain degree of noise, and hence are inexact. One way to deal with the inexactness is to assume that the noise comes from some reasonable distribution (e.g., the multiplicative noise model [8]) and search for a maximum-likelihood estimator of the configuration. Such problem formulations lead to nonlinear SDPs that are difficult to solve [5, 8]—a usual trade-off between statistical motivation and computational efficiency. A different approach, more attune to computation, is to consider a rank-constrained least squares formulation:

$$\begin{array}{ll} \min\limits_{X} & \sum_{ij \in \mathcal{E}} |X_{ii} + X_{jj} - 2X_{ij} - d_{ij}|^2 \\ \text{s.t.} & Xe = 0 \\ & \operatorname{rank} X = r \\ & X \succeq 0. \end{array} \tag{5}$$

This is a smooth optimization problem with many local minima, and therefore nonlinear programming methods and manifold optimization techniques are highly sensitive to the initial point and often output irrelevant local minimizers. As a result most algorithms utilizing this framework have two stages: (1) obtain a good guess of the localization, and (2) refine the guess with a local nonlinear programming method. Most of the difficulty and novelty is in dealing with the first stage. A popular heuristic for obtaining an initial point for a local search method is to relax the rank constraint (as before), solve the least squares SDP, and project the resulting point onto the set of rank r matrices. For large scale problems, this approach is expensive and can perform poorly, though some techniques utilizing parallelism are available; for a discussion see [4]. Moreover, it seems desirable to avoid the use of expensive semidefinite programming in the first stage altogether. We propose a provably robust algorithm that does just that.

3 Robust facial reduction for noisy SNL

An immediate obstacle to the success of the facial reduction algorithm of [16] in the noisy setting is that the algorithm is highly unstable. The reason is simple: randomly perturbed faces of the semidefinite cone typically intersect only at the origin. Hence small perturbations in the distance measurements will generally lead to poor guesses of the face intersection arising from pairs of cliques. Moreover, even if pairs of cliques can robustly yield some facial information, the accumulated error compounds as the algorithm moves from clique to clique. The algorithm we propose revolves around the "dual" representations of faces. To this end, consider a face $\mathcal{F}$ of $\mathcal{S}^n_+$ in the "primal" form (4). Subdivide U now into two parts $U = \begin{bmatrix} U_R & U_N \end{bmatrix}$, where U_R has k columns and U_N has $n-k$ columns. Then $\mathcal{F}$ can equivalently be written as

$$\mathcal{F} = (U_N \cdot B \cdot U_N^T)^\perp \cap \mathcal{S}^n_+, \tag{6}$$

for any nonsingular matrix B in $\mathcal{S}^{n-r}_+$, where $\perp$ denotes the orthogonal complement with respect to the trace inner product. Going back and forth between primal (4) and dual (6) representations is easy, requiring a single eigenvalue decomposition, or more cheaply a QR factorization, for example. In general, if a face has the form $\mathcal{F} = Y^\perp \cap \mathcal{S}^n_+$ for some positive semidefinite matrix Y, then we say that Y *exposes* $\mathcal{F}$.

The salient feature of the dual representation is that it is much better adapted at handling noise. Before proceeding with the details of the proposed algorithmic framework, we provide some intuition. To this end, an easy computation shows that if Y_1 exposes a face $\mathcal{F}_1$ and Y_2 exposes a face $\mathcal{F}_2$, then the sum $Y_1 + Y_2$ exposes the intersection $\mathcal{F}_1 \cap \mathcal{F}_2$. Thus the faces $\mathcal{F}_1$ and $\mathcal{F}_2$ intersect trivially if and only if the sum $Y_1 + Y_2$ is positive definite. On the other hand, for the SNL problem if the true exposing vectors arising from the cliques are corrupted by noise, then one can round off the small eigenvalues of $Y_1 + Y_2$ (due to noise) to guess at the true intersection of the faces arising from the noiseless data.

3.1 The algorithmic framework

To describe our proposed algorithmic framework, we first need some notation. To this end, define the *Lindenstrauss mapping* $\mathcal{K} : \mathcal{S}^n \to \mathcal{S}^n$ by

$$\mathcal{K}(X)_{ij} := X_{ii} + X_{jj} - 2X_{ij},$$

and, for notational convenience, define the *centered* sets

$$\begin{aligned} \mathcal{S}^n_c &:= \{X \in \mathcal{S}^n : Xe = 0\}, \\ \mathcal{S}^n_{c,+} &:= \{X \in \mathcal{S}^n_+ : Xe = 0\}, \\ \mathcal{S}^{n,r}_{c,+} &:= \{X \in \mathcal{S}^n_{c,+} : \operatorname{rank} X \leq r\}. \end{aligned}$$

It is easy to see that $\mathcal{S}^n_{c,+}$ is a face of $\mathcal{S}^n_+$, and is linearly isomorphic to $\mathcal{S}^{n-1}_+$. Indeed, the matrix ee^T exposes $\mathcal{S}^n_{c,+}$. More specifically, for any $n \times n$ orthogonal matrix $\begin{bmatrix} \frac{1}{\sqrt{n}}e & U \end{bmatrix}$, we have the representation

$$\mathcal{S}^n_{c,+} = U\mathcal{S}^{n-1}_+ U. \tag{7}$$

Consequently, we now make the following important convention: the *ambient space* of $\mathcal{S}^n_{c,+}$ will always be taken as $\mathcal{S}^n_c$. The notion of faces of $\mathcal{S}^n_{c,+}$ and the corresponding notion of exposing matrices naturally adapts to this convention by appealing to (7) and the respective standard notions for $\mathcal{S}^{n-1}_+$. Namely, we will say that $\mathcal{F}$ is a face of $\mathcal{S}^n_{c,+}$ if it has the form $\mathcal{F} = U\widehat{\mathcal{F}}U^T$ for some face $\widehat{\mathcal{F}}$ of $\mathcal{S}^{n-1}_+$, and that a matrix Y exposes $\mathcal{F}$ whenever it has the form $U\widehat{Y}U^T$ for some matrix $\widehat{Y}$ exposing $\widehat{\mathcal{F}}$. Finally, for any convex subset $\Omega \subset \mathcal{S}^n_{c,+}$, the symbol $\operatorname{face}(\Omega; \mathcal{S}^n_{c,+})$ will denote the minimal face of $\mathcal{S}^n_{c,+}$ containing Ω.

We note that the *Moore-Penrose pseudoinverse* of $\mathcal{K}$ is easy to describe: for any matrix $D \in \mathcal{S}^n$ having all-zeros on the diagonal, we have

$$\mathcal{K}^\dagger(D) = \frac{1}{2} J \cdot D \cdot J,$$

where $J := I - \frac{1}{n}ee^T$ is the projection onto $e^\perp$. These and other related results have appeared in a number of publications; see for example [1, 13, 14, 17, 18, 24–27].

The following result (in a primal form) was the basis for the algorithm in [16]. An easier proof appears in the recent manuscript [10, Theorem 4.9].

