An Iterative Eigensolver for Rank-Constrained Semidefinite Programming

Rajat Sanyal Aditya V. Singh Kunal N. Chaudhury
Department of Electrical Engineering, Indian Institute of Science, Bangalore, India
Email: sanyalrajat91@gmail.com {adityavs, kunal}@iisc.ac.in

***Abstract*—Rank-constrained semidefinite programming (SDP) arises naturally in various applications such as max-cut, angular (phase) synchronization, and rigid registration. Based on the alternating direction method of multipliers, we develop an iterative solver for this nonconvex form of SDP, where the dominant cost per iteration is the partial eigendecomposition of a symmetric matrix. We prove that if the iterates converge, then they do so to a KKT point of the SDP. In the context of rigid registration, we perform several numerical experiments to study the convergence behavior of the solver and its registration accuracy. As an application, we use the solver for wireless sensor network localization from range measurements. The resulting algorithm is shown to be competitive with existing optimization methods for sensor localization in terms of speed and accuracy.**

***Index Terms*—semidefinite programming, ADMM, eigensolver, convergence, registration, sensor network localization.**

I. Introduction

We consider a class of rank-constrained semidefinite programs that arise in applications such as finding the largest cut in a graph [1], determining angles from their differences [2], and the registration of point clouds using rigid transforms [3]. We will focus on the registration problem, where we have N points in $\mathbb{R}^d$, which are divided into M overlapping point clouds. The local coordinates (and the label) of points in each point cloud are known. The task is to compute the global coordinates of all the N points [3]. We will refer to this as *rigid registration*, which has found applications in sensor network localization [4]. The maximum likelihood estimator for this problem involves optimization over rigid transforms (translations, rotations and reflections) [3]. In particular, the maximum likelihood estimate for this problem is given by the solution of

$$\min_{\mathbf{O}_1,\ldots,\mathbf{O}_M\in\mathbb{O}(d)} \sum_{i,j=1}^{M} \operatorname{Tr}\left([\mathbf{C}]_{ij}\mathbf{O}_j^\top \mathbf{O}_i\right). \tag{1}$$

where $\mathbb{O}(d)$ is the set of $d\times d$ orthogonal matrices, and $[\mathbf{C}]_{ij}\in\mathbb{R}^{d\times d}$ denotes the (i,j)-th block of a certain $\mathbf{C}\in\mathbb{R}^{Md\times Md}$. We refer the reader to [3] for details. Interestingly, (1) can be seen as a non-commutative analogue of the Boolean optimization

$$\min_{x_i,\ldots,x_M\in\{-1,1\}} \sum_{i,j=1}^{M} c_{ij}x_jx_i,$$

which comes up in max-cut [1]. From the point of view of continuous optimization, the main challenge with (1) is that $\mathbb{O}(d)$ is not a connected manifold (apart from being nonconvex). In the context of local optimization, this means that we cannot hope to compute the maximum likelihood estimate unless we initialize the iterations on the correct component of the domain. Since the domain in (1) has 2^M components, getting the right initialization is difficult. It was observed in [3] that if we work with the Gram matrix of $\mathbf{O}_1,\ldots,\mathbf{O}_M$, then we can express (1) as a standard semidefinite program (SDP), albeit with an additional rank-constraint. The authors proposed to relax the rank constraint to obtain a standard SDP, whose solution can be computed to arbitrary precision using an interior point solver. Later, an efficient and scalable SDP solver was proposed in [4]. Though these methods can find the global minimum of the relaxed problem, there is no guarantee that the rank of the solution is exactly d (we would have solved (1) in this case); if the rank is greater than d, then the solution becomes infeasible for (1). This necessitates "rounding" of the solution of the convex relation to obtain a feasible solution for (1), which will generally be suboptimal. Our idea is to build an efficient solver that can directly tackle (1). In this regard, our contributions are as follows:

- Based on the alternating direction method of multipliers (ADMM) [5], we develop an iterative solver for (1) that involves simple updates. In particular, one of the updates is trivial, while the other is a partial eigendecomposition of a symmetric matrix.
- We show that any fixed point of our solver is a KKT point of (1). This result is novel because, as will be made explicit, a crucial assumption behind the existing analyses on nonconvex ADMM [6] [7], [8] does not hold in our case.
- We present numerical results for rigid registration and sensor network localization, which demonstrate the effectiveness of the solver in terms of accuracy and timing.

We note that ADMM based algorithms have become popular for structured convex programming [5]. Lately, the ADMM framework has been successfully applied to various nonconvex problems, even though rigorous convergence guarantees are not available. In fact, while the analysis for convex ADMM is well established, nonconvex ADMM is still a developing area of research. Some results have been reported in [6]–[8], but the analysis in these works relies on regularity assumptions on the objective that are not met for our ADMM formulation. More precisely, both updates of our ADMM solver involves constrained optimization, while at least one update in [6]–[8] is assumed to be a smooth unconstrained optimization.

978-1-5386-9286-8/19/$31.00

The rest of the paper is as follows. In Section II, we formulate the optimization problem, based on which we develop the solver in Section III. The fixed point analysis is undertaken in Section IV. Numerical results are reported in Section V, and we conclude with a summary of the results in Section VI.

