

SOLVING LOW-RANK SEMIDEFINITE PROGRAMS VIA MANIFOLD OPTIMIZATION *

JIE WANG[†] AND LIANGBING HU[‡]

Abstract. We propose a manifold optimization approach to solve linear semidefinite programs (SDP) with low-rank solutions. This approach incorporates the augmented Lagrangian method and the Burer-Monteiro factorization, and features the adaptive strategies for updating the factorization size and the penalty parameter. We prove that the present algorithm can solve SDPs to global optimality, despite of the non-convexity brought by the Burer-Monteiro factorization. Along with the algorithm, we release an open-source SDP solver **ManiSDP**. Comprehensive numerical experiments demonstrate that **ManiSDP** achieves state-of-the-art in terms of efficiency, accuracy and scalability, and is faster than several advanced SDP solvers (**MOSEK**, **SDPLR**, **SDPNAL+**, **STRIDE**) by up to orders of magnitudes on a variety of linear SDPs. The largest SDP solved by **ManiSDP** (in about 8.5 hours with maximal KKT residue 3.5e-13) is the second-order moment relaxation of a binary quadratic program with 120 variables, which has matrix dimension 7261 and contains 17,869,161 affine constraints.

Key words. semidefinite programming, low-rank solution, polynomial optimization, moment relaxation, Burer-Monteiro factorization, augmented Lagrangian method, manifold optimization

AMS subject classifications. Primary, 90C22; Secondary, 90C23,90C30

1. Introduction. In this paper, we aim to efficiently solve the following semi-definite programming (SDP) problem:

$$(\text{SDP}) \quad \begin{cases} \inf_{X \succeq 0} & \langle C, X \rangle \\ \text{s.t.} & \mathcal{A}(X) = b, \mathcal{B}(X) = d, \end{cases}$$

where $\mathcal{A} : \mathbb{S}_n \rightarrow \mathbb{R}^m$, $\mathcal{B} : \mathbb{S}_n \rightarrow \mathbb{R}^l$ are linear maps (\mathbb{S}_n denotes the set of $n \times n$ symmetric matrices), $b \in \mathbb{R}^m, d \in \mathbb{R}^l$. In (SDP) the linear constraints $\mathcal{A}(X) = b$ are arbitrary while the linear constraints $\mathcal{B}(X) = d$, if present, are assumed to define certain manifold structure. Moreover, we assume that (SDP) admits a low-rank solution X^* , i.e., $\text{rank}(X^*) \ll n$.

Such (SDP) arises from a diverse set of fields, e.g., systems and control [28], signal processing [17, 26], optimal power flow [4], matrix completion [11], quantum steering [13], computer vision [44], to just name a few; see also the surveys [36, 42] and references therein. It also serves as a tractable convex relaxation for many difficult (usually NP-hard) non-convex optimization problems, e.g., quadratically constrained quadratic programs (QCQP) [26], combinatorial optimization problems [19], polynomial optimization problems [24].

In view of the wide applications, a large amount of efforts have been dedicated to solving (SDP) over the past decades and lots of practical algorithms have been developed from various angles. For small/medium-scale SDPs, interior-point methods are believed to be the most accurate, efficient and robust algorithms [3, 35]. However, for large-scale SDPs interior-point methods are no longer reliable because of their extensive memory occupation and high computational costs in solving a large and dense

*Submitted to the editors DATE.

Funding: JW was supported by the NSFC grant 12201618.

[†]Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China (wangjie212@amss.ac.cn)

[‡]Nanjing Research Institute of Electronics Technology, Nanjing, China (huliangbing2000@163.com)

linear system at each iteration. Aiming to tackle large-scale SDPs, Yang, Sun and Toh proposed an augmented Lagrangian algorithm combined with the semismooth Newton method [47]. The related solver **SDPNAL+** has been shown to have good scalability on *degenerate* SDPs [33]. There are also quite a few attempts to overcome the memory issue by relying on first-order methods. For instance, Wen et al. proposed an alternating-direction augmented Lagrangian method (ALM) [41]. The main drawback of first-order methods is that they can hardly achieve high accuracy.

In many cases, large-scale SDPs possess certain structures, e.g., chordal sparsity, constant trace, admitting low-rank solutions. Such structures can be exploited to speed up computation either in interior-point methods or in first-order methods. Zhang and Lavaei proposed to exploit chordal sparsity for interior-point methods via dualized clique tree conversion [50]. Zheng et al. [51] and Garstka et al. [18] proposed to exploit chordal sparsity in a framework of the alternating direction methods of multipliers (ADMM). By exploiting the constant trace property, Helmberg and Rendl proposed a spectral bundle method for solving (SDP) [21]. More recently, Yurtsever et al. proposed to exploit the constant trace property in a conditional-gradient-based augmented Lagrangian framework combined with a matrix sketching technique [48]. The property of admitting low-rank solutions can be exploited in the framework of interior-point methods [5, 20, 49] or in the framework of operator splitting methods [32]. For SDPs arising as convex relaxations of polynomial optimization problems and admitting rank-one solutions, Yang et al. proposed a projected gradient method accelerated by local search [45], which is scalable on a variety of SDPs emerging from computer vision problems [44].

Another notable approach taking the low-rank property into account is to perform a factorization $X = YY^\top$ with $Y \in \mathbb{R}^{n \times p}$, $p \ll n$ so that (SDP) becomes a non-convex QCQP (called the factorized problem), which is called the *Burer-Monteiro factorization* in the literature [9, 10], leading to a low-rank SDP solver **SDPLR**. For the factorized problem, nonlinear programming tools can find a second-order critical point at much less cost, and under mild conditions, this second-order critical point is guaranteed to be globally optimal whenever $\frac{p(p+1)}{2} > m$ [7, 8, 15]. Hence for the approach being efficient, m is required not to be too large. Recent treatments of this approach for certifiably correct machine perception and the graph equipartition problem could be found in [30, 34].

Related literature. Following Burer and Monteiro’s idea, Journée et al. reformulated (SDP) as a nonlinear program on a manifold under the assumption that there is no arbitrary linear constraint (i.e., $m = 0$) and the feasible set $\mathcal{N}_p = \{Y \in \mathbb{R}^{n \times p} \mid \mathcal{B}(YY^\top) = d\}$ is a smooth manifold [22]. Consequently, the nonlinear program becomes a Riemannian optimization problem which can be efficiently solved by off-the-shelf Riemannian optimization tools. They also provided a strategy for escaping from saddle points so that global convergence can be guaranteed. See [29] for an extension of this approach to the regularized convex matrix optimization problem and [31] to the special Euclidean synchronization problem. A drawback of Journée et al.’s approach is that they did not deal with linear constraints that do not define a manifold. The first work to treat constrained optimization on Riemannian manifolds using an augmented Lagrangian framework appears to be [25]. Later, Zhou et al. developed an ALM for solving a class of manifold optimization problems with non-smooth objective functions and nonlinear constraints [52]. Building on constrained Riemannian optimization, Wang et al. recently extended Journée et al.’s approach for solving SDPs with nonsmooth objective functions and arbitrary linear constraints

$\mathcal{A}(X) = b$ within an augmented Lagrangian framework in which a Riemannian semismooth Newton method is employed to solve the augmented Lagrangian subproblem [40].

Contributions. Our contributions are threefold:

- We present a manifold optimization algorithm for solving (SDP) by adopting the Burer-Monteiro factorization and the idea of Journée et al.. More concretely, our algorithm employs an ALM framework to handle arbitrary affine constraints and applies the Burer-Monteiro factorization to the ALM subproblem in order to exploit the low-rank property. The ALM subproblem is then recast as a manifold optimization problem which is solved using the Riemannian trust-region method. To circumvent the non-convexity introduced by the Burer-Monteiro factorization, we design a strategy (inspired by Journée et al.) for escaping from saddle points. In addition, our algorithm possesses two distinguished features: (i) the factorization size p is dynamically adjusted so that the number of decision variables of the ALM subproblem is as small as possible; (ii) the penalty parameter used in the ALM is self-adaptively increased or decreased. These two features significantly improve the practical performance of the algorithm. Although the overall schemes are similar (both incorporate the Burer-Monteiro factorization, the ALM and manifold optimization), our algorithm differs from [40] in four aspects: (1) our algorithm concentrates on linear and smooth SDPs while [40] deals with more general nonlinear and nonsmooth SDPs; (2) we utilize the Riemannian trust-region method to solve the augmented Lagrangian subproblem while [40] utilizes a Riemannian semismooth Newton method; (3) the strategy of updating the factorization size (as well as escaping from saddle points) is different from [40]; (4) we propose a *novel* and *simple* strategy of self-adaptively updating the penalty parameter which allows to decrease the penalty parameter. We stress that a large penalty parameter may make the augmented Lagrangian subproblem difficult to solve.