Theorem 1 (One clique facial reduction)**.** *Consider a noiseless SNL instance on the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with distance measurements $d \in \mathbb{R}^{\mathcal{E}}$. Define the set*

$$\widehat{\Omega} \;\; := \;\; \{X \in \mathcal{S}^n_{c,+} : [\mathcal{K}(X)]_{ij} = d_{ij} \text{ for all } ij \in \mathcal{E}\}$$

of all the feasible solutions of (3)*. Suppose that the vertices $\{1, \ldots, k\}$ span a clique in $\mathcal{G}$. Let the matrix $\widehat{d} \in \mathcal{S}^k$ be the restriction of d to the edges between any vertices in $\{1, \ldots, k\}$. Then for any matrix $\widehat{Y}$ exposing* $\text{face}\big(\mathcal{K}^\dagger \widehat{d}; \mathcal{S}^k_{c,+}\big)$*,*

$$\text{the matrix} \quad \begin{bmatrix} \widehat{Y} & 0 \\ 0 & 0 \end{bmatrix} \quad \text{exposes} \quad \text{face}(\widehat{\Omega}; \mathcal{S}^n_{c,+}).$$

Thus any clique in the graph $\mathcal{G}$ certifies that the entire feasible region of the convex relaxation (3) to the noiseless SNL problem is contained in a certain face of the cone $\mathcal{S}^n_{c,+}$. If the clique has more than r vertices, then the corresponding face is guaranteed to be proper, meaning that it is strictly contained in $\mathcal{S}^n_{c,+}$. We can now state our proposed algorithmic framework for the noisy SNL problem, which works for general noisy Euclidean distance matrix (EDM) completion problem. (Recall that $D \in \mathcal{S}^n$ is a *Euclidean distance matrix* if there exist vectors $x_1, \ldots, x_n$ of the same length such that $D_{ij} = \|x_i - x_j\|_2^2$ for all i, j, or equivalently, if $D \in \mathcal{K}(\mathcal{S}^n_+)$.)

Algorithm 1 Basic strategy for Noisy EDM completion

INPUT: A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, noisy distance measurements $d \in \mathbb{R}^{\mathcal{E}}$, and a target rank $r \geq 0$;
OUTPUT: a Gram matrix X;
generate a set of cliques Θ in $\mathcal{G}$;
generate a set of weight functions $\{\omega_\alpha \colon \mathbb{R}^{\mathcal{E}} \to \mathbb{R}_+\}_{\alpha \in \Theta}$;
for each clique α in Θ **do**
  $k \leftarrow |\alpha|$;
  $X_\alpha \leftarrow$ a nearest matrix in $\mathcal{S}^{k,r}_{c,+}$ to $\mathcal{K}^\dagger d_\alpha$;
  $U_\alpha \leftarrow$ exposing vector of face$(X_\alpha, \mathcal{S}^k_{c,+})$;
end for
$U \leftarrow \sum_{\alpha \in \Theta} \omega_\alpha(d) \cdot U_\alpha$;
$Y \leftarrow$ a nearest matrix in $\mathcal{S}^{n,n-r}_{c,+}$ to U;
$X \leftarrow$ a solution of

$$\begin{array}{ll} \min\limits_{X} & \sum_{ij \in \mathcal{E}} |X_{ii} + X_{jj} - 2X_{ij} - d_{ij}|^2 \\ \text{s.t.} & X \in \mathcal{S}^n_{c,+} \cap Y^\perp; \end{array} \tag{8}$$

return X;

3.2 Robustness guarantees

In this section, we outline a rudimentary robustness guarantee of the facial reduction method given in Algorithm 1. To this end, consider a noiseless SNL problem on the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with distance measurements d, and a target embedding dimension r. For any subgraph $\mathcal{H}$ of $\mathcal{G}$, we let $d[\mathcal{H}]$ denote the restriction of d to $\mathcal{H}$. Following [22], the noisy SNL problem is said to be *uniquely r-localizable* if either of the equivalent conditions in Observation 2 holds.

In what follows, fixing a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we let $Alg(d)$ be the output of Algorithm 1 on the SNL instance with distance measurements $d \in \mathbb{R}^{\mathcal{E}}$.

Theorem 2 (Robustness)**.** *Consider an SNL instance on the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with noiseless distance measurements $d \in \mathbb{R}^{\mathcal{E}}$. Suppose the following:*

- *for any clique $\alpha \in \Theta$, the subnetwork on α is uniquely r-localizable; and*
- *the noiseless SNL instance is uniquely r-localizable and Alg(d) is the localization.*

Then there exist constants $\varepsilon > 0$ and $\kappa > 0$ so that

$$\|Alg(d + \Delta d) - Alg(d)\|_F \leq \kappa \|\Delta d\|_F \text{ whenever } \|\Delta d\|_F < \varepsilon.$$

The proof is standard but lengthy. We refer the interested reader to the followup full length paper.

3.3 Implementing facial reduction for noisy EDM

In the next few subsections, we elaborate on the some of the main ingredients of Algorithm 1:

- the choice of sets of cliques Θ and weight functions $\{\omega_\alpha\}_{\alpha \in \Theta}$ (in Section 3.3.1);
- the nearest-point mapping to $\mathcal{S}^{k,r}_{c,+}$ (in Section 3.3.2); and
- the solution of the facially reduced least squares problem (8) (in Section 3.3.3).

To improve the solution quality of Algorithm 1, we also use preprocessing and postprocessing steps. The preprocessing involves refining the clique set Θ and ensures that the exposing vector U computed in Algorithm 1 has exactly $r + 1$ eigenvalues that are very close to zero; see Section 3.3.4. The postprocessing is the use of a nonlinear optimization technique to find a local solution of (5) using the solution X from Algorithm 1 as the initial point. Such a *local refinement* procedure, which by itself often fails to find a global optimal solution, can greatly improve the solution quality of Algorithm 1.

3.3.1 Choosing the clique set and the weights

The clique set Θ is crucial for the success of Algorithm 1, since the exposing vector Y is formed based on the clique information. In practice, it is inefficient to get the set of *all* cliques of $\mathcal{G}$ (noting that in general, determining whether a graph has a clique of an arbitrary given size is NP-hard), so we can only hope to find a subset of cliques of the graph.

We use a simple and fast brute-force subroutine applied on the adjacency matrix H of the given graph to find a collection Θ of cliques. First, for each vertex v, we use a greedy heuristic to find a sufficiently large all-ones principal minor of $H(\delta_v, \delta_v) + I$ (where $\delta_v := \bigcup\{u \in \mathcal{V} : uv \in \mathcal{E}\}$), which would give a clique covering v; let Θ_1 be the set of distinct cliques found in this manner. Then we use a brute-force method to find a collection Θ_k of k-cliques α of $\mathcal{G}$ such that α is not contained in any clique in Θ_1, for $k = 2$ up to a user-defined maximum size: for $k = 2$ we collect all the edges that are not in any clique in Θ_1, and for each $k \geq 3$, we consider each of the $k - 1$ cliques and determine if the vertices in that clique have common neighbors. This results in a collection of cliques $\Theta := \bigcup_k \Theta_k$.