II. Problem Formulation

It was observed in [3] that we can express (1) as a rank-constrained semidefinite program (SDP). More specifically, consider the Gram matrix $\mathbf{G}$ of size $m \times m$, whose (i,j)-th block is $[\mathbf{G}]_{ij} = \mathbf{O}_i^\top \mathbf{O}_j$, where $i,j \in [\![1, M]\!]$. Here and henceforth, $m = Md$, and we use $[\![p, q]\!]$ to denote the integers $\{p, \dots, q\}$. In terms of $\mathbf{G}$, we can reformulate (1) as

$$\begin{array}{ll} \min\limits_{\mathbf{G} \in \mathbb{S}_+^m} & \mathrm{Tr}(\mathbf{C}\mathbf{G}) \\ \text{subject to} & [\mathbf{G}]_{ii} = \mathbf{I}_d, \quad i \in [\![1, M]\!], \ \mathrm{rank}(\mathbf{G}) = d, \end{array} \tag{2}$$

where $\mathbb{S}_+^m$ is the set of symmetric positive semidefinite matrices of size $m \times m$. This is a standard SDP, except for the additional rank constraint, which in fact makes the problem nonconvex. Following this observation, a convex relaxation (`GRET-SDP`) was proposed in [3] simply by dropping the rank constraint. However, the relaxation is not guaranteed to return a rank-d solution, and one is required to "round" the solution if the rank is greater than d. This can produce suboptimal solutions, i.e., the objective value of the rounded solution can be much larger than the optimum of (2).

As against this, we propose to directly tackle the original problem (1). In particular, we will demonstrate that an iterative solver can be developed for (1), where the subproblems admit closed-form solutions that can be computed efficiently. In particular, consider the variable

$$\mathbf{W} = \frac{1}{\sqrt{M}} [\mathbf{O}_1 \dots \mathbf{O}_M]^\top \in \mathbb{R}^{m \times d}.$$

Notice that we can write (1) as

$$\begin{array}{ll} \min\limits_{\mathbf{W} \in \mathbb{R}^{m \times d}} & \mathrm{Tr}(\mathbf{C}\mathbf{W}\mathbf{W}^\top) \\ \text{subject to} & [\mathbf{W}\mathbf{W}^\top]_{ii} = M^{-1}\mathbf{I}_d, \quad i \in [\![1, M]\!]. \end{array} \tag{3}$$

As mentioned above, the authors in [3] choose to work with the Gram matrix $\mathbf{W}\mathbf{W}^\top$. We will however continue to work with $\mathbf{W}$. Moreover, for reasons that will be apparent in Section III, we propose to add the redundant constraint $\mathbf{W}^\top\mathbf{W} = \mathbf{I}_d$. That is, we replace (3) by the equivalent problem

$$\begin{array}{ll} \min\limits_{\mathbf{W} \in \mathbb{R}^{m \times d}} & \mathrm{Tr}(\mathbf{C}\mathbf{W}\mathbf{W}^\top) \\ \text{subject to} & \mathbf{W}^\top\mathbf{W} = \mathbf{I}_d, \\ & [\mathbf{W}\mathbf{W}^\top]_{ii} = M^{-1}\mathbf{I}_d, \quad i \in [\![1, M]\!]. \end{array} \tag{4}$$

By "equivalent", we mean that any optimal solution of (3) is also optimum for (4), and vice versa. Importantly, notice that unlike (2), there are no (explicit) rank constraints in (4).

III. Proposed Solver

We propose to solve (4) using variable splitting and the alternating direction method of multipliers [5]. More specifically, by introducing the variable $\mathbf{X} = \mathbf{W}\mathbf{W}^\top$, we first transform (4) into the following problem:

$$\begin{array}{ll} \min\limits_{\mathbf{W} \in \mathbb{R}^{m \times d}, \ \mathbf{X} \in \mathbb{S}^m} & \mathrm{Tr}(\mathbf{C}\mathbf{W}\mathbf{W}^\top) \\ \text{subject to} & [\mathbf{X}]_{ii} = M^{-1}\mathbf{I}_d, \quad i \in [\![1, M]\!], \\ & \mathbf{W}^\top\mathbf{W} = \mathbf{I}_d, \quad \mathbf{X} = \mathbf{W}\mathbf{W}^\top, \end{array} \tag{5}$$

where $\mathbb{S}^m$ is the set of symmetric matrices of size m. In terms of the sets

$$\Theta = \left\{ \mathbf{W} \in \mathbb{R}^{m \times d} : \mathbf{W}^\top\mathbf{W} = \mathbf{I}_d \right\},$$

and

$$\Omega = \left\{ \mathbf{X} \in \mathbb{S}^m : [\mathbf{X}]_{ii} = M^{-1}\mathbf{I}_d, i \in [\![1, M]\!] \right\},$$

we can compactly write (5) as follows:

$$\begin{array}{ll} \min\limits_{\mathbf{W} \in \Theta, \ \mathbf{X} \in \Omega} & \mathrm{Tr}(\mathbf{C}\mathbf{W}\mathbf{W}^\top) \\ \text{subject to} & \mathbf{X} = \mathbf{W}\mathbf{W}^\top. \end{array} \tag{6}$$

This is a constrained optimization problem with variables $\mathbf{X}$ and $\mathbf{W}$. The augmented Lagrangian for (6) is given by

$$\begin{aligned} \mathcal{L}_\rho(\mathbf{W}, \mathbf{X}, \mathbf{\Lambda}) = \ & \langle \mathbf{C}, \mathbf{W}\mathbf{W}^\top \rangle + \langle \mathbf{\Lambda}, \mathbf{X} - \mathbf{W}\mathbf{W}^\top \rangle \\ & + \frac{\rho}{2} \|\mathbf{X} - \mathbf{W}\mathbf{W}^\top\|^2, \end{aligned} \tag{7}$$

where $\mathbf{W}$ and $\mathbf{X}$ are the primal variables, and $\mathbf{\Lambda} \in \mathbb{S}^m$ is the dual variable associated with the constraint $\mathbf{X} = \mathbf{W}\mathbf{W}^\top$; $\rho > 0$ is a penalty parameter [5]. Notice that we have used the inner-product $\langle \mathbf{X}, \mathbf{Y} \rangle = \mathrm{Tr}(\mathbf{X}\mathbf{Y})$ and the Frobenius norm $\|\mathbf{X}\| = \langle \mathbf{X}, \mathbf{X} \rangle^{1/2}$, both defined on $\mathbb{S}^m$.