- Despite of the non-convexity, we prove global convergence of the proposed algorithm, that is, we show that under certain conditions, the output of the algorithm satisfies the KKT conditions of (SDP) up to a small tolerance.

- As another main contribution, we release an open-source SDP solver named **ManiSDP** that implements the proposed algorithm. Extensive numerical experiments were performed to benchmark the solver, which demonstrate that **ManiSDP** is accurate, efficient and scalable on a variety of SDPs with low-rank solutions, and (substantially) outperforms a few popular SDP solvers.

The rest of the paper is organized as follows. In Section 2, we collect notations and some preliminary results. In Section 3, we give the augmented Lagrangian framework with the Burer-Monteiro factorization. In Section 4, we describe some computational details on solving the ALM subproblem with the Riemannian trust-region method. In Section 5, we describe the strategies of updating the factorization size and the penalty parameter, and prove global convergence of the algorithm. Results of numerical experiments are provided in Section 6. Conclusions are made in Section 7.

2. Notation and preliminaries. \mathbb{R} (resp. \mathbb{N}) denotes the set of real numbers (resp. nonnegative integers). For a positive integer n , let $[n] := \{1, 2, \dots, n\}$. Let \mathbb{S}_n (resp. \mathbb{S}_n^+) denote the set of (resp. positive semidefinite/PSD) symmetric matrices of size n . We use $\text{Tr}(A)$ (resp. A^\top) to denote the trace (resp. transpose) of a matrix $A \in \mathbb{R}^{n \times m}$. For two matrix $A, B \in \mathbb{R}^{n \times m}$, the inner product is defined as $\langle A, B \rangle = \text{Tr}(A^\top B)$. For $A \in \mathbb{R}^{n \times n}$, $\text{diag}(A)$ denotes the diagonal of A and $\text{Diag}(A)$ denotes the diagonal matrix with the same diagonal as A . For a vector v , $\|v\|$ is the

2-norm of v and for a matrix A , $\|A\|_F$ is the Frobenius norm of A . For a function $f(x)$, we write $\nabla f(x)$ (resp. $\text{grad } f(x)$) for the Euclidean (resp. Riemannian) gradient, and write $\nabla^2 f(x)[u]$ (resp. $\text{Hess } f(x)[u]$) for the Euclidean (resp. Riemannian) Hessian acting on u . For a set E , $|E|$ denotes its cardinality.

Let us consider (SDP) with $\mathcal{A}(X) := (\langle A_i, X \rangle)_{i=1}^m$ and $\mathcal{B}(X) := (\langle B_i, X \rangle)_{i=1}^l$ for $A_i, B_i \in \mathbb{S}_n$. Let $\mathcal{A}^* : \mathbb{R}^m \rightarrow \mathbb{S}_n$ be the adjoint operator of \mathcal{A} defined as $\mathcal{A}^*(y) := \sum_{i=1}^m y_i A_i$ for $y \in \mathbb{R}^m$; similarly, let $\mathcal{B}^* : \mathbb{R}^l \rightarrow \mathbb{S}_n$ be the adjoint operator of \mathcal{B} defined as $\mathcal{B}^*(z) := \sum_{i=1}^l z_i B_i$ for $z \in \mathbb{R}^l$. Throughout the paper, we assume that strong duality holds for (SDP).

LEMMA 2.1. *A matrix $X \in \mathbb{S}_n$ is a minimizer of (SDP) if and only if there exist Lagrange multipliers $y \in \mathbb{R}^m$ and $z \in \mathbb{R}^l$ such that*

$$\begin{aligned} (2.1a) \quad & \mathcal{A}(X) = b, \\ (2.1b) \quad & \mathcal{B}(X) = d, \\ (2.1c) \quad & X \succeq 0, \\ (2.1d) \quad & S := C - \mathcal{A}^*(y) - \mathcal{B}^*(z) \succeq 0, \\ (2.1e) \quad & XS = 0. \end{aligned}$$

Proof. These are the standard KKT conditions for (SDP). \square

3. An augmented Lagrangian framework with the Burer-Monteiro factorization. In this section, we introduce an augmented Lagrangian framework combined with the Burer-Monteiro factorization for solving (SDP).

3.1. An augmented Lagrangian framework for solving semidefinite programs. Recall that the constraints $\mathcal{B}(X) = d$ in (SDP) define certain manifold structure (which will be rigorously defined later). Let us denote

$$(3.1) \quad \mathcal{M} := \{X \in \mathbb{R}^{n \times n} \mid \mathcal{B}(X) = d, X \succeq 0\}.$$

Note that if the constraints $\mathcal{B}(X) = d$ are not present (i.e., $l = 0$), then $\mathcal{M} = \mathbb{S}_n^+$. Other typical choices of \mathcal{M} are

$$\begin{aligned} (\text{Unit-trace}) \quad & \mathcal{M} = \{X \in \mathbb{R}^{n \times n} \mid X \succeq 0, \text{Tr}(X) = 1\}, \\ (\text{Unit-diagonal}) \quad & \mathcal{M} = \{X \in \mathbb{R}^{n \times n} \mid X \succeq 0, \text{diag}(X) = \mathbf{1}\}. \end{aligned}$$

Remark 3.1. For the case that X has a constant trace c , we can scale X by the factor $\frac{1}{c}$ to match the (Unit-trace) case.

Then (SDP) can be equivalently written as

$$(SDP\text{-M}) \quad \begin{cases} \inf_{X \in \mathcal{M}} & \langle C, X \rangle \\ \text{s.t.} & \mathcal{A}(X) = b. \end{cases}$$

For $p \in [n]$, we denote by \mathcal{N}_p the set of matrices after applying the Burer-Monteiro factorization to \mathcal{M} , i.e.,

$$(3.2) \quad \mathcal{N}_p := \{Y \in \mathbb{R}^{n \times p} \mid YY^\top \in \mathcal{M}\} = \{Y \in \mathbb{R}^{n \times p} \mid \mathcal{B}(YY^\top) = d\}.$$

We call p the *factorization size*. For $\mathcal{M} = \mathbb{S}_n^+$ or \mathcal{M} in **(Unit-trace)**, **(Unit-diagonal)**, the corresponding \mathcal{N}_p are

$$\begin{aligned} \text{(Euclidean)} \quad & \mathcal{N}_p = \{Y \in \mathbb{R}^{n \times p}\}, \\ \text{(Sphere)} \quad & \mathcal{N}_p = \{Y \in \mathbb{R}^{n \times p} \mid \|Y\|_F = 1\}, \\ \text{(Oblique)} \quad & \mathcal{N}_p = \{Y \in \mathbb{R}^{n \times p} \mid \|Y(i, \cdot)\| = 1, i = 1, \dots, n\}, \end{aligned}$$

where $Y(i, \cdot)$ stands for the i -th row of Y .

We further make the following assumptions on **(SDP-M)**:

ASSUMPTION 3.2. **(SDP-M)** admits a low-rank optimal solution.

ASSUMPTION 3.3. \mathcal{N}_p is a submanifold embedded in the Euclidean space $\mathbb{R}^{n \times p}$.

ASSUMPTION 3.4. Either $l = 0, 1$, or the matrices $\{B_i\}_{i=1}^l$ satisfy $B_i B_j = 0$ for any $1 \leq i \neq j \leq l$.