The first step of computing Θ_1 is very fast since it involves only repeatedly removing rows and columns of $H(\delta_v, \delta_v) + I$ that contain zero. As for the second step, while the brute-force method of listing all cliques of fixed sizes would be prohibitive in practice, we find that the restriction imposed by Θ_1 cuts down a huge number of smaller non-maximal cliques that we need to keep track of, and the use of Θ_1 significantly speeds up the second step.

Another feature of Algorithm 1 is that we do not treat each clique in Θ equally, given that the noise in the distance measurements does not have to be uniform and it may not be possible to recover all the cliques with the same level of accuracy. We gauge the amount of noise present in the distance measurements of cliques as follows: for each clique $\alpha \in \Theta$, letting $\hat{d} \in \mathcal{S}^\alpha$ be the restriction of the distance measurements d in the clique, we estimate the noise present in $\hat{d}$ by considering the eigenvalues of $\mathcal{K}^\dagger \hat{d}$:

$$\nu_\alpha(d) := \frac{\sum_{j=1}^{|\alpha|-r} \lambda_j(\mathcal{K}^\dagger \hat{d})^2}{0.5|\alpha|(|\alpha| - 1)}. \tag{9}$$

In the case where no noise is present in the distance measurements d, we have $\nu_\alpha(d) = 0$ since the matrix $\mathcal{K}^\dagger \hat{d} \in \mathcal{S}_+^\alpha$ is of rank r. To each clique α, we assign the weight

$$\omega_\alpha(d) := \frac{\sum_{\beta\in\Theta} \nu_\beta(d) - \nu_\alpha(d)}{\sum_{\beta\in\Theta} \nu_\beta(d)}.$$

A rationale behind this choice of weight is to reflect the contribution of noise in the clique α to the total noise of all cliques (where the noise is measured by (9)). If a clique α is relatively noisy compared to other cliques in Θ, the weight $\omega_\alpha(d)$ would be smaller than $\omega_\beta(d)$ for most $\beta \in \Omega$.

3.3.2 Nearest-point mapping to $\mathcal{S}_{c,+}^{k,r}$

We now describe how to evaluate the nearest-point-mapping to the set $\mathcal{S}_{c,+}^{k,r}$—an easy and standard operation due to the Eckart-Young Theorem. To describe this operation, consider any matrix $X \in \mathcal{S}^n$ and a set $\mathcal{Q} \subset \mathcal{S}^n$. Define the *distance function* and the *projection*, respectively:

$$\operatorname{dist}(X;\mathcal{Q}) = \inf_{Y\in\mathcal{Q}} \|X - Y\|_F,$$

$$\operatorname{proj}(X;\mathcal{Q}) = \{Y \in \mathcal{Q} : \|X - Y\|_F = \operatorname{dist}(X;\mathcal{Q})\}.$$

In this notation, we would like to find a matrix Y in the set $\operatorname{proj}(X;\mathcal{S}_{c,+}^{k,r})$. To this end, let $\begin{bmatrix}\frac{1}{\sqrt{k}}e & U\end{bmatrix}$ be any $k \times k$ orthogonal matrix. First dealing with the centering constraint, one can verify

$$\operatorname{proj}\left(X;\mathcal{S}_{c,+}^{k,r}\right) = U\Big[\operatorname{proj}\left(U^T X U;\mathcal{S}_+^{k-1,r}\right)\Big]U^T.$$

On the other hand, we have

$$\operatorname{proj}\left(Z;\mathcal{S}_+^{k-1,r}\right) = W \operatorname{Diag}\Big(0,\ldots,0,\lambda_{k-r}^+(Z),\ldots,\lambda_{k-1}^+(Z)\Big)W^T,$$

where $\lambda_1(Z) \le \ldots \le \lambda_{k-1}(Z)$ are the eigenvalues of Z and the subscript $\lambda_i^+(Z)$ refers to their positive part, and W is any orthogonal matrix in the eigenvalue decomposition $Z = W\operatorname{Diag}(\lambda(Z))W^T$. Thus computing a matrix in $\operatorname{proj}(X;\mathcal{S}_{c,+}^{k,r})$ requires no more than an eigenvalue decomposition.

3.3.3 Solving the small least squares problem

We now describe how to easily solve the least squares system (8). Typically, the matrix Y will have rank $n - r$. Then the face $\mathcal{S}_{c,+}^n \cap Y^\perp$ can be written as $\mathcal{S}_{c,+}^n \cap Y^\perp = U\mathcal{S}_+^r U^T$, where the $n \times r$ matrix U has as columns an orthonormal basis for the kernel of Y. Consequently we are interested in solving an optimization problem of the form

$$\begin{aligned}\min_Z \ & \|\mathcal{A}(Z) - d\|_2^2 \\ \text{s.t.} \ & Z \in \mathcal{S}_+^r,\end{aligned}$$

where the linear operator $\mathcal{A}\colon \mathcal{S}^n \to \mathbb{R}^{\mathcal{E}}$ is defined by $[\mathcal{A}(Z)]_{ij} = [\mathcal{K}(UZU^T)]_{ij}$ for all $ij \in \mathcal{E}$. Let $\operatorname{svec}(Z)$ be the vectorization of Z and let A be a $|\mathcal{E}| \times \frac{r(r+1)}{2}$ matrix representation of the operator $\mathcal{A}$. Thus we are interested in solving the system

$$\begin{aligned}\min_Z \ & \|A(\operatorname{svec} Z) - d\|_2^2 \\ \text{s.t.} \ & Z \in \mathcal{S}_+^r,\end{aligned} \tag{10}$$

where A is a tall-skinny matrix. One approach now is simply to expand the objective

$$\|A(\operatorname{svec} Z) - d\|_2^2 = \langle (A^T A)(\operatorname{svec} Z), \operatorname{svec} Z\rangle - 2\langle A^* d, \operatorname{svec} Z\rangle + \|d\|^2,$$

and then apply any standard iterative method to solve the problem (10). Alternately, we may first form an economic QR factorization $A = QR$ (where $Q \in \mathbb{R}^{|\mathcal{E}|\times\frac{1}{2}r(r+1)}$ has orthonormal columns and $R \in \mathbb{R}^{\frac{1}{2}r(r+1)\times\frac{1}{2}r(r+1)}$ is upper triangular) and then write the objective as $\|A(\operatorname{svec} Z)-d\|_2^2 = \|R(\operatorname{svec} Z)-Q^Td\|^2$. We can then pose the problem (10) as a small linear optimization problem over the product of the semidefinite cone $\mathcal{S}_+^r$ and a small second-order cone of dimension $\mathbb{R}^{\frac{r(r+1)}{2}}$, and quickly solve it by an off-the-shelf Interior Point Method.

In practice, very often the cone constraint in (8) is *inactive*. Heuristically, we can simply drop the cone constraint in (8) and consider the unconstrained least squares problem

$$\min_Z \ \|A(\operatorname{svec} Z) - d\|_2^2, \tag{11}$$

which with the help of economic QR can be solved very efficiently. If the unique solution of (11) is positive definite (which is often the case), then it is immediately the unique solution of (8). With this observation, we often can solve (8) *without using any optimization software*.