Starting with some initialization $\mathbf{X}, \mathbf{\Lambda} \in \mathbb{S}^m$ and $\rho_0 > 0$, the ADMM iterates for $k = 0, 1, \dots$ are given by

$$\mathbf{W}^{k+1} = \operatorname*{arg\,min}_{\mathbf{W} \in \Theta} \ \mathcal{L}_{\rho_k}(\mathbf{W}, \mathbf{X}^k, \mathbf{\Lambda}^k), \tag{8}$$

$$\mathbf{X}^{k+1} = \operatorname*{arg\,min}_{\mathbf{X} \in \Omega} \ \mathcal{L}_{\rho_k}(\mathbf{W}^{k+1}, \mathbf{X}, \mathbf{\Lambda}^k), \tag{9}$$

$$\mathbf{\Lambda}^{k+1} = \mathbf{\Lambda}^k + \rho_k \big(\mathbf{X}^{k+1} - \mathbf{W}^{k+1}{\mathbf{W}^{k+1}}^\top \big),$$

$$\rho_{k+1} = \min(\gamma \rho_k, \rho_\infty),$$

where $\gamma > 1$. Notice that ρ_k is allowed to increase at each iteration up till ρ_∞, and then it is held fixed.

It is not difficult to verify that by combining the linear and quadratic terms, we can write (9) as

$$\mathbf{X}^{k+1} = \operatorname*{arg\,min}_{\mathbf{X} \in \Omega} \ \|\mathbf{X} - \mathbf{A}^k\|^2 = \mathcal{P}_\Omega(\mathbf{A}^k), \tag{10}$$

where $\mathbf{A}^k = \mathbf{W}^{k+1}{\mathbf{W}^{k+1}}^\top - {\rho_k}^{-1}\mathbf{\Lambda}^k$, and $\mathcal{P}_\Omega$ is the projection operator. From the definition of Ω, it follows that $\mathbf{X}^{k+1}$ is obtained simply by replacing the diagonal blocks of $\mathbf{A}^k$ with $M^{-1}\mathbf{I}_d$, keeping the other blocks unchanged.

We next compute the update in (8). After some manipulations, we can write this as

$$\mathbf{W}^{k+1} = \operatorname*{arg\,min}_{\mathbf{W} \in \Theta} \ \langle \mathbf{B}^k, \mathbf{W}\mathbf{W}^\top \rangle, \tag{11}$$

where $\mathbf{B}^k = \mathbf{C} - \mathbf{\Lambda}^k - \rho_k \mathbf{X}^k$. It is now clear that (11) is an eigenvalue problem. Indeed, if $\mathbf{w}_1, ..., \mathbf{w}_d$ are the columns of $\mathbf{W}$, then we can write (11) as

$$\min_{\mathbf{w}_1,\ldots,\mathbf{w}_d} \sum_{i=1}^{d} \mathbf{w}_i^\top \mathbf{B}^k \mathbf{w}_i. \tag{12}$$

Moreover, since $\mathbf{W} \in \Theta$, it follows that $\mathbf{w}_1, ..., \mathbf{w}_d$ form an orthonormal system. Let $\lambda_1, \ldots, \lambda_m$ be the eigenvalues of $\mathbf{B}^k$ sorted in non-decreasing order, with corresponding eigenvectors $\mathbf{q}_1, \ldots, \mathbf{q}_m$. Then the solution of (12) is given by $\mathbf{w}_i^\star = \mathbf{q}_i, i \in [\![1, d]\!]$. In other words,

$$\mathbf{W}^{k+1} = \big[\mathbf{q}_1 \cdots \mathbf{q}_d\big]. \tag{13}$$

The ADMM updates are summarized in Algorithm 1. The dominating cost per iteration is the (partial) eigendecomposition of $\mathbf{B}^k$, and this can be performed efficiently using off-the-shelf eigensolvers. Since the problem is nonconvex, the initialization of $\mathbf{X}$ plays an important role. In this regard, we note that the eigendecomposition of $\mathbf{C}$ can be used to obtain a non-trivial initialization of $\mathbf{X}$ [3]. As for $\mathbf{\Lambda}$, we simply initialize it to zero for all the experiments.