ASSUMPTION 3.5. $d_i \neq 0$ for $i = 1, \dots, l$.

It is clear that for \mathcal{N}_p in **(Euclidean)** or **(Sphere)** with $p \geq 1$, or \mathcal{N}_p in **(Oblique)** with $p \geq 2$, Assumptions 3.3–3.5 are satisfied.

Remark 3.6. It is possible to weaken Assumptions 3.4–3.5 to the assumption that the matrices $\{B_i Y\}_{i=1}^l$ are linearly independent in $\mathbb{R}^{n \times p}$ for all $Y \in \mathcal{N}_p$; see [8] for detailed discussions.

The augmented Lagrangian function associated with **(SDP-M)** is defined by

$$(3.4) \quad L_\sigma(X, y) = \langle C, X \rangle - y^\top (\mathcal{A}(X) - b) + \frac{\sigma}{2} \|\mathcal{A}(X) - b\|^2.$$

As for usual constrained optimization problems, **(SDP-M)** can be solved within an ALM which is presented in Algorithm 3.1 (c.f. [40]).

Algorithm 3.1

Input: $\mathcal{A}, b, \mathcal{B}, d, C, \sigma_0 > 0$

Output: (X, y, S)

- 1: $k \leftarrow 0, y^0 \leftarrow 0$;
 - 2: **while** stopping criteria do not fulfill **do**
 - 3: Solve the ALM subproblem (3.5) to obtain X^{k+1} ;
 - 4: $y^{k+1} \leftarrow y^k - \sigma_k (\mathcal{A}(X^{k+1}) - b)$;
 - 5: Compute S^{k+1} ;
 - 6: Determine σ_{k+1} ;
 - 7: $k \leftarrow k + 1$;
 - 8: **end while**
 - 9: **return** (X^k, y^k, S^k) ;
-

At the k -th iteration of Algorithm 3.1, we need to solve the ALM subproblem

$$(3.5) \quad \min_{X \in \mathcal{M}} \Phi_k(X) := L_{\sigma^k}(X, y^k).$$

The following lemma characterizes the optimality conditions of (3.5).

LEMMA 3.7. A matrix $X \in \mathcal{M}$ is a minimizer of (3.5) if and only if there exist Lagrange multipliers $z \in \mathbb{R}^l$ such that

$$(3.6a) \quad S := \nabla\Phi_k(X) - \mathcal{B}^*(z) \succeq 0,$$

$$(3.6b) \quad XS = 0,$$

where $\nabla\Phi_k(X) = C + \sigma_k \mathcal{A}^*(\mathcal{A}(X) - b - y^k/\sigma_k)$. Furthermore, under Assumptions 3.4–3.5, the vector z is unique and has a closed-form expression:

$$(3.7) \quad z_i := \frac{\text{Tr}(B_i X \nabla\Phi_k(X))}{\text{Tr}(B_i^2 X)} \text{ for } i = 1, \dots, l.$$

Proof. As (3.5) is convex, X is a minimizer if and only if the KKT conditions hold, i.e., there exist $z \in \mathbb{R}^l$ such that

$$(3.8a) \quad \mathcal{B}(X) = d,$$

$$(3.8b) \quad X \succeq 0,$$

$$(3.8c) \quad S \succeq 0,$$

$$(3.8d) \quad XS = 0.$$

Since $X \in \mathcal{M}$, (3.8a) and (3.8b) are valid by definition, and so X is a minimizer if and only if (3.6a) and (3.6b) hold for some $z \in \mathbb{R}^l$. To get the closed-form expression of z , we note that $XS = 0$ can be rewritten as

$$(3.9) \quad X \nabla\Phi_k(X) - \sum_{j=1}^l X B_j z_j = 0.$$

For each $i \in [l]$, multiplying (3.9) by B_i and noting $B_i B_j = 0$ for $i \neq j$, we get

$$(3.10) \quad X \nabla\Phi_k(X) B_i - X B_i^2 z_i = 0.$$

Let $X = Y Y^\top$ for some $Y \in \mathcal{N}_p$. We have

$$\text{Tr}(X B_i^2) = \text{Tr}(Y^\top B_i^2 Y) = \|B_i Y\|_F^2 \neq 0,$$

where the last inequality is because $B_i Y$ cannot be a zero matrix since $d_i \neq 0$. Then (3.7) follows by taking trace of (3.10). \square

Remark 3.8. For \mathcal{M} in (Unit-trace), (3.7) becomes

$$(3.11) \quad z = \text{Tr}(X \nabla\Phi_k(X)),$$

and for \mathcal{M} in (Unit-diagonal), (3.7) becomes

$$(3.12) \quad z = \text{diag}(X \nabla\Phi_k(X)).$$

COROLLARY 3.9. Given a matrix $X \in \mathcal{M}$, let $S := \nabla\Phi_k(X) - \mathcal{B}^*(z)$ with z being given by (3.7). Then X is a minimizer of the convex subproblem (3.5) if and only if $S \succeq 0$.

Proof. This is an immediate consequence of Lemma 3.7. \square

3.2. The Burer-Monteiro factorization. To exploit the fact that (SDP-M) admits a low-rank optimal solution (Assumption 3.2), we now apply the Burer-Monteiro factorization to $\Phi_k(X)$ and define $\Psi_k(Y) = \Phi_k(Y Y^\top)$ for $Y \in \mathbb{R}^{n \times p}$. Consequently, the convex subproblem (3.5) becomes the non-convex factorized subproblem:

$$(3.13) \quad \min_{Y \in \mathcal{N}_p} \Psi_k(Y) = \langle C, Y Y^\top \rangle - (y^k)^\top (\mathcal{A}(Y Y^\top) - b) + \frac{\sigma_k}{2} \|\mathcal{A}(Y Y^\top) - b\|^2.$$

Then, we solve the non-convex factorized subproblem (3.13) on the manifold \mathcal{N}_p (Assumption 3.3) instead of solving the convex subproblem (3.5) on \mathcal{M} . The global optimality condition of (3.13) can be characterized in terms of positive semidefiniteness of the matrix S as well.

PROPOSITION 3.10 ([22], Theorem 4). *Let $Y \in \mathcal{N}_p$, $X = Y Y^\top \in \mathcal{M}$, and z be given by (3.7). Let $S = \nabla \Phi_k(X) - \mathcal{B}^*(z)$. Then Y is a global minimizer of the non-convex factorized subproblem (3.13) if and only if $S \succeq 0$.*

Proof. Note that Y is a global minimizer of (3.13) if and only if X is a minimizer of the convex subproblem (3.5). Then the conclusion follows from Corollary 3.9. \square

THEOREM 3.11. *A matrix $Y \in \mathcal{N}_p$ provides a minimizer $X = Y Y^\top$ of (SDP-M) if and only if $\mathcal{A}(X) = b$ and $S = \nabla \Phi_k(X) - \mathcal{B}^*(z) \succeq 0$ with z being given by (3.7).*

Proof. If X is a minimizer of (SDP-M), then $\mathcal{A}(X) = b$ and X is a minimizer of (3.5). So by Corollary 3.9, $S \succeq 0$. Conversely, by Corollary 3.9, $S \succeq 0$ implies X is a minimizer of (3.5) and so $X S = 0$. Moreover, from $\mathcal{A}(X) = b$, we get $S = C - \mathcal{A}^*(y^k) - \mathcal{B}^*(z)$. Then we see that (X, y^k, z, S) satisfies the KKT conditions (2.1), which yields that X is a minimizer of (SDP-M). \square

4. Solve the ALM subproblem by the Riemannian trust-region method.

Since \mathcal{N}_p is assumed to be a smooth manifold, we can solve the non-convex factorized subproblem (3.13) by off-the-shelf efficient manifold optimization methods. Here we choose the Riemannian trust-region method for its superior performance in both accuracy and efficiency, though other choices are also possible.