3.3.4 Preprocessing: sequential clique union

To increase the robustness of Algorithm 1, our implementation includes a preprocessing step, which finds an ordering to the cliques in the set Θ in Algorithm 1 and performs a *sequential clique union* procedure.

Specifically, we would like an ordering $\Theta = \{\alpha_1, \ldots, \alpha_{|\Theta|}\}$ such that

$$\alpha_{j-1} \text{ and } \alpha_j \text{ intersect in at least 2 vertices, } \ \forall\, j. \tag{12}$$

Such an ordering can be found using a greedy approach: start with the largest clique $\alpha_1 \in \Theta$ and $\widehat{\Theta} := \Theta$, and for each $j \geq 2$, pick α_j among all the cliques in $\widehat{\Theta}$ intersecting with α_{j-1} in at least 2 vertices, the one that maximizes the set difference $|\alpha\backslash\alpha_{j-1}|$; then update $\widehat{\Theta}$ by removing from it all the cliques whose nodes are covered by $\bigcup_{l=1}^{j}\alpha_l$.

If the graph $\mathcal{G}$ is sparse, such an ordering may not exist. Nonetheless, as long as $\mathcal{G}$ does not have a cut vertex, it is possible to cover all the vertices of $\mathcal{G}$ with multiple sequences of cliques, each satisfying the condition (12). (Note that if a noiseless SNL instance is r-uniquely localizable for any $r \geq 2$, then the corresponding graph cannot have a cut vertex.) In the following we discuss the *clique union* technique under the assumption that an ordering of the elements of Θ satisfying (12) exists; the technique can be easily extended even when such an assumption does not hold.

Then we would perform a sequential clique union procedure, whose goals are to ensure that the matrix U found in Algorithm 1 is not too far from $\mathcal{S}_{c,+}^{n,n-r}$, and to avoid errors arising from (nearly) nonrigid intersection, which we illustrate in the following example.

Example 3.1. Suppose that we have 5 sensors with radio range 0.05, whose true locations are given by

$$P = \begin{bmatrix} 0.4582 & 0.4793 & 0.5031 & 0.4360 & 0.4467 \\ -0.4116 & -0.3952 & -0.3221 & -0.3150 & -0.3393 \end{bmatrix}^T.$$

Then the corresponding graph is as in the left picture in Figure 1: only the distance between sensors 1 and 5 is missing. The graph has two cliques $\alpha_1 = \{1, 2, 3, 4\}$ and $\alpha_2 = \{2, 3, 4, 5\}$. Sensors $3, 4, 5$ in the clique intersection are almost collinear; α_1 and α_2 almost intersect *nonrigidly*, in the sense that the realization of sensor locations:

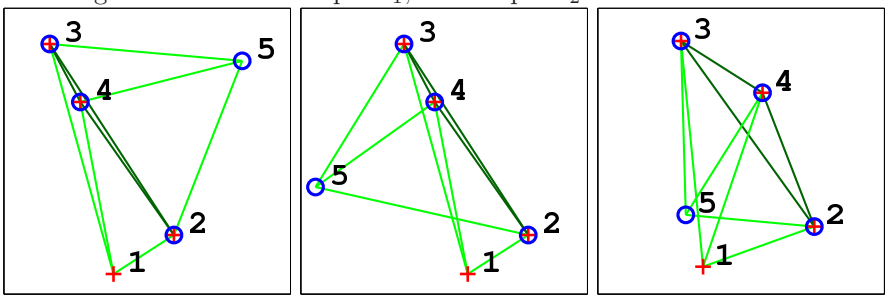
$$\tilde{P} = \begin{bmatrix} 0.4582 & 0.4793 & 0.4360 & 0.4467 & 0.4051 \\ -0.4116 & -0.3952 & -0.3150 & -0.3393 & -0.3750 \end{bmatrix}^T$$

obtained by reflecting α_1 along the line passing through vertices 2 and 3 would give almost the same partial EDM:

$$\mathcal{P}\big(\mathcal{K}(PP^T) - \mathcal{K}(\tilde{P}\tilde{P}^T)\big) \approx 6.84 \times 10^{-4},$$

where $\mathcal{P} : \mathcal{S}^n \to \mathbb{R}^{\mathcal{E}}$ is the canonical projection. (See the middle picture in Figure 1 for the sensor locations depicted by $\tilde{P}$.) In the presence of uncertainty in distance measurements, both P and $\tilde{P}$ seem to be reasonable realization of the sensor locations. Yet with the additional knowledge of the radio range, we know that it is unlikely $\tilde{P}$ gives the approximate sensor locations, since that would mean sensors 1 and 5 are in each other's radio range.

Figure 1: Left: true location of 5 sensors (given by P), with edges indicating known distances. Center: realization of sensor location (given by $\tilde{P}$) satisfying the known distances but violating the radio range. Right: solution from Algorithm 1. Blue: clique α_1; red: clique α_2.



Now suppose that the distance measurements are corrupted with 5% Gaussian noise (see the multiplicative noise model outlined in Algorithm 2). If we apply Algorithm 1 on the noisy input, then the realization of sensor locations could be as in the right picture in Figure 1, where sensors 1 and 5 are much closer than they should be. Note that the right picture in Figure 1 shows a minor perturbation of the "incorrect" realization $\tilde{P}$.

Scenarios depicted in Example 3.1 can be quite prevalent; to ensure the robustness of the facial reduction algorithm, we perform a clique union on α_{j-1} and α_j (for each $j = 2, \ldots, |\Theta|$), by using a Procrustes rotation to match the cliques α_{j-1}, α_j at the intersection. After we obtain a realization of $\alpha_{j-1} \cup \alpha_j$, we use that realization to compute an exposing vector corresponding to $\alpha_{j-1} \cup \alpha_j$, and replace α_{j-1}, α_j in Θ by $\alpha_{j-1} \cup \alpha_j$.

3.3.5 Postprocessing: local refinement

Following Algorithm 1, we implement a *local refinement*, which could greatly improve the solution quality. By local refinement, we mean the use of a nonlinear optimization technique for solving the nonconvex problem (5) (which has a lot of local minima) using the output of Algorithm 1 as the initial point. Local refinement has been commonly used for SDP-based algorithm for SNL problems and noisy EDM completion problem; see [4, 6].

For local refinement, we use the `Manopt`, a MATLAB package for optimization on manifolds (also known as Riemannian optimization) [9], which is well-suited for solving rank-constrained optimization problems like (5). The Riemannian optimization techniques have been previously used to solve *noiseless* Euclidean distance matrix completion problem [19]. Specifically, we use a Riemannian trust region method to solve

$$\min_{P \in \mathbb{R}^{(n-1)\times 2}} \|\mathrm{proj}\,(\widehat{\mathcal{K}}(PP^T) - D)\|_2^2 \tag{13}$$

(which is equivalent to (5)). By itself, the Riemannian trust region method, like most nonlinear optimization techniques, usually fails to find the *global* optimal solution of (13) and instead gets trapped at one of the many critical points (as we can see in Table 1 in Section 3.4 on numerical results), since the problem is highly nonconvex. Yet by providing the trust region with a "good" initial point (obtained from Algorithm 1), we can greatly improve the performance of the Riemannian trust region method applied on (13), as illustrated in the numerical results.