Algorithm 1: ADMM Solver

Input: $\mathbf{C}, \gamma > 1, \rho_\infty > 0$.
Initialize: $\mathbf{X}, \mathbf{\Lambda}$, and $\rho > 0$.
while *some stopping criteria is not met*
  $\mathbf{B} = \mathbf{C} - \mathbf{\Lambda} - \rho \mathbf{X}$.
  $\{\mathbf{q}_1, \ldots, \mathbf{q}_d\}$: bottom d eigenvectors of $\mathbf{B}$.
  $\mathbf{W} = [\mathbf{q}_1 \cdots \mathbf{q}_d]$.
  $\mathbf{X} \leftarrow \Pi_\Omega(\mathbf{W}\mathbf{W}^\top - \rho^{-1}\mathbf{\Lambda})$.
  $\mathbf{\Lambda} \leftarrow \mathbf{\Lambda} + \rho(\mathbf{X} - \mathbf{W}\mathbf{W}^\top)$.
  $\rho \leftarrow \min(\gamma\rho, \rho_\infty)$.
end

IV. Fixed Point Analysis

Convergence analysis of ADMM for convex problems is a well-researched topic [5]. However, a theoretical understanding of why ADMM solvers applied to nonconvex programs succeed so often in practice remains elusive. Lately, there have been a handful convergence results for nonconvex ADMM [6] [7], [8]. Unfortunately, they rely on assumptions that do not hold for our problem. More precisely, note that we can rewrite (6) as

$$\begin{aligned} \min_{\mathbf{W}\in\mathbb{R}^{m\times d},\ \mathbf{X}\in\mathbb{S}^m} \quad & \mathrm{Tr}(\mathbf{C}\mathbf{W}\mathbf{W}^\top) + \iota_\Theta(\mathbf{W}) + \iota_\Omega(\mathbf{X}) \\ \text{subject to} \quad & \mathbf{X} = \mathbf{W}\mathbf{W}^\top, \end{aligned} \tag{14}$$

where ι_Γ is the indicator function associated with a feasible set Γ, namely, $\iota_\Gamma(\mathbf{Y}) = 0$ if $\mathbf{Y} \in \Gamma$, and $\iota_\Gamma(\mathbf{Y}) = \infty$ otherwise [5]. Note that, because of the indicator functions, the objective function in (14) is non-smooth in *both* $\mathbf{W}$ and $\mathbf{X}$. This violates a crucial regularity assumption common in existing analyses of nonconvex ADMM, namely, that the objective must be smooth in *at least* one variable. As a result, none of the existing results on convergence are applicable to the proposed ADMM solver. Nevertheless, we will show in Section V that Algorithm 1 performs well empirically and, in particular, it is found to converge with the spectral initialization. On the theoretical front, we have succeeded in establishing that *if* the iterates of Algorithm 1 converge, then they do so to a KKT (stationary) point (e.g., see [9, Chapter 3]).

Before formally stating our result, we write (1) as a nonlinear program:

$$\begin{aligned} \min_{\mathbf{O}_1,\cdots,\mathbf{O}_M\in\mathbb{R}^{d\times d}} \quad & \sum_{i,j=1}^{M} \mathrm{Tr}\left([\mathbf{C}]_{ij}\mathbf{O}_j^\top\mathbf{O}_i\right) \\ \text{subject to} \quad & \mathbf{I}_d - \mathbf{O}_i^\top\mathbf{O}_i = 0, \qquad i \in [\![1, M]\!]. \end{aligned} \tag{15}$$

This allows us to write the Lagrangian of (15) and use KKT theory. In particular, the Lagrangian is given by

$$\mathcal{L} = \sum_{i,j=1}^{M} \mathrm{Tr}\left([\mathbf{C}]_{ij}\mathbf{O}_j^\top\mathbf{O}_i\right) + \sum_{i=1}^{M} \mathrm{Tr}(\mathbf{\Lambda}_k\left(\mathbf{I}_d - \mathbf{O}_i^\top\mathbf{O}_i\right)), \tag{16}$$

where the symmetric matrices $\mathbf{\Lambda}_i \in \mathbb{R}^{d\times d}, i \in [\![1, M]\!]$, are the Lagrange multipliers for the equality constraints (these should not be confused with the multiplier in (7)). We have the following characterization of the KKT point of (16) (see Appendix VII-A for the proof). Recall that $\mathbf{G}$ is the Gram matrix of the $\mathbf{O}_i$'s, whose (i, j)-th block is $[\mathbf{G}]_{ij} = \mathbf{O}_i^\top\mathbf{O}_j$.

Lemma 1. *The variables $\mathbf{O}_1^*, \ldots, \mathbf{O}_M^* \in \mathbb{R}^{d\times d}$ are a KKT point of* (15) *if and only if, for $i \in [\![1, M]\!]$,*

(a) $[\mathbf{G}^*]_{ii} = \mathbf{I}_d$, *and*
(b) $[\mathbf{C}\mathbf{G}^*]_{ii} = [\mathbf{G}^*\mathbf{C}]_{ii}$.

In this case, we will say that $\mathbf{G}^*$ is a KKT point of (15).

We now make precise the notion of convergence that is used in our analysis. We say that Algorithm 1 has *converged* if it "stops making any progress", i.e., the variables stop getting updated. Stated differently, if we view the progress from one iteration to the next as a map (from some space into itself), then this is equivalent to the iterates converging to a *fixed point* of this map. Indeed, note that if

$$\mathbf{W}^{k+1}\mathbf{W}^{k+1^\top} = \mathbf{X}^k \tag{17}$$

for some $k = k_0$, then we must have for $k \geq k_0$:

$$\mathbf{X}^{k+1} = \mathbf{W}^{k+1}\mathbf{W}^{k+1^\top} \ \text{ and } \ \mathbf{\Lambda}^{k+1} = \mathbf{\Lambda}^k. \tag{18}$$

Conversely, if (18) holds at iteration $k = k_0$, (17) must hold for $k \geq k_0$. In summary, the convergence of Algorithm 1 is equivalent to the condition that (17) holds for some $k = k_0$. We are now in a position to state our main result (the proof is provided in Appendix VII-B).