4.1. The Riemannian trust-region method. In this subsection, we calculate the ingredients that are necessary to perform optimization on the manifold \mathcal{N}_p via the Riemannian trust-region method. For a detailed introduction to the Riemannian trust-region method, we refer the reader to [1]. First of all, we note that at a point $Y \in \mathcal{N}_p$, the tangent space of \mathcal{N}_p is given by

$$(4.1) \quad T_Y \mathcal{N}_p = \{U \in \mathbb{R}^{n \times p} \mid \mathcal{B}(U Y^\top) = 0\},$$

and the normal space to \mathcal{N}_p is given by

$$(4.2) \quad N_Y \mathcal{N}_p = \left\{ \mathcal{B}^*(u) Y = \sum_{i=1}^l u_i B_i Y \mid u \in \mathbb{R}^l \right\}.$$

LEMMA 4.1. *Let Y be a point on \mathcal{N}_p . The orthogonal projector $P_Y : \mathbb{R}^{n \times p} \rightarrow T_Y \mathcal{N}_p$ is given by*

$$(4.3) \quad P_Y(U) = U - \mathcal{B}^*(u) Y = U - \sum_{i=1}^l u_i B_i Y, \text{ for } U \in \mathbb{R}^{n \times p},$$

where $u \in \mathbb{R}^l$ is determined by

$$(4.4) \quad u_i = \frac{\text{Tr}(B_i U Y^\top)}{\text{Tr}(B_i^2 Y Y^\top)}, \text{ for } i = 1, \dots, l.$$

Proof. Any matrix $U \in \mathbb{R}^{n \times p}$ admits a unique decomposition $U = U_{T_Y \mathcal{N}_p} + U_{N_Y \mathcal{N}_p}$, where $U_{\mathcal{X}}$ belongs to the space \mathcal{X} . So the projection $P_Y(U)$ can be assumed to be of form $P_Y(U) = U - \mathcal{B}^*(u)Y$ for some $u \in \mathbb{R}^l$. Since $P_Y(U) \in T_Y \mathcal{N}_p$, it holds

$$\mathcal{B}(P_Y(U)Y^\top) = \mathcal{B}(UY^\top) - \mathcal{B}(\mathcal{B}^*(u)YY^\top) = 0,$$

i.e., $\text{Tr}(B_i U Y^\top) = \text{Tr}(B_i \mathcal{B}^*(u) Y Y^\top)$ for $i = 1, \dots, l$. This yields (4.4) as $B_i B_j = 0$ for $i \neq j$ by Assumption 3.4. \square

PROPOSITION 4.2 (c.f. [40], Proposition 2.3). *Consider the non-convex factorized subproblem (3.13). Let $X = Y Y^\top$, $\tilde{S} = \nabla \Phi_k(X)$ and $S = \tilde{S} - \mathcal{B}^*(z)$ with z being given in (3.7). Then the Riemannian gradient at Y is given by*

$$(4.5) \quad \text{grad } \Psi_k(Y) = 2SY.$$

For $U \in T_Y \mathcal{N}_p$, let $\tilde{H} = \nabla^2 \Psi_k(Y)[U] = 2(\tilde{S}U + \sigma_k \mathcal{A}^*(\mathcal{A}(YU^\top + UY^\top))Y)$. Then the Riemannian Hessian is given by

$$(4.6) \quad \text{Hess } \Psi_k(Y)[U] = P_Y(\tilde{H}) - 2\mathcal{B}^*(z)U + 2\mathcal{B}^*(u)Y,$$

where

$$(4.7) \quad u_i = \frac{\text{Tr}(B_i^2 U Y^\top) z_i}{\text{Tr}(B_i^2 X)}, \text{ for } i = 1, \dots, l.$$

In particular, for \mathcal{N}_p in (Euclidean), we have

$$(4.8) \quad \text{Hess } \Psi_k(Y)[U] = \tilde{H};$$

for \mathcal{N}_p in (Sphere), we have

$$(4.9) \quad \text{Hess } \Psi_k(Y)[U] = \tilde{H} - \text{Tr}(\tilde{H}Y^\top)Y - 2\text{Tr}(\tilde{S}X)U;$$

for \mathcal{N}_p in (Oblique), we have

$$(4.10) \quad \text{Hess } \Psi_k(Y)[U] = \tilde{H} - \text{Diag}(\tilde{H}Y^\top)Y - 2\text{Diag}(\tilde{S}X)U.$$

Proof. First note $\nabla \Psi_k(Y) = 2\tilde{S}Y$. By (3) of [2], $\text{grad } \Psi_k(Y) = P_Y(\nabla \Psi_k(Y))$ and thus we can write

$$\text{grad } \Psi_k(Y) = 2\tilde{S}Y - 2\mathcal{B}^*(z)Y$$

for some $z \in \mathbb{R}^l$. As $\text{grad } \Psi_k(Y) \in T_Y \mathcal{N}_p$, we have

$$(4.11) \quad \mathcal{B}(\text{grad } \Psi_k(Y)Y^\top) = 2\mathcal{B}((\tilde{S} - \mathcal{B}^*(z))X) = 0.$$

Solving (4.11) for z we get exactly (3.7) and (4.5) then follows.

By (10) of [2], it holds

$$(4.12) \quad \text{Hess } \Psi_k(Y)[U] = P_Y(\nabla^2 \Psi_k(Y)[U]) + \mathfrak{A}_Y(U, P_Y^\perp(\nabla \Psi_k(Y))),$$

where \mathfrak{A}_Y is the Weingarten map at Y and $P_Y^\perp = I - P_Y$ is the orthogonal projector at Y to $N_Y\mathcal{N}_p$. Let $D_Y(\cdot)[U]$ be the directional derivative at Y along U . Then we have

$$\begin{aligned} \mathfrak{A}_Y(U, P_Y^\perp(\nabla\Psi_k(Y))) &= \mathfrak{A}_Y(U, 2\mathcal{B}^*(z)Y) \\ &= -P_Y(D_Y(2\mathcal{B}^*(z)Y)[U]) \\ &= -2P_Y(\mathcal{B}^*(z)U) - 2P_Y(\mathcal{B}^*(D_Y(z)[U])Y) \\ &= -2\mathcal{B}^*(z)U + 2\mathcal{B}^*(u)Y, \end{aligned}$$

where we have used the fact that $P_Y(\mathcal{B}^*(z)U) = \mathcal{B}^*(z)U - \mathcal{B}^*(u)Y$ and $P_Y(\mathcal{B}^*(u')Y) = 0$ for any $u' \in \mathbb{R}^l$. (4.6) then follows.

The remaining conclusions of the proposition can be easily verified. \square

4.2. Escaping from saddle points. Since the subproblem (3.13) is non-convex, in order to solve (3.13) to global optimality, it is then crucial to design a strategy for escaping from saddle points. We next show that we can always compute a second-order descent direction U to escape from saddle points whenever $S = \nabla\Phi_k(X) - \mathcal{B}^*(z) \not\leq 0$.

LEMMA 4.3. *For any $U \in \mathbb{R}^{n \times p}$ satisfying $YU^\top = 0$, it holds*

$$(4.13) \quad \langle U, \text{Hess } \Psi_k(Y)[U] \rangle = 2\text{Tr}(U^\top S U).$$

Proof. The assumption $YU^\top = 0$ implies $U \in T_Y\mathcal{N}_p$. By (4.6) and Lemma 4.1, $\tilde{H} = 2\tilde{S}U$ with $\tilde{S} = \nabla\Phi_k(X)$ and

$$\begin{aligned} \text{Hess } \Psi_k(Y)[U] &= 2P_Y(\tilde{S}U) - 2\mathcal{B}^*(z)U + 2\mathcal{B}^*(u)Y \\ &= 2\tilde{S}U - 2\mathcal{B}^*(u')Y - 2\mathcal{B}^*(z)U + 2\mathcal{B}^*(u)Y \\ &= 2SU + 2\mathcal{B}^*(u - u')Y \end{aligned}$$

for z, u in Proposition 4.2 and some $u' \in \mathbb{R}^l$. Thus,

$$\langle U, \text{Hess } \Psi_k(Y)[U] \rangle = 2\text{Tr}(U^\top S U) + 2\text{Tr}(U^\top \mathcal{B}^*(u - u')Y) = 2\text{Tr}(U^\top S U). \quad \square$$