3.4 Numerics

Now we present some numerical results of applying Algorithm 1 to randomly generated instances, based on a *multiplicative noise model* (see e.g. [4,5]) outlined below.

Algorithm 2 Multiplicative noise model

INPUT: # sensors n, noise factor σ, radio range R;
For each $i, j = 1, \ldots, n$:
- pick i.i.d. $p_i \in [-0.5, 0.5]^2$ with uniform distribution
- pick i.i.d. $\epsilon_{ij} \in \mathcal{N}(0,1)$ (standard normal distribution)

Compute $D \in \mathcal{S}^n$ by

$$D_{ij} = (1 + \sigma\epsilon_{ij})^2 \|p_i - p_j\|_2^2, \quad \forall\, i, j = 1, \ldots, n;$$

Build graph $\mathcal{G} = (\{1, \ldots, n\}, \mathcal{E})$, where

$$ij \in \mathcal{E} \iff \|p_i - p_j\|_2 \leq R;$$

$d \leftarrow [D_{ij}]_{ij \in \mathcal{E},\, i<j} \in \mathbb{R}^{\mathcal{E}}$;
OUTPUT: noisy distance measurements $d \in \mathbb{R}^{\mathcal{E}}$ and graph $\mathcal{G}$.

Since instances generated by the multiplicative noise model come with the true sensor locations, we can gauge the performance of the robust facial reduction on random instances from the multiplicative noise model using the *root mean standard deviation* (RMSD): given the true locations $P \in \mathbb{R}^{n\times 2}$ of the sensors, the RMSD of the estimated locations $\tilde{P} \in \mathbb{R}^{n\times 2}$ of the sensors estimates how far the estimated locations are from the true locations:

$$RMSD := \frac{1}{\sqrt{n}} \|P - \tilde{P}\|_F,$$

where $\|\cdot\|_F$ is the Frobenius norm of a matrix (i.e., the ℓ_2 norm of the vectorization of that matrix).

A typical output of Algorithm 1 applied on an instance generated by the multiplicative noise model is illustrated in Figure 2, for an instance with 1000 sensors *and no anchor*, with noise factor 0.05 and radio range $R = 0.1$. The left figure shows the localization before local refinement (with RMSD being $61.52\%R$), and the center figure shows the localization after local refinement (with RMSD being $1.39\%R$). While the solution produced by Algorithm 1 may not seem very impressive, with the help of standard local refinement techniques we can attain very high quality solution even with the high number of sensors and the presence of noise. The right figure shows in comparison the optimal solution of (5) obtained using `Manopt` starting at a (uniformly) randomly generated initial point.

Table 1 shows some numerical results on instances with 1000 sensors (and no anchors) generated as in Algorithm 2, with varying number of sensors, noise factor and radio range. The tests were run on MATLAB version R2014a, on a Linux machine with Intel(R) Xeon(R) CPU E5620 @ 2.40 GHz and 46.76 GB RAM. We show the RMSD (as a percentage of the radio range) of the solutions provided by Algorithm 1 (in the column "before refinement"), and also the RMSD of the solution after the local refinement using `Manopt` (in the column "after refinement"). For comparison, we run `Manopt` on the same instances with a random initial point; their RMSD values are shown in the column `Manopt`. We see that using Algorithm 1 together with local refinement (which by itself does not perform very well) gives rather satisfactory results. The time used by Algorithm 1, including the selection of cliques and computation of the exposing vectors, is very little considering the size of the problem instances, usually around 1 minute.

Table 2 shows some numerical results on larger instances.

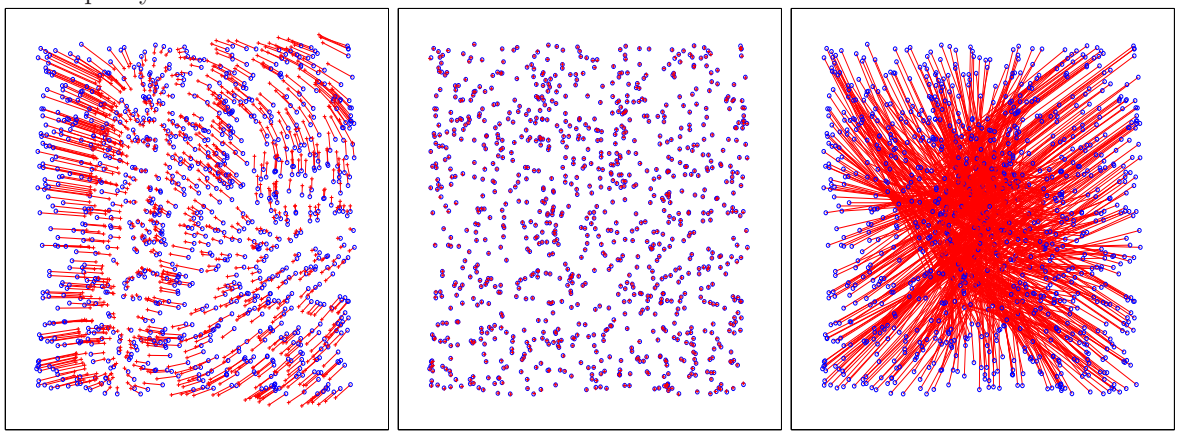
Table 1: Numerical results of robust facial reduction on 17 instances generated using the multiplicative noise model on a $[-0.5, 0.5]^2$ grid. n: number of sensors; R: radio range; nf: noise factor; m: number of edge.

Instance	n	nf	R	m	Time used by Alg. 1 (s)	RMSD $\%R$		
						before refinement	after refinement	`Manopt`
1016-1018	1000	0.1	0.15	31078	15.3	12.21	1.42	380.11
1016-1019	1000	0.1	0.15	30474	15	12.15	1.39	586.8
1016-1145	1000	0.1	0.15	30208	14.7	9.82	1.49	546.29
1016-1146	1000	0.1	0.15	31521	15.7	11	1.4	616.8
1016-1153	1000	0.1	0.15	30861	15.3	16.19	1.44	417.78
1016-1155	1000	0.1	0.15	30477	14.8	14.22	1.43	356.94
1016-1156	1000	0.1	0.15	30980	15.2	12.65	1.5	526.49
1016-1200	1000	0.1	0.15	31051	15	10.04	1.45	300.67
1016-1211	1000	0.15	0.15	30808	15.1	108.96	2.22	568.18
1016-1223	1000	0.15	0.15	31605	15.5	74.99	9.22	621.44
1016-1235	1000	0.15	0.15	30229	15	153.67	2.18	527.74
1016-1241	1000	0.15	0.15	31110	15.5	79.07	2.18	536.26
1016-1248	1000	0.15	0.15	31516	15.4	96.88	27.64	548.19
1016-1320	1000	0.2	0.15	31547	15	115.15	10.94	386.3
1016-1338	1000	0.2	0.15	31093	15.9	111.61	18.66	544.48
1016-1341	1000	0.2	0.15	31390	15.5	174.23	25.74	570.42
1016-1347	1000	0.2	0.15	30939	15.9	163.79	3.1	216.95

Table 2: Numerical results of robust facial reduction on 27 instances generated using the multiplicative noise model on a $[-0.5, 0.5]^2$ grid. n: number of sensors; R: radio range; nf: noise factor; m: number of edge.