Theorem 2. *Suppose $\mathbf{\Lambda}^0 = \mathbf{0}$ and $\mathbf{X}^k = \mathbf{W}^{k+1}\mathbf{W}^{k+1^\top}$ at iteration $k = k_0$. Then $\mathbf{G}^* = M\mathbf{X}^{k_0}$ is a KKT point of* (15).

The practical significance of this result is that the feasibility gap $\|\mathbf{X}^k - \mathbf{W}^{k+1}\mathbf{W}^{k+1^\top}\|$ can be used to monitor the evolution of the iterates to a fixed point. For example, we can stop the iterations when this gap is less than a specified tolerance.

V. Numerical Experiments

We now report some numerical experiments on rigid registration to analyze performance of the proposed solver. We also compare its performance with `GRET-SDP` [3], which solves a convex relaxation of (2). Moreover, as a concrete application, we use our algorithm for sensor network localization (SNL) using a registration-based framework [10]. In this context, we also compare our method with `SNLSDP` [11] and `ESDP` [12], which are still considered state-of-the-art SNL solvers as far as localization accuracy is concerned. `SNLSDP` solves the semidefinite relaxation of the SNL using a interior point solver [11]. Due to the poor scalability of the interior method, a further edge-based relaxation is proposed in [12] known as `ESDP`. The simulations were executed on a 3.4 GHz, quad-core machine with 32 GB memory, using the MATLAB implementation of Algorithm 1. For the experiments, we increase the value of ρ for the first 100 iterations, and fix it for subsequent iterations.

A. Performance analysis for rigid registration

Experiment 1. For two point clouds, i.e., when $M = 2$, the registration problem has a closed-form solution [13]. In this case, global optimum of (1) can be computed exactly. As a result, we can check optimality of the solution computed by the proposed solver for two point clouds. Specifically, we consider a point cloud with $N = 500$ points. We apply a random rigid transformation on each point cloud and corrupt the local coordinates. If the global coordinates are $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^d$, then the generated local coordinates are as follows:

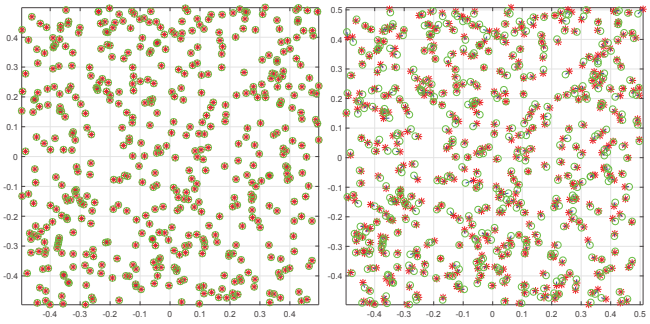
$$\mathbf{x}_{k,i} = \mathbf{O}_i \mathbf{x}_k + \mathbf{t}_i + \boldsymbol{\epsilon}_{k,i}, \quad \boldsymbol{\epsilon}_{k,i} \sim \mathcal{N}(\mathbf{0}, \eta \mathbf{I}_d),$$

where $\mathbf{x}_{k,i}$ is the coordinate of the k-th point in the i-th point cloud ($i = 1, 2$). A typical simulation result is shown in Fig 1. In this case, $\rho_0 = $ 1e-4 and $\gamma = 1.1$, and we ran the algorithm for 500 iterations. We used a random initialization for our solver. Interestingly, the iterates converged in just two iterations, for the noiseless and noisy scenarios. The reconstructed point cloud (after alignment) and the original point cloud are shown in Fig 1. To measure the reconstruction accuracy, we use the average normalized error (ANE) [14]:

$$\text{ANE} = \left\{ \frac{\sum_{i=1}^{N} \|\widehat{\mathbf{x}}_i - \bar{\mathbf{x}}_i\|^2}{\sum_{i=1}^{N} \|\bar{\mathbf{x}}_i - \bar{\mathbf{x}}_c\|^2} \right\}^{1/2},$$

where $(\widehat{\mathbf{x}}_i)$ are the coordinates of the reconstructed point cloud (after alignment), $(\bar{\mathbf{x}}_i)$ are the original coordinates (ground truth), and $\bar{\mathbf{x}}_c = (\bar{\mathbf{x}}_1 + \cdots + \bar{\mathbf{x}}_N)/N$ is the centroid. We notice that the proposed method can solve the registration problem exactly for any random initialization when $\eta = 0$.

Experiment 2. We now study how the proposed algorithm behaves for different values of ρ_0. We consider the setup in Experiment 1, but we use $M = 10$ point clouds. The evolution of the objective function with iterations is shown in Fig 2, for noise level $\eta = 0.01$. Notice that the objective converges within few iterations, though it converges to different values depending on ρ_0. In particular, the ANEs are identical ($= 0.7$) for $\rho_0 =$ 1e-4, 1e-3, and 1e-2; however, ANE $= 17.3$ when



(a) $\eta = 0$, ANE $=$ 1.2e-12. (b) $\eta = 0.01$, ANE $=$ 5.4e-1.

Fig. 1. Registration of two point clouds with $N = 500$ point each, both with and without noise in the local coordinates. The estimated and the original coordinates are marked using $\star$ and $\circ$. We have used random initializations.

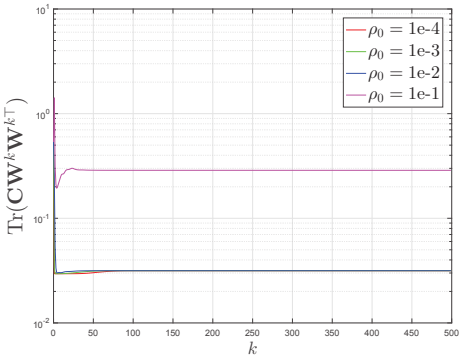


Fig. 2. Evolution of the objective function with iterations for different ρ_0.