THEOREM 4.4. *Suppose $S = \nabla\Phi_k(X) - \mathcal{B}^*(z) \succeq 0$ with z being given by (3.7). Let $\delta \in \mathbb{N}$ be a positive number and let $V \in \mathbb{R}^{n \times \delta}$ be a matrix whose columns consist of eigenvectors corresponding to negative eigenvalues of S . Then $U := [0_{n \times p}, V]$ is a second-order descent direction of (3.13) with $p := p + \delta$ at the point $Y := [Y, 0_{n \times \delta}]$, namely, U satisfies*

$$(4.14) \quad \langle U, \text{grad } \Psi_k(Y) \rangle = 0, \quad \langle U, \text{Hess } \Psi_k(Y)[U] \rangle < 0.$$

Proof. By construction, we have $YU^\top = 0$. Therefore, by (4.5),

$$\langle U, \text{grad } \Psi_k(Y) \rangle = 2\text{Tr}(U^\top S Y) = 2\text{Tr}(S Y U^\top) = 0.$$

By Lemma 4.3,

$$\langle U, \text{Hess } \Psi_k(Y)[U] \rangle = 2\text{Tr}(U^\top S U) = 2\text{Tr}(V^\top S V) < 0. \quad \square$$

Remark 4.5. The fact that an eigenvector corresponding to a negative eigenvalue of S yields a second-order descent direction of (3.13) was first observed in [22] where only one eigenvector corresponding to the smallest eigenvalue was used. In [40], the authors used eigenvectors corresponding to all negative eigenvalues to obtain a descent direction.

The following theorem adapted from [22] (see also [40] for an extension to the nonsmooth case) tells us that by escaping from saddle points, we are capable of finding a global minimizer of the non-convex factorized subproblem (3.13).

THEOREM 4.6 ([22], Theorem 7). *A second-order critical point $Y \in \mathcal{N}_p$ of the non-convex factorized subproblem (3.13) provides a minimizer $X = YY^\top$ of the convex subproblem (3.5) if it is rank deficient, i.e., $\text{rank } Y < p$.*

Proof. Let $\text{rank } Y = r < p$. We can write $Y = Y'P^\top$ for some full-rank matrices $Y' \in \mathbb{R}^{n \times r}$ and $P \in \mathbb{R}^{p \times r}$. Let $P_\perp \in \mathbb{R}^{p \times (p-r)}$ be the matrix whose columns form an orthogonal basis of the orthogonal complement of the column space of P such that $P^\top P_\perp = 0$. Let $V \in \mathbb{R}^{n \times (p-r)}$ be an arbitrary matrix and let $U = VP_\perp^\top$. We have $UY^\top = VP_\perp^\top P(Y')^\top = 0$. As Y is a second-order critical point of the non-convex factorized subproblem (3.13), $\text{Hess } \Psi_k(Y)$ is positive semidefinite. Then by Lemma 4.3, we have

$$0 \leq \langle U, \text{Hess } \Psi_k(Y)[U] \rangle = 2\text{Tr}(U^\top S U) = 2\text{Tr}(V^\top S V).$$

The arbitrariness of V implies that S is positive semidefinite and so by Proposition 3.10, $Y \in \mathcal{N}_p$ is a global minimizer of (3.13), which further implies $X = YY^\top$ is a minimizer of (3.5). \square

5. The main algorithm. The main algorithm is summarized in Algorithm 5.1.

Algorithm 5.1 ManiSDP

Input: $A, b, \mathcal{B}, d, C, \sigma_0 > 0, \sigma_{\min} > 0, p_0 \in \mathbb{N}, \gamma > 1, \tau > 0, \delta_{\text{ne}} \in \mathbb{N}, (\epsilon_k)_{k \in \mathbb{N}} \subseteq \mathbb{R}^+$
with $\lim_{k \rightarrow \infty} \epsilon_k = 0, \theta \in (0, 1)$

Output: (X, y, S)

- 1: $k \leftarrow 0, y^0 \leftarrow 0, p \leftarrow p_0, Y^0 \leftarrow 0_{n \times p}, U \leftarrow 0_{n \times p}$;
 - 2: **while** stopping criteria do not fulfill **do**
 - 3: Solve the ALM subproblem (3.13) with U as a descent direction to obtain Y^{k+1} such that $\|\text{grad } \Psi_k(Y^{k+1})\|_F \leq \epsilon_k$;
 - 4: $X^{k+1} \leftarrow Y^{k+1}(Y^{k+1})^\top, y^{k+1} \leftarrow y^k - \sigma_k(\mathcal{A}(X^{k+1}) - b)$;
 - 5: $z_i \leftarrow \frac{\text{Tr}(B_i X^{k+1} \nabla \Phi_k(X^{k+1}))}{\text{Tr}(B_i^2 X^{k+1})}, i = 1, \dots, l$;
 - 6: $S^{k+1} \leftarrow C + \sigma_k \mathcal{A}^*(\mathcal{A}(X^{k+1}) - b - y^k / \sigma_k) - \mathcal{B}^*(z)$;
 - 7: Compute a descent direction U from S^{k+1} according to Theorem 4.4;
 - 8: Update p as described in Section 5.1;
 - 9: **if** $\|\mathcal{A}(X^{k+1}) - b\| / (1 + \|b\|) > \tau \|\text{grad } \Psi_k(Y^{k+1})\|_F$ **then**
 - 10: $\sigma_{k+1} \leftarrow \gamma \sigma_k$;
 - 11: **else**
 - 12: $\sigma_{k+1} \leftarrow \max\{\sigma_k / \gamma, \sigma_{\min}\}$;
 - 13: **end if**
 - 14: $k \leftarrow k + 1$;
 - 15: **end while**
 - 16: **return** (X^k, y^k, S^k) ;
-

In Algorithm 5.1, the initial value p_0 of the factorization size can be typically set to 1 or 2, though for large-scale SDPs, a larger p_0 might be more advantageous; the value of γ is set to 2; the optimal choices for the values of the other parameters are problem-dependent. We discuss two important implementation details of Algorithm 5.1 in the subsequent subsections.

5.1. Dynamically adjusting the factorization size p . In implementation of Algorithm 5.1, the value of the factorization size p is dynamically adjusted. In particular, we increase p to escape from saddle points as discussed in Section 5.1; we decrease p to reduce the size of the ALM subproblem (3.13) so that the computational cost is as low as possible. Suppose that $\{s_i\}_i$ are the singular values of Y sorted from large to small. Then the rank of Y is estimated by

$$(5.1) \quad r = \arg \max_i \{i \mid s_i > \theta s_1\},$$

provided some threshold $\theta \in (0, 1)$. Once the estimated rank r of Y is determined, we are able to construct a rank- r approximation of Y as follows. Suppose that Y has the singular value decomposition

$$Y = WDV^\top,$$

where the diagonal of D is sorted from large to small. We then take

$$Y' = W_r D_r,$$

as a rank- r approximation of Y , where W_r is the submatrix of W consisting of the first r columns and D_r is the upper-left $r \times r$ submatrix of D . Let

$$\delta = \min \{n_{\text{ne}}, \delta_{\text{ne}}\},$$

where n_{ne} is the number of negative eigenvalues of S and $\delta_{\text{ne}} \in \mathbb{N}$ is a tunable parameter. We then update the factorization size p by letting $p = r + \delta$ and accordingly let $Y = [Y', 0_{n \times \delta}]$. To obtain a descent direction, let $U = [0_{n \times r}, v_1, \dots, v_\delta]$ where v_1, \dots, v_δ are the eigenvectors corresponding to the δ smallest eigenvalues of S .

Remark 5.1. A small δ_{ne} typically makes the ALM to converge slowly whereas a large δ_{ne} makes the factorization size p to grow rapidly. Therefore, the value of the parameter δ_{ne} is chosen to balance these two aspects.

Remark 5.2. The above strategy of adjusting the factorization size is adapted from [40] with two adjustments: (1) we rely on a different procedure for estimating ranks; (2) we introduce the parameter δ_{ne} to control the increment of the factorization size at every step.