Instance	n	nf	R	m	Time used by Alg. 1 (min)	RMSD $\%R$	
						before refinement	after refinement
1016-1442	2000	0.05	0.1	57620	0.79	10.57	0.75
1016-1451	2000	0.05	0.1	57347	0.26	11.05	0.76
1016-1638	2000	0.05	0.1	57821	0.76	10.03	0.76
1016-1646	2000	0.05	0.1	57309	0.75	24.91	0.78
1016-1652	2000	0.05	0.1	57170	0.73	7.91	0.76
1016-1657	2000	0.05	0.1	57522	0.8	13.27	0.84
1016-1701	2000	0.05	0.1	57492	0.73	12.7	0.77
1016-1713	2000	0.05	0.1	57685	0.76	9.67	0.76
1016-1717	2000	0.05	0.1	58146	0.79	9.53	0.78
1016-1724	2000	0.05	0.1	57855	0.75	9.14	0.73
1016-1729	2000	0.05	0.1	56819	0.74	10.98	0.78
1016-1855	2000	0.05	0.07	29164	0.83	37.71	1.32
1016-1925	2000	0.05	0.07	28558	0.86	47.53	1.17
1016-1939	2000	0.05	0.07	29163	0.9	32.59	1.26
1016-2045	2000	0.05	0.07	29248	1.05	58.22	1.3
1016-2058	2000	0.05	0.07	28537	0.83	54.23	1.18
1016-2119	2000	0.05	0.07	28846	0.86	63.82	1.19
1016-2134	2000	0.05	0.07	28371	0.83	55.96	1.24
1016-2109	2000	0.05	0.07	28713	0.82	33.58	1.27
1019-94	5000	0.05	0.05	94107	10.22	116.31	1.12
1019-428	5000	0.05	0.05	94085	9.83	53.11	1.07
1019-122	5000	0.05	0.05	93563	9.83	48.58	1.05
1018-224	5000	0.05	0.05	94059	10.03	44.98	1.05
1020-1115	8000	0.15	0.1	922111	38.91	58.61	1.04
1020-1525	8000	0.15	0.1	930180	39.26	75.39	1.06
1020-174	8000	0.15	0.1	921351	39.11	100.05	1.04
1020-1856	8000	0.15	0.1	921946	40.94	92.84	1.04

Figure 2: Illustration of robust facial reduction with refinement applied on an instance with 1000 sensors (no anchors) on a $[-0.5, 0.5]^2$ box, with noise factor 0.05 and radio range 0.1. From left to right: (1) using Algorithm 1 without refinement (RMSD= $61.52\%R$); (2) using Algorithm 1 with refinement via `Manopt` (RMSD= $1.39\%R$); (3) using only `Manopt` (RMSD= $380.59\%R$). Blue: true location; red: estimated location and discrepancy.



4 The Pareto frontier of the unfolding heuristic

4.1 Robust formulation of the noisy SNL

Another common approach to the noisy SNL problem is to consider the optimization problem in the robust form:

$$\begin{aligned} \max_X \quad & \operatorname{tr} X \\ \text{s.t.} \quad & \frac{1}{2}\|\mathcal{P} \circ \mathcal{K}(X) - d\|_2^2 \leq \sigma \\ & Xe = 0 \\ & X \succeq 0,\ X \in \mathcal{S}^n, \end{aligned} \tag{14}$$

where $\mathcal{P}$ is the canonical projection $\mathcal{P}\colon \mathcal{S}^n \to \mathbb{R}^{\mathcal{E}}$. Here, the tolerance $\sigma > 0$ is typically known and the Frobenius norm can be replaced by another measure of deviation depending on context. For simplicity we restrict to the Frobenius norm in the discussion. Trace maximization encourages the solution X to have a lower rank. This is in contrast to the usual min-trace strategy in compressed sensing; see [2, 28, 29] for a discussion. We focus on the max-trace regularizer for concreteness, though an entirely analogous analysis holds for the min-trace.

We propose a first-order method for this problem using a Pareto search strategy originating in portfolio optimization. This technique has recently garnered much attention in wider generality; e.g. [28–30]. The idea is simple: exchange the objective and the difficult constraint, and then use the easier flipped problem to solve the original. Thus we are led to consider the parametric optimization problem

$$\begin{aligned} \varphi(\tau) := \min_X \quad & \frac{1}{2}\|\mathcal{P} \circ \mathcal{K}(X) - d\|_2^2 \\ \text{s.t.} \quad & \operatorname{tr} X \geq \tau \\ & Xe = 0 \\ & X \succeq 0. \end{aligned}$$

To solve the original problem (14), we simply need to find a real number τ satisfying $\varphi(\tau) = \sigma$. This is a one-dimensional root finding problem and can be solved with Newton's method, provided that we can

compute the derivative $\varphi\prime(\tau)$. Namely we iterate:

$$\tau_{k+1} \leftarrow \tau_k + \frac{\sigma - \varphi(\tau_k)}{\varphi\prime(\tau_k)};$$

see [2] for more details.

To illustrate, let $V \in \mathbb{R}^{n\times(n-1)}$ be any matrix satisfying

$$V \text{ full rank and } \operatorname{range}(V) = e^{\perp}. \tag{15}$$

Then the main problem (14) has the form

$$\begin{aligned} \max_{Z} \ & \operatorname{tr} Z \\ \text{s.t.} \ & \frac{1}{2}\|\mathcal{P} \circ \widehat{\mathcal{K}}(Z) - d\|_2^2 \le \sigma \\ & Z \succeq 0, \ Z \in \mathcal{S}^{n-1}, \end{aligned}$$

where we define $\widehat{\mathcal{K}}(Z) := \widehat{\mathcal{K}_V}(Z) := \mathcal{K}(VZV^T)$ for every $Z \in \mathcal{S}^{n-1}$. (The map $\widehat{\mathcal{K}_V}$ depends on the choice of V indicated by the suffix, but for simplicity we drop the suffix in the following discussion.) In this reduced form, the flipped problem

$$\begin{aligned} \min_{Z} \ & \frac{1}{2}\|\mathcal{P} \circ \widehat{\mathcal{K}}(Z) - d\|_2^2 \\ \text{s.t.} \ & \operatorname{tr} Z \ge \tau \\ & Z \succeq 0 \end{aligned} \tag{16}$$

is strictly feasible. Therefore, strong duality holds with the corresponding dual problem

$$\begin{aligned} \max_{y} \ & -\frac{1}{2}\|y\|_2^2 + \langle d, y\rangle - \tau\lambda \\ \text{s.t.} \ & \widehat{\mathcal{K}}^* \circ \mathcal{P}^*(y) \preceq \lambda I \\ & \lambda \le 0, \ y \in \mathbb{R}^{\mathcal{E}}. \end{aligned} \tag{17}$$

If $\bar{\lambda}$ is the optimal Lagrange multiplier for the trace constraint, then $\phi'(\tau) = -\bar{\lambda}$.