$\rho_0 = 0.1$ (see the plots in Fig 2). A possible explanation is that the iterates converge to a poor local minimum in the latter case. Based on exhaustive simulations, it appears that the ANE is small when ρ_0 is in the range [1e-4, 1e-2]. Unfortunately, unlike when $M = 2$, since we cannot ascertain the global minimum of (1) in this case, we cannot assert that the iterates converge to the optimal solution when $\rho_0 \in$ [1e-4, 1e-2].

Experiment 3. We now perform an experiment using the setup in Experiment 2, but at zero noise level. We measure the feasibility gap $\|\mathbf{X}^k - \mathbf{W}^k \mathbf{W}^{k\top}\|$ at each iteration. These are shown in Fig 3. Notice that the gap seems to vanish (up to machine precision) after a finite number of iterations. For completeness, the ANE is 9.3e-11 (exact reconstruction).

Experiment 4. We next compare with `GRET-SDP` [3] in terms of timing and accuracy. We set $N = 500, M = 100, d = 2$ and $\eta =$ 1e-2. For both methods, we initialize with the spectral solution `GRET-SPEC` [3]. Evolution of the objective is shown in Fig 4. Note that both methods converge to the same objective. However, the proposed algorithm converges much faster than the convex `GRET-SDP` solver. Moreover, as far as the per-iteration cost is concerned, our method requires just the bottom d eigenvectors (d is 2 or 3 for most practical problems), whereas `GRET-SDP` requires the full eigendecomposition. A comparison with `GRET-SPEC` in terms of accuracy is provided in Fig 5. Notice that the proposed method performs better than `GRET-SPEC`. Moreover, as with `GRET-SPEC`, the ANE for

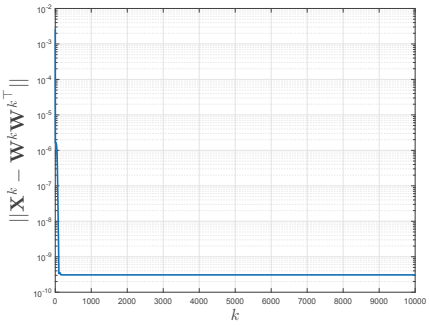


Fig. 3. Evolution of the feasibility gap with iterations.

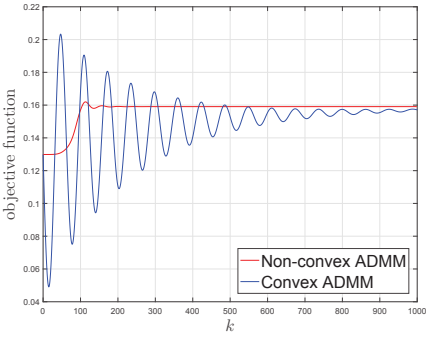


Fig. 4. Convergence results for $N = 500, M = 100, d = 2$ and $\eta = 0.01$.

our method show a linear trend with the noise level.

B. Application to sensor network localization

We now demonstrate the effectiveness of our algorithm for range-based wireless SNL. Recall that the problem in SNL is to determine the location of a network of sensors from inter-sensor distances and locations of a few selected sensors (called anchors) [11], [12]. More specifically, the distance between two sensors is assumed to be known if they are within the radio range (denoted by r) of each other. It was shown in [10] that the localization problem can be solved efficiently by mapping it into a registration problem. More specifically, the idea was to divide the wireless network into smaller overlapping cliques (wherein all the pairwise distances are known). Each clique is then efficiently localized (in parallel) using classical multidimensional scaling. Finally, the cliques are registered in a global coordinate system using rigid registration. We propose to use Algorithm (1) in place of `GRET-SDP` which was originally used in [10].

As for the network topology, we consider random geometric graphs (RGGs) and structured datasets. To simulate a RGG, we randomly sample points from the unit square $[-0.5, 0.5]^2$, and consider them as the sensors. We connect two sensors by an edge only if their distance is less than r. For each edge, the distance measurement is modeled as follows [11], [12]:

$$d_{ij} = |1 + \epsilon_{ij}| \cdot \|\bar{\mathbf{x}}_i - \bar{\mathbf{x}}_j\|, \quad \epsilon_{ij} \sim \mathcal{N}(0, \eta),$$

when i and j are sensors, and as

$$d_{ik} = |1 + \epsilon_{ik}| \cdot \|\bar{\mathbf{x}}_i - \mathbf{a}_k\|, \quad \epsilon_{ik} \sim \mathcal{N}(0, \eta),$$

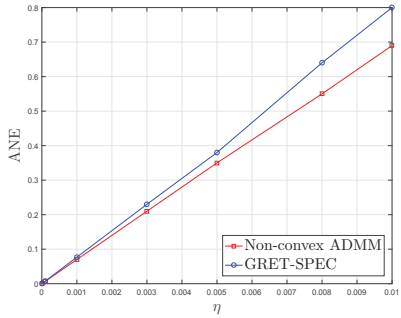
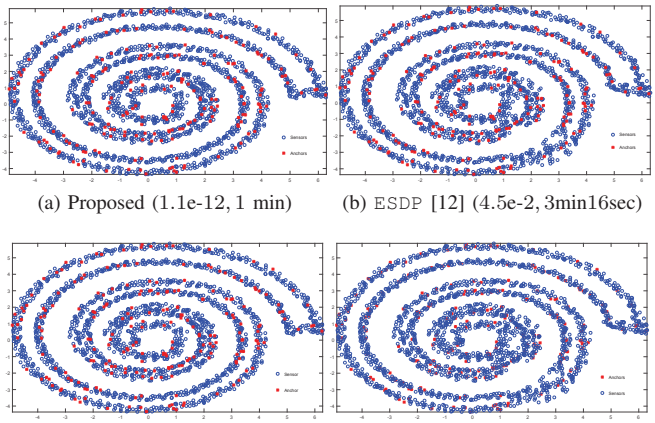


Fig. 5. Comparisons of ANEs for various η ($N = 500, M = 100, d = 2$).