5.2. Self-adaptively updating the penalty parameter σ . Here we describe the strategy of self-adaptively updating the penalty parameter σ . Unlike usual ALMs using a monotonically nondecreasing sequence of penalty parameters, we allow to increase or decrease the penalty parameter self-adaptively. More concretely, we propose the following updating rules:

$$(5.2) \quad \sigma_{k+1} = \begin{cases} \gamma \sigma_k, & \text{if } \|\mathcal{A}(X^{k+1}) - b\| / (1 + \|b\|) > \tau \|\text{grad } \Psi_k(Y^{k+1})\|_F, \\ \max \{\sigma_k / \gamma, \sigma_{\min}\}, & \text{otherwise,} \end{cases}$$

where $\gamma > 1, \tau > 0, \sigma_{\min} > 0$ are constants. The intuition behind (5.2) is the following: the inequality $\|\mathcal{A}(X^{k+1}) - b\| / (1 + \|b\|) > \tau \|\text{grad } \Psi_k(Y^{k+1})\|_F$ indicates that the progress of feasibility is not satisfactory and hence we increase the penalty parameter by setting $\sigma_{k+1} = \gamma \sigma_k$; otherwise, we decrease the penalty parameter by setting

$\sigma_{k+1} = \max\{\sigma_k/\gamma, \sigma_{\min}\}$ so that the penalty parameter will not become too large or too small. We emphasize that a large penalty parameter probably make the ALM subproblem (3.13) difficult to solve. Numerical experiments demonstrate that (5.2) works well in practice and significantly improves the performance of Algorithm 5.1.

5.3. Global convergence. We now give the global convergence result of Algorithm 5.1.

THEOREM 5.3. *Assume that the sequence $(X^k)_{k \in \mathbb{N}}$ generated by Algorithm 5.1 satisfies $\lim_{k \rightarrow \infty} X^k = X^*$. Assume that the sequences $(y^k)_{k \in \mathbb{N}}$, $(\Psi_k(Y^{k+1}))_{k \in \mathbb{N}}$ are bounded. Then X^* is an optimal solution of (SDP-M). Moreover, let $\lim_{k \rightarrow \infty} y^k = y^*$ and $\lim_{k \rightarrow \infty} S^k = S^*$. Then (X^*, y^*, S^*) is a KKT point of (SDP-M), i.e., it satisfies (2.1) with z being given by*

$$z_i := \frac{\text{Tr}(B_i X^*(C - \mathcal{A}^*(y^*)))}{\text{Tr}(B_i^2 X^*)} \text{ for } i = 1, \dots, l.$$

Proof. We first show that for any threshold $\epsilon > 0$, there exists $K \in \mathbb{N}$ such that $\|\mathcal{A}(X^k) - b\| < \epsilon$ for all $k > K$. Suppose on the contrary that such K does not exist, i.e., $\|\mathcal{A}(X^k) - b\| \geq \epsilon$ for all sufficiently large k . Then Line 10 of Algorithm 5.1 is executed for infinitely many times and so $\lim_{k \rightarrow \infty} \sigma_k = +\infty$. Then note that

$$\Psi_k(Y^{k+1}) = \langle C, X^{k+1} \rangle + \frac{\sigma_k}{2} \left\| \mathcal{A}(X^{k+1}) - b - \frac{y^k}{\sigma_k} \right\|^2 - \frac{(y^k)^2}{2\sigma_k}$$

tends to infinity since $(y^k)_{k \in \mathbb{N}}$ is bounded, which contradicts to the assumption that $(\Psi_k(Y^{k+1}))_{k \in \mathbb{N}}$ is bounded. Therefore, $\|\mathcal{A}(X^k) - b\| = 0$ and X^* is a feasible solution of (SDP-M).

If X^* is not an optimal solution of (SDP-M), then by Theorem 3.11, we have $S^* \not\leq 0$. So there exists a second-order descent direction for (3.13) starting from $Y^* = \lim_{k \rightarrow \infty} Y^k$ due to Theorem 4.4, which contradicts to the fact that X^* is the limit point of $(X^k)_{k \in \mathbb{N}}$.

The last assertion of the theorem can be easily verified. \square

6. Numerical experiments. In this section, we conduct comprehensive numerical experiments to benchmark our solver **ManiSDP** which implements Algorithm 5.1 in MATLAB. In particular, **Manopt 7.1** [6] is employed by **ManiSDP** to solve the Riemannian manifold optimization problem (3.13). **ManiSDP** is freely available at

<https://github.com/wangjie212/ManiSDP-matlab>.

Hardware. All numerical experiments were performed on a desktop computer with Intel(R) Core(TM) i9-10900 CPU@2.80GHz and 64G RAM.

Baseline Solvers. We compare the performance of **ManiSDP** with that of four advanced SDP solvers: **MOSEK 10.0**, **SDPLR 1.03**, **SDPNAL+**, **STRIDE**. We explain why to choose these four baseline solvers: **MOSEK** is chosen as it is a representative interior-point solver; **SDPLR** is chosen as it is a representative solver that also exploits the low-rank property via the Burer-Monteiro factorization; **SDPNAL+** is chosen as it is a representative solver that combines first-order with second-order methods and is designed to solve large-scale SDPs; **STRIDE** is chosen as it specializes to solve large-scale SDP relaxations arising from polynomial optimization problems and exploits the rank-one property. In the following, running time of solvers is measured in seconds; “-” indicates that the solver encounters an out of memory error; “*” indicates that running time exceeds 10,000s; “**” indicates that the solver returns certain numerical error.

Stopping Criteria. To measure the feasibility and optimality of an approximate solution $(X, y, S) \in \mathbb{S}_n^+ \times \mathbb{R}^m \times \mathbb{S}_n^+$, we define the following KKT residues:

$$(6.1) \quad \eta_p = \frac{\|\mathcal{A}(X) - b\|}{1 + \|b\|}, \quad \eta_d = \frac{|\lambda_{\min}(S)|}{1 + |\lambda_{\max}(S)|}, \quad \eta_g = \frac{|\langle C, X \rangle - b^\top y|}{1 + |\langle C, X \rangle| + |b^\top y|}.$$

Given a tolerance $\text{tol} > 0$, the SDP solver terminates when $\eta_{\max} := \max\{\eta_p, \eta_d, \eta_g\} \leq \text{tol}$, and we set $\text{tol} = 1\text{e-}8$ for all our experiments.

Benchmark Problems. To benchmark the solvers, we solve six classes of SDPs arising from different situations (the Max-Cut problem, the matrix completion problem, binary quadratic programs, minimizing quartic polynomials on the unit sphere, the robust rotation search problem, nearest structured rank deficient matrices), with a focus on second-order SDP relaxations for polynomial optimization problems as they are highly degenerate and are challenging for most SDP solvers.

6.1. The Max-Cut problem. The Max-Cut problem is one of the basic combinatorial optimization problems, which is known to be NP-complete. Suppose that $G(V, E)$ is an undirected graph with nodes $V = \{1, \dots, N\}$ and with edge weights $w_{ij} = w_{ji}$ for $\{i, j\} \in E$. Then the Max-Cut problem for G , aiming to find the maximum cut, can be formulated as the following binary quadratic program:

$$(6.1) \quad \begin{cases} \max & \frac{1}{2} \sum_{\{i,j\} \in E} w_{ij} (1 - x_i x_j) \\ \text{s.t.} & 1 - x_i^2 = 0, \quad i = 1, \dots, N. \end{cases}$$

To provide an upper bound on the maximum cut, we can consider the following SDP relaxation for (Max-Cut):

$$(6.2) \quad \begin{cases} \max & \frac{1}{4} \langle L, X \rangle \\ \text{s.t.} & X_{ii} = 1, \quad i = 1, \dots, N, \\ & X \succeq 0, \end{cases}$$

where $L \in \mathbb{S}_N$ is the Laplacian matrix of G , defined by

$$L_{ij} = \begin{cases} -w_{ij}, & \text{if } \{i, j\} \in E, \\ \sum_k w_{ik}, & \text{if } i = j, \\ 0, & \text{otherwise.} \end{cases}$$

Note that (6.2) fits in (SDP-M) with $m = 0$.