4.2 Solving the flipped problem

4.2.1 Projected gradient

An immediate approach for solving the flipped problem (16) is to apply the standard *gradient projection method* to the problem (16). Indeed each projection onto the feasible region $\{Z \succeq 0 : \operatorname{tr} Z \ge \tau\}$ requires only a single eigenvalue decomposition. Hence for moderately sized problems each iteration of the method is relatively cheap. In turn, recovering the Lagrange multiplier is an easy operation.

4.2.2 Frank-Wolfe algorithm and dual subgradient descent

For large scale problems, the projected gradient method is too costly because one needs to compute the eigenvalue decomposition of a dense matrix. We here propose an alternative utilizing sparsity, the *Frank-Wolfe algorithm* [11], which has recently found many applications in machine learning (see e.g. [12,15]). The general method considers problems of the form

$$\begin{aligned} \min_{x} \ & f(\mathcal{A}(x)) \\ \text{s.t.} \ & x \in D, \end{aligned} \tag{18}$$

where $f : \mathbb{R}^m \to \mathbb{R}$ is a smooth convex function, $\mathcal{A}$ is a linear map from some Euclidean space (i.e., finite dimensional inner product space) $\mathbb{E}$ to $\mathbb{R}^m$, and $D \subseteq \mathbb{E}$ is a compact set. The Frank-Wolfe algorithm for problem (18) is described in Algorithm 3. Assuming that the trace constraint in (16) is tight, our main problem (16) certainly fits into this framework with $f(\cdot) = \frac{1}{2}\|\cdot - d\|_2^2$, $\mathcal{A} = \mathcal{P} \circ \widehat{\mathcal{K}}$, and $D = \{Z \succeq 0 : \operatorname{tr} Z = \tau\}$.

Algorithm 3 Frank-Wolfe algorithm

Let $k \leftarrow 0$ and $l_0 \leftarrow -\infty$. Pick any point x_0 in D, and set a tolerance $\varepsilon > 0$.
while $f(\mathcal{A}x_k) - l_k > \varepsilon$ **do**
Choose a direction

$$s_k \in \operatorname*{argmin}_{s \in D} \langle \mathcal{A}^* \nabla f(\mathcal{A}x_k), s \rangle; \tag{19}$$

Set the stepsize: $\gamma_k \leftarrow \frac{2}{k+2}$;
Update the iterate: $x_{k+1} \leftarrow x_k + \gamma_k(s_k - x_k)$;
Set the lower bound:

$$l_{k+1} \leftarrow \max\{l_k, f(\mathcal{A}x_k) + \langle s_k - x_k, \mathcal{A}^* \nabla f(\mathcal{A}x_k) \rangle\};$$

Increment the iterate: $k \leftarrow k + 1$;
end while

The Frank-Wolfe algorithm is guaranteed to complete in $\mathcal{O}(\frac{1}{\varepsilon})$ iterations. The computational burden of the method is the minimization problem (19). In our situation (i.e., with $f = \frac{1}{2}\| \cdot - d\|_2^2$ and $\mathcal{A} = \mathcal{P} \circ \widehat{\mathcal{K}}$), one can verify that for all x,

$$\begin{aligned} \mathcal{A}^* \nabla f(\mathcal{A}x) &= \widehat{\mathcal{K}}^* \circ \mathcal{P}^*\big(\mathcal{P} \circ \widehat{\mathcal{K}}(Z_k) - d\big) \\ &= V^T\big(\mathcal{K}^* \circ \mathcal{P}^*\big(\mathcal{P} \circ \widehat{\mathcal{K}}(Z_k) - d\big)\big)V. \end{aligned}$$

It is important to notice is that in light of the standard formula $\mathcal{K}^*(X) = 2(\operatorname{Diag}(Xe) - X)$ and the fact that $\mathcal{P}^* \circ \mathcal{P}$ corresponds to the adjacency matrix of the graph, the inner matrix $W_k := \mathcal{K}^* \circ \mathcal{P}^*\big(\mathcal{P} \circ \widehat{\mathcal{K}}(Z_k) - d\big)$ has the same sparsity pattern, modulo the diagonal, as the adjacency matrix of the graph. As a result, when the graph $\mathcal{G}$ is *sparse*, we claim that the linear optimization problem (19) is easy to solve. Indeed we may find a minimal eigenvalue λ_k and a corresponding eigenvector v_k of the matrix W_k fairly quickly by a Lanczos method, and in particular, by orders of magnitude quicker than the full eigenvalue decomposition. One can now easily check that v_k is orthogonal to e and therefore lies in the range of V, for any V satisfying (15). Letting w_k be any vector satisfying $Vw_k = v_k$, the matrix $\beta w_k w_k^T$ solves the linear optimization problem (19). Consequently, the Frank-Wolfe method is perfectly adapted to our problem instance, since no full eigenvalue decompositions are required, in contrast to the projected gradient method.

Moreover, recall that within the Newton method scheme we are not exactly after the solution to (16), but rather to the Lagrange multiplier λ corresponding to the trace constraint. That is, we are interested in the dual problem

$$\max_{y \in \mathbb{R}^{\mathcal{E}}} \; -\frac{1}{2}\|y\|^2 + \langle d, y \rangle - \tau \cdot \lambda_{\max}\Big(V^T\big(\mathcal{K}^* \circ \mathcal{P}^*(y)\big)V\Big). \tag{20}$$

The value $\lambda_{\max}\Big(V^T\big(\mathcal{K}^*\mathcal{P}^*(y)\big)V\Big)$ at optimality is the Lagrange multiplier that we seek. One can verify that if $\{Z_k\}$ are the iterates generated by the Frank-Wolfe algorithm, then the images $\{y_k\} = \{d - \mathcal{P} \circ \widehat{K}(Z_k)\}$ are the iterates for the classical *subgradient method* on the dual problem (20), with the same step length. The lower-bounds in the Frank-Wolfe algorithm l_k are then the maximal objective values of the dual up to iteration k, and therefore the stopping criterion is nothing more than the current best estimate on the duality gap. Moreover, the eigenvalues λ_k are then precisely the required estimates on the Lagrange multiplier that we seek.

5 Conclusion and work in progress

In this paper, we described two algorithms (robust facial reduction and a search along the Pareto frontier) to solve the noisy sensor network localization problem, which has important applications and is a numerically challenging problem due to its highly nonconvex nature. The two algorithms are intended for sensor networks of different densities: the Pareto frontier algorithm outlined in Section 4 is designed for sparse graphs whereas the robust facial reduction outlined in Algorithm 1 in Section 3 tends to work better for denser graphs. As we illustrated, both approaches can work in anchorless sensor networks, since only the inter-sensor distance measurements are used. Moreover, availability of anchor information will only improve the performance of the algorithms.