(a) Proposed (1.1e-12, 1 min) (b) `ESDP` [12] (4.5e-2, 3min16sec)

(c) Proposed (6.6e-4, 59 sec) (d) `ESDP` [12] (4.6e-2, 3min17sec)

Fig. 6. Localization results for the spiral dataset [14]. The first row corresponds to clean measurements ($\eta = 0$), while the bottom row is for $\eta = 0.01$. In either case, $r = 0.8$. The anchors are shown in $\star$, while the localized sensors are shown in $\circ$. The (ANE, timing) are also reported.

if i is a sensor and k is an anchor.

Experiment 5. A detailed comparison with `SNLSDP` [11] and `ESDP` [12] is provided in Table I. For each noise level, we have averaged the results over 100 realizations. For a given N, we randomly picked 10% points and set them as anchors [11]. The ANEs and timings are compared for various network sizes and noise levels in Table I. The proposed solver can find the sensor locations at machine level precision for noiseless scenarios. Moreover, our method outperforms `ESDP` both in terms of the ANE and timing. Note that although the accuracy of `SNLSDP` and our method are comparable, `SNLSDP` cannot be scaled to large networks.

Experiment 6. Finally, we compare the proposed algorithm with `ESDP` on the spiral dataset [14]. This consists of 2259 points and its diameter (distance between two furthest points) is 11.2. We randomly select 226 points (about 10% points) as anchors, and set the radio range to $r = 0.8$. The reconstructions are reported in Fig 6, along with the corresponding ANEs and timings. Notice that, unlike `ESDP`, our method is able to preserve the network structure.

VI. Conclusion

We proposed an iterative solver for rank-constrained SDP with block diagonal constraints. The per-iteration complexity is

TABLE I

COMPARISON OF TIMING AND LOCALIZATION ACCURACY OF THE PROPOSED METHOD WITH SNLSDP [11] AND ESDP [12] FOR RANDOM GEOMETRIC GRAPHS. WE USE A $\star$ TO MEAN THAT THE INTERIOR-POINT SOLVER COULD NOT SOLVE THE SDP IN SNLSDP FOR THAT SETTING.

				Timing			Accuracy (ANE)		
N	K	r	η	Proposed	ESDP [12]	SNLSDP [11]	Proposed	ESDP [12]	SNLSDP [11]
100	10	0.4	0	0.6sec	4.9sec	3.4sec	1.5e-14	1.8e-5	7.6e-10
			0.1	0.6sec	2.4sec	5.1sec	2.4e-2	3.2e-1	2.4e-2
500	50	0.18	0	5.6sec	1.1min	5.9min	2.5e-14	6.6e-7	6.6e-7
			0.1	5.7sec	18.7sec	8.3min	1e-2	2.3e-2	1e-2
1000	100	0.12	0	23.8sec	2.6min	$\star$	3.5e-11	1.4e-6	$\star$
			0.01	23sec	1.2min	$\star$	7.7e-4	1.3e-3	$\star$
4000	400	0.06	0	14.9min	43.2min	$\star$	1e-13	1.8e-6	$\star$
			0.01	14.8 min	17.8min	$\star$	3.9e-4	1e-3	$\star$

essentially the computation of the bottom d eigenvectors of a symmetric matrix. We proved that if the iterates converge, then they do so to a KKT point. Results of numerical simulations were reported to show that the algorithm indeed converges (and often quite rapidly) for the registration problem. Moreover, our solver was shown to compare favorably with existing methods, both in terms of the timing and accuracy. We also showed how the proposed solver can be used for sensor localization by integrating it with the registration-based framework proposed in [10]. This was shown to yield promising results (competitive with existing optimization methods [11], [12]) for both random and structured networks.

VII. APPENDIX

A. Proof of Lemma 1

For a minimization problem with equality constraints, KKT conditions amount to primal feasibility and stationarity of Lagrangian with respect to the primal variables [9]. Primal feasibility gives us condition (a). On the other hand, setting the derivative of (16) with respect to $\mathbf{O}_i$ to zero, we obtain

$$\mathbf{O}_i\mathbf{\Lambda}_i = \sum_{j=1}^{M} \mathbf{O}_j[\mathbf{C}]_{ji}, \quad i \in [\![1, M]\!]. \tag{19}$$

Left multiplying (19) by $\mathbf{O}_i^\top$, we have

$$\mathbf{\Lambda}_i = \sum_{j=1}^{M} \mathbf{O}_i^\top\mathbf{O}_j[\mathbf{C}]_{ji} = \sum_{j=1}^{M}[\mathbf{G}]_{ij}[\mathbf{C}]_{ji} = [\mathbf{GC}]_{ii}.$$

Also, note that $\mathbf{\Lambda}_i^\top = [\mathbf{CG}]_{ii}$. Since $\mathbf{\Lambda}_i$ is symmetric, condition (b) follows immediately. Conversely, it is not difficult to see that conditions (a) and (b) together imply that $\mathbf{G}$ is a stationary point of (15).