We select test graphs from the webpage <https://web.stanford.edu/~yyye/yyye/Gset/> with N varying from 800 to 20000. For each instance, we solve (6.2) using the solvers MOSEK, SDPLR, SDPNAL+, and ManiSDP, respectively. The results are presented in Table 1. The following conclusions can be drawn from the table. (i) MOSEK can solve the instances with $N \leq 10000$ to high accuracy, but the running time significantly grows as N increases. When $N = 20000$, MOSEK runs out of space due to the large memory consumption of interior point methods. (ii) SDPNAL+ is very inefficient in solving this type of SDPs. For instance, when $N \geq 3000$, SDPNAL+ needs over 10000s to output the final result. (iii) Both SDPLR and ManiSDP can solve all instances to high accuracy, while ManiSDP is even more accurate and is faster than SDPLR by a factor of $2 \sim 10$. We refer the reader to [7, 40] for similar experiments on these graphs.

TABLE 1
Results for the Max-Cut problem.

graph	N	MOSEK 10.0		SDPLR 1.03		SDPNAL+		ManiSDP	
		η_{\max}	time	η_{\max}	time	η_{\max}	time	η_{\max}	time
G1	800	2.2e-09	3.51	7.4e-08	5.17	2.1e-09	52.2	1.6e-11	0.54
G2	800	2.7e-09	3.54	2.1e-07	3.29	3.5e-09	52.5	5.3e-14	0.79
G3	800	5.0e-09	3.50	2.1e-07	4.15	4.5e-09	38.6	4.3e-13	0.81
G4	800	2.6e-09	3.41	2.3e-07	3.14	2.4e-09	37.9	9.8e-13	0.58
G22	2000	1.0e-09	49.1	8.2e-08	12.5	8.6e-09	818	8.0e-12	1.48
G23	2000	1.7e-09	51.3	2.7e-07	22.4	2.3e-08	555	5.8e-12	2.09
G24	2000	9.2e-10	49.4	1.4e-06	6.95	4.3e-09	721	3.7e-12	1.36
G25	2000	1.3e-09	53.8	3.6e-07	10.4	2.8e-09	770	1.9e-12	1.43
G32	2000	2.3e-09	45.6	1.2e-07	22.9	7.0e-07	6463	1.6e-09	4.34
G43	1000	4.0e-09	6.31	7.3e-08	2.47	3.0e-09	59.1	2.7e-13	0.68
G44	1000	5.0e-09	6.27	3.1e-07	2.79	1.8e-08	61.2	4.2e-13	0.62
G45	1000	1.1e-09	6.34	2.0e-07	2.73	1.6e-08	61.6	1.5e-12	0.59
G48	3000	2.2e-09	108	1.1e-08	3.99	*	*	1.8e-17	1.81
G49	3000	3.0e-10	100	4.6e-08	4.21	*	*	1.1e-16	1.94
G50	3000	3.9e-14	112	3.0e-08	6.03	*	*	1.2e-14	2.25
G55	5000	3.7e-09	963	2.3e-07	34.5	*	*	6.6e-12	17.7
G56	5000	1.5e-09	847	7.5e-08	23.1	*	*	2.6e-12	15.4
G57	5000	7.0e-10	877	1.5e-07	120	*	*	5.0e-09	30.8
G58	5000	7.7e-10	1081	1.7e-07	101	*	*	1.2e-10	29.7
G59	5000	4.5e-09	931	8.3e-08	63.6	*	*	2.3e-13	30.3
G60	7000	1.1e-09	2722	7.5e-08	67.6	*	*	4.5e-12	35.9
G61	7000	4.6e-10	2735	1.0e-07	114	*	*	4.4e-10	47.0
G62	7000	2.2e-09	2484	3.1e-08	333	*	*	5.1e-09	124
G63	7000	2.6e-09	2978	3.2e-07	224	*	*	9.6e-09	49.7
G64	7000	1.2e-09	2886	5.8e-08	236	*	*	6.2e-09	51.9
G65	8000	1.5e-09	3794	4.6e-08	307	*	*	1.2e-09	127
G66	9000	2.6e-09	5464	5.7e-08	386	*	*	4.3e-09	169
G67	10000	4.6e-09	7363	4.1e-08	610	*	*	7.0e-09	138
G70	10000	4.1e-09	9451	3.1e-07	202	*	*	3.0e-12	73.3
G72	10000	9.7e-11	7728	1.5e-07	614	*	*	8.3e-09	132
G77	14000	*	*	6.9e-08	1177	*	*	1.9e-09	452
G81	20000	-	-	5.2e-08	3520	*	*	8.3e-09	1934

6.2. The matrix completion problem. The matrix completion problem seeks to recover a low-rank matrix $M \in \mathbb{R}^{s \times t}$ from a subset of entries $\{M_{ij}\}_{(i,j) \in \Omega}$. This

can be formulated as the convex optimization problem:

$$(MC) \quad \begin{cases} \inf_{Z \in \mathbb{R}^{s \times t}} & \|Z\|_* \\ \text{s.t.} & Z_{ij} = M_{ij}, \quad \forall (i, j) \in \Omega, \end{cases}$$

where $\|Z\|_* := \text{Tr}(Z^\top Z)^{\frac{1}{2}}$ is the nuclear norm of Z . Note that (MC) can be equivalently cast as an SDP of size $(n, m) = (s + t, |\Omega|)$:

$$(6.3) \quad \begin{cases} \inf_{X \in \mathbb{S}_n} & \text{Tr}(X) \\ \text{s.t.} & \begin{bmatrix} 0_{s \times s} & E_{ij}^\top \\ E_{ij} & 0_{t \times t} \end{bmatrix} X = 2M_{ij}, \quad \forall (i, j) \in \Omega, \\ & X = \begin{bmatrix} U & Z^\top \\ Z & V \end{bmatrix} \succeq 0, \end{cases}$$

where E_{ij} is a $s \times t$ matrix with 1 at its (i, j) -position and 0 otherwise. A famous result by Candes and Recht [11], later improved by Candes and Tao [12] states that, when M is low-rank and incoherent, and the number of samples satisfies $|\Omega| \geq Cn(\log n)^2$ with some constant C , then M can be exactly recovered by solving (6.3). In this subsection, we consider random instances of the matrix completion problem (MC). To this end, we select $\Omega \subseteq [s] \times [t]$ uniformly at random from all subsets with cardinality m , and set $M = M_1 M_2^\top$, where the entries of $M_1 \in \mathbb{R}^{s \times k}$ and $M_2 \in \mathbb{R}^{t \times k}$ are selected i.i.d. from the standard normal distribution. Here, we set $k = 10$, $m = 400n$ and take $s = t = 1000, 1500, 2000, 2500, 3000, 4000, 5000, 6000$ respectively to generate test instances.

For each instance, we solve (6.3) using the solvers MOSEK, SDPLR, SDPNAL+, and ManiSDP, respectively. The results are presented in Table 2, from which we make the following observations. (i) MOSEK cannot solve any instance due to lack of enough memory. (ii) ManiSDP is not only the most efficient but also the most accurate among the remaining three solvers. Particularly, ManiSDP is faster than SDPLR by a factor of $1.5 \sim 2$, and is faster than SDPNAL+ by an order of magnitude.

6.3. Binary quadratic programs. Let us consider the binary quadratic program given by

$$(BQP) \quad \begin{cases} \inf_{\mathbf{x} \in \mathbb{R}^q} & \mathbf{x}^\top Q \mathbf{x} + \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} & x_i^2 = 1, \quad i = 1, \dots, q, \end{cases}$$

where $Q \in \mathbb{S}_q$ and $\mathbf{c} \in \mathbb{R}^q$. (BQP) includes the Max-Cut problem (Max-Cut) as well as many other combinatorial optimization problems as special cases. On the other hand, (BQP) belongs to the more general class of polynomial optimization problems whose objective functions and constraints are given by polynomials. For a polynomial optimization problem, there is a systematic way to construct a hierarchy of increasingly tighter SDP relaxations, known as the moment-SOS hierarchy or the Lasserre hierarchy¹ [24]. The moment SDP relaxation arising from the Lasserre hierarchy typically admits low-rank optimal solutions. Interestingly, for the binary quadratic program (BQP), the second-order moment relaxation is empirically tight on randomly

¹Under mild conditions, the optima of the hierarchy converge to the optimum of the polynomial optimization problem.