We outlined some intuitive ideas behind the robustness of the two approaches, and the formal statements of their robustness guarantees will be presented in our upcoming journal paper. We demonstrated some preliminary numerical results for the robust facial reduction technique. Though not studied in this work, it is possible to develop a distributed implementation of the robust facial reduction technique in order to solve even larger scale SNL problem. In principle, the robust facial reduction technique should work with networks that are sparser than those we have tested, but we have yet to verify this in practice.

The Pareto frontier estimation technique is promising for handling large scale SNL problems, since first-order methods become immediately applicable and sparsity of the underlying graph can be exploited when searching for a maximum eigenvalue-eigenvector pair via a Lanczos procedure. In the upcoming journal paper, we will illustrate the numerical promise and implementation details of this approach.

References

[1] S. Al-Homidan and H. Wolkowicz. Approximate and exact completion problems for Euclidean distance matrices using semidefinite programming. *Linear Algebra Appl.*, 406:109–141, 2005. 6

[2] A. Aravkin, J. Burke, and M. Friedlander. Variational properties of value functions. *SIAM J. Optim.*, 23(3):1689–1717, 2013. 2, 14, 15

[3] J. Aspnes, T. Eren, D. Goldenberg, A. Morse, W. Whiteley, Y. Yang, B. Anderson, and P. Belhumeur. Semidefinite programming approaches for sensor network localization with noisy distance measurements. *IEEE Transactions on Automation Science and Engineering*, 3(4):360–371, October 2006. 2

[4] P. Biswas, T.-C. Lian, T.-C. Wang, and Y. Ye. Semidefinite programming based algorithms for sensor network localization. *ACM Trans. Sen. Netw.*, 2(2):188–220, 2006. 1, 4, 10, 11

[5] P. Biswas, T.-C. Liang, K.-C. Toh, , Y. Ye, and T.-C. Wang. Semidefinite programming approaches for sensor network localization with noisy distance measurements. *IEEE Transactions on Automation Science and Engineering*, 3:360–371, 2006. 1, 4, 11

[6] 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(3):1251–1277, 2008. 1, 10

[7] P. Biswas and Y. Ye. Semidefinite programming for ad hoc wireless sensor network localization. Technical report, Dept. of Management Science and Engineering, Stanford University, 2003. 1

[8] P. Biswas and Y. Ye. Semidefinite programming for ad hoc wireless sensor network localization. In *IPSN '04: Proceedings of the 3rd international symposium on Information processing in sensor networks*, pages 46–54, New York, NY, USA, 2004. ACM. 1, 4

[9] N. Boumal, B. Mishra, P.-A. Absil, and R. Sepulchre. Manopt, a matlab toolbox for optimization on manifolds. *Journal of Machine Learning Research*, 15:1455–1459, 2014. 10

[10] D. Drusvyatskiy, G. Pataki, and H. Wolkowicz. Coordinate shadows of semi-definite and euclidean distance matrices. *Preprint, arXiv:1405.2037*, 2014. 6

[11] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Res. Logist. Quart.*, 3:95–110, 1956. 15

[12] Z. Harchaoui, A. Juditsky, and A. Nemirovski. Conditional gradient algorithms for norm-regularized smooth convex optimization. *Mathematical Programming*, pages 1–38, 2014. 15

[13] T. Hayden, J. Lee, J. Wells, and P. Tarazaga. Block matrices and multispherical structure of distance matrices. *Linear Algebra Appl.*, 247:203–216, 1996. 6

[14] T. Hayden, J. Wells, W.-M. Liu, and P. Tarazaga. The cone of distance matrices. *Linear Algebra Appl.*, 144:153–169, 1991. 6

[15] M. Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *ICML*, 2013. 15

[16] N. Krislock and H. Wolkowicz. Explicit sensor network localization using semidefinite representations and facial reductions. *SIAM J. Optim.*, 20(5):2679–2708, 2010. 1, 2, 3, 4, 5, 6

[17] H. Kurata and P. Tarazaga. Multispherical Euclidean distance matrices. *Linear Algebra Appl.*, 433(3):534–546, 2010. 6

[18] H. Kurata and P. Tarazaga. Majorization for the eigenvalues of Euclidean distance matrices. *Linear Algebra Appl.*, 436(5):1473–1481, 2012. 6

[19] B. Mishra, G. Meyer, and R. Sepulchre. Low-rank optimization for distance matrix completion. In *CDC-ECE'11*, pages 4455–4460, 2011. 10

[20] T. Pong and P. Tseng. Robust edge-based semidefinite programming relaxation of sensor network localization. Technical report, U. of Washington, 2009. 1

[21] J. Saxe. Embeddability of weighted graphs in k-space is strongly NP-hard. In *Seventeenth Annual Allerton Conference on Communication, Control, and Computing, Proceedings of the Conference held in Monticello, Ill., October 10–12, 1979*, pages xiv+1036, Urbana, 1979. University of Illinois Department of Electrical Engineering. Proceedings of the International School of Physics "Enrico Fermi", LXX*. 3

[22] A. So and Y. Ye. Theory of semidefinite programming for sensor network localization. *Math. Program.*, 109(2):367–384, 2007. 1, 3, 6

[23] A. M.-C. So and Y. Ye. Theory of semidefinite programming for sensor network localization. *Math. Program. Ser. B*, pages 367–384, 2007. 1

[24] P. Tarazaga. Faces of the cone of Euclidean distance matrices: characterizations, structure and induced geometry. *Linear Algebra Appl.*, 408:1–13, 2005. 6

[25] P. Tarazaga and J. Gallardo. Euclidean distance matrices: new characterization and boundary properties. *Linear Multilinear Algebra*, 57(7):651–658, 2009. 6

[26] P. Tarazaga, T. Hayden, and J. Wells. Circum-Euclidean distance matrices and faces. *Linear Algebra Appl.*, 232:77–96, 1996. 6

[27] P. Tarazaga, B. Sterba-Boatwright, and K. Wijewardena. Euclidean distance matrices: special subsets, systems of coordinates and multibalanced matrices. *Comput. Appl. Math.*, 26(3):415–438, 2007. 6

[28] E. van den Berg and M. Friedlander. Probing the Pareto frontier for basis pursuit solutions. *SIAM J. Sci. Comput.*, 31(2):890–912, 2008/09. 2, 14

[29] E. van den Berg and M. Friedlander. Sparse optimization with least-squares constraints. *SIAM J. Optim.*, 21(4):1201–1229, 2011. 2, 14

[30] E. van den Berg and M. P. Friedlander. SPGL1: A solver for large-scale sparse reconstruction, June 2007. http://www.cs.ubc.ca/labs/scl/spgl1. 2, 14

[31] Z. Wang, S. Zheng, Y. Ye, and S. Boyd. Further relaxations of the semidefinite programming approach to sensor network localization. *SIAM J. Optim.*, pages 655–673, 2008. 1

[32] K. Weinberger, F. Sha, and L. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 106, New York, NY, USA, 2004. ACM. 2

[33] Y. Yemini. Some theoretical aspects of position-location problems. In *20th Annual Symposium on Foundations of Computer Science (San Juan, Puerto Rico, 1979)*, pages 1–8. IEEE, New York, 1979. 3

Index