B. Proof of Theorem 2

To show that $\mathbf{G}^*$ is a KKT point of (15), we use Lemma 1. Since $\mathbf{G}^* = M\mathbf{X}^{k_0}$ and $\mathbf{X}^{k_0} \in \Omega$, it is clear that $[\mathbf{G}^*]_{ii} = \mathbf{I}_d, i \in [\![1, M]\!]$. This verifies condition (a) in Lemma 1. We now verify condition (b): $[\mathbf{CG}^*]_{ii} = [\mathbf{G}^*\mathbf{C}]_{ii}, i \in [\![1, M]\!]$.

Since $\mathbf{G}^* = M\mathbf{X}^{k_0}$ and $\mathbf{X}^{k_0} = \mathbf{W}^{k_0+1}\mathbf{W}^{k_0+1^\top}$, it follows from (13) that the eigenvectors of $\mathbf{G}^*$ and $\mathbf{B}^{k_0}$ are identical. Therefore, $\mathbf{G}^*$ and $\mathbf{B}^{k_0}$ must commute: $\mathbf{B}^{k_0}\mathbf{G}^* = \mathbf{G}^*\mathbf{B}^{k_0}$. Moreover, since $\mathbf{B}^{k_0} = \mathbf{C} - \mathbf{\Lambda}^{k_0} - \rho_k\mathbf{X}^{k_0}$, we obtain that

$$(\mathbf{C} - \mathbf{\Lambda}^{k_0})\mathbf{G}^* = \mathbf{G}^*(\mathbf{C} - \mathbf{\Lambda}^{k_0}).$$

In particular, $[(\mathbf{C} - \mathbf{\Lambda}^{k_0})\mathbf{G}^*]_{ii} = [\mathbf{G}^*(\mathbf{C} - \mathbf{\Lambda}^{k_0})]_{ii}$. That is,

$$[\mathbf{CG}^*]_{ii} - [\mathbf{\Lambda}^{k_0}\mathbf{G}^*]_{ii} = [\mathbf{G}^*\mathbf{C}]_{ii} - [\mathbf{G}^*\mathbf{\Lambda}^{k_0}]_{ii}.$$

Since the update in (10) affects only the diagonal blocks and $\mathbf{\Lambda}^0 = \mathbf{0}$ by assumption, it is easily verified that $\mathbf{\Lambda}^k$ is block diagonal for $k \geq 0$. Therefore, $[\mathbf{\Lambda}^{k_0}\mathbf{G}^*]_{ii} = [\mathbf{\Lambda}^{k_0}]_{ii}[\mathbf{G}^*]_{ii}$ and $[\mathbf{G}^*\mathbf{\Lambda}^{k_0}]_{ii} = [\mathbf{G}^*]_{ii}[\mathbf{\Lambda}^{k_0}]_{ii}$. Now, since $[\mathbf{G}^*]_{ii} = \mathbf{I}_d$, condition (b) follows.

REFERENCES

[1] M. X. Goemans and D. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *Journal of the ACM*, vol. 42, pp. 1115–1145, 1995.

[2] A. Singer, "Angular synchronization by eigenvectors and semidefinite programming," *Applied and computational harmonic analysis*, vol. 30, no. 1, pp. 20–36, 2011.

[3] K. N. Chaudhury, Y. Khoo, and A. Singer, "Global registration of multiple point clouds using semidefinite programming," *SIAM Journal on Optimization*, vol. 25, no. 1, pp. 468–501, 2015.

[4] R. Sanyal, S. M. Ahmed, M. Jaiswal, and K. N. Chaudhury, "A scalable ADMM algorithm for rigid registration," *IEEE Signal Processing Letters*, vol. 24, no. 10, pp. 1453–1457, 2017.

[5] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[6] C. Lu, J. Feng, Z. Lin, and S. Yan, "Nonconvex sparse spectral clustering by alternating direction method of multipliers and its convergence analysis," *Proc. AAAI Conference on Artificial Intelligence*, 2018.

[7] Y. Wang, W. Yin, and J. Zeng, "Global convergence of ADMM in nonconvex nonsmooth optimization," *arXiv preprint arXiv:1511.06324*, 2015.

[8] M. Hong, Z.-Q. Luo, and M. Razaviyayn, "Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems," *SIAM Journal on Optimization*, vol. 26, no. 1, pp. 337–364, 2016.

[9] D. P. Bertsekas, *Nonlinear Programming*. Athena scientific Belmont, 1999.

[10] R. Sanyal, M. Jaiswal, and K. N. Chaudhury, "On a registration-based approach to sensor network localization," *IEEE Trans. Signal Processing*, vol. 65, no. 20, pp. 5357–5367, 2017.

[11] 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 Trans. Automation Science and Enggineering*, vol. 3, no. 4, pp. 360–371, 2006.

[12] Z. Wang, S. Zheng, Y. Ye, and S. Boyd, "Further relaxations of the semidefinite programming approach to sensor network localization," *SIAM Journal on Optimization*, vol. 19, no. 2, pp. 655–673, 2008.

[13] K. S. Arun, T. S. Huang, and S. D. Bolstein, "Least-squares fitting of two 3-D point sets," *IEEE Trans. Pattern Analysis and Machine Intelligence*, no. 5, pp. 698–700, 1987.

[14] M. Cucuringu, Y. Lipman, and A. Singer, "Sensor network localization by eigenvector synchronization over the Euclidean group," *ACM Trans. Sensor Networks*, vol. 8, no. 3, pp. 19–42, 2012.