TABLE 2
Results for the matrix completion problem.

n	trial	m	MOSEK 10.0		SDPLR 1.03		SDPNAL+		ManiSDP	
			η_{\max}	time	η_{\max}	time	η_{\max}	time	η_{\max}	time
2000	#1	550,536	-	-	1.7e-06	15.1	1.1e-08	69.9	2.1e-10	11.1
	#2	550,565	-	-	1.3e-06	14.7	4.5e-09	131	1.5e-10	10.7
	#3	550,590	-	-	8.6e-07	15.3	4.1e-09	143	7.4e-11	11.7
3000	#1	930,328	-	-	1.8e-06	51.4	3.1e-08	238	7.2e-10	28.0
	#2	929,882	-	-	6.7e-07	49.4	3.2e-08	217	3.6e-10	28.6
	#3	930,080	-	-	3.6e-06	45.4	3.1e-08	216	2.9e-10	28.3
4000	#1	1,318,563	-	-	1.0e-06	88.7	4.8e-08	532	2.7e-10	57.0
	#2	1,318,488	-	-	1.6e-06	99.1	1.9e-09	548	1.4e-10	63.6
	#3	1,318,885	-	-	2.2e-06	96.8	4.7e-08	519	3.2e-10	61.7
5000	#1	1,711,980	-	-	1.2e-06	157	1.4e-09	1143	1.2e-10	104
	#2	1,711,445	-	-	1.0e-06	166	1.6e-09	1084	5.4e-11	103
	#3	1,711,660	-	-	1.1e-06	177	1.3e-09	1111	1.9e-10	104
6000	#1	2,107,303	-	-	2.2e-07	272	2.1e-09	1883	2.5e-10	150
	#2	2,106,628	-	-	1.1e-06	260	2.2e-09	2001	2.8e-11	147
	#3	2,106,039	-	-	1.3e-06	271	2.5e-09	1979	1.3e-10	154
8000	#1	2,900,179	-	-	2.1e-06	498	1.5e-08	3417	9.2e-11	249
	#2	2,900,585	-	-	3.4e-06	449	3.0e-08	4374	3.3e-10	259
	#3	2,900,182	-	-	3.2e-06	490	3.5e-08	4307	1.8e-10	248
10000	#1	3,695,929	-	-	1.1e-06	800	1.4e-09	8370	1.9e-10	446
	#2	3,696,602	-	-	2.1e-06	789	8.6e-09	8849	1.3e-10	456
	#3	3,696,604	-	-	1.1e-06	798	7.2e-09	8502	4.4e-10	472
12000	#1	4,493,420	-	-	7.8e-07	1310	*	*	2.6e-10	661
	#2	4,494,532	-	-	7.1e-07	1291	*	*	2.0e-10	696
	#3	4,493,391	-	-	3.5e-07	1330	*	*	2.2e-11	716

generated instances as observed in [23, 45]. In the following we outline the ingredients of the second-order moment relaxation for (BQP). Let

$$v(\mathbf{x}) := [1, x_1, \dots, x_q, x_1x_2, x_1x_3, \dots, x_{q-1}x_q]^\top$$

be the vector of monomials in \mathbf{x} up to degree two (excluding $x_i^2, i = 1, \dots, q$) and $M := v(\mathbf{x})v(\mathbf{x})^\top$ be the corresponding *moment matrix*. Then the objective function of (BQP) can be linearly expressed in terms of the entries of M . There are linear relationships among the entries of M consisting of $M_{ij} = M_{kr}$ whenever $M_{ij} - M_{kr}$ is reduced to 0 in the Gröbner basis $\{x_i^2 - 1\}_{i=1}^q$. Let $\mathcal{A}(X) = b$ collect all independent linear constraints obtained from these linear relationships when relaxing M to an unknown PSD matrix X . Moreover, because of the constraints $x_i^2 = 1, i = 1, \dots, q$, the diagonal entries of M are all ones and so we let $\mathcal{B}(X) = d$ impose the unit-diagonal constraint on X . Consequently, we obtain the second-order moment relaxation for (BQP).

For each $q \in \{10, 20, 30, 40, 50, 60\}$, we generate three random instances of (BQP) by taking $Q \in \mathbb{S}_q$ with $Q_{ij} \sim \mathcal{N}(0, 1)$ and $\mathbf{c} \in \mathbb{R}^q$ with $c_i \sim \mathcal{N}(0, 1)$. For each instance, we solve the second-order moment relaxation using the solvers MOSEK, SDPLR, SDPNAL+,

STRIDE and ManiSDP, respectively. The sizes of SDPs are recorded in Table 3 and the

TABLE 3
The sizes of SDPs for binary quadratic programs.

q	10	20	30	40	50	60
n	56	211	466	821	1276	1831
m	1,256	16,361	77,316	236,121	564,776	1,155,281

computational results are presented in Table 4. The following conclusions can be drawn from Table 4. (i) MOSEK can solve small-scale instances ($q \leq 20$) to high accuracy, but the running time significantly grows as q increases (< 1 s for $q = 10$ while ~ 50 s for $q = 20$). When $q \geq 30$, MOSEK runs out of space due to large memory consumption. (ii) SDPLR can solve small/medium-scale instances ($q \leq 30$) to medium accuracy, but the running time significantly grows as q increases. When $q \geq 40$, SDPLR needs over 10000s to output the final result. (iii) SDPNAL+ can solve large-scale instances to medium/high accuracy, but the running time is pretty significant for large cases. (iv) Both STRIDE and ManiSDP can solve large-scale instances to high accuracy while ManiSDP is faster than STRIDE by a factor of $2 \sim 35$.

TABLE 4
Results for binary quadratic programs.

q	trial	MOSEK 10.0		SDPLR 1.03		SDPNAL+		STRIDE		ManiSDP	
		η_{\max}	time	η_{\max}	time	η_{\max}	time	η_{\max}	time	η_{\max}	time
10	#1	2.6e-12	0.71	1.5e-06	0.52	1.9e-09	0.65	4.7e-13	0.79	3.9e-15	0.14
	#2	5.7e-14	0.84	6.0e-07	0.53	3.9e-09	1.37	3.4e-10	0.65	3.3e-15	0.18
	#3	8.0e-11	0.67	1.0e-06	1.27	1.5e-08	1.91	6.7e-13	0.68	4.2e-15	0.29
20	#1	9.8e-10	49.0	3.9e-07	30.8	3.0e-09	28.8	7.4e-13	6.12	1.5e-14	0.53
	#2	9.0e-10	50.3	2.3e-08	113	1.7e-08	29.0	6.4e-13	6.98	1.3e-14	0.61
	#3	2.1e-12	47.9	6.6e-08	119	2.3e-07	12.5	2.9e-09	5.86	1.2e-14	0.72
30	#1	-	-	2.1e-06	8384	1.7e-04	187	1.2e-12	65.4	2.8e-14	3.93
	#2	-	-	2.7e-07	2796	6.4e-09	95.5	3.1e-09	36.2	3.2e-14	2.96
	#3	-	-	1.6e-06	5698	7.8e-08	156	1.0e-12	60.3	2.8e-14	4.01
40	#1	-	-	*	*	2.1e-08	813	4.4e-13	249	4.6e-14	10.5
	#2	-	-	*	*	1.3e-06	1514	8.5e-09	294	4.7e-14	8.50
	#3	-	-	*	*	1.3e-07	857	1.6e-12	321	4.4e-14	10.0
50	#1	-	-	*	*	1.6e-07	3058	7.8e-09	826	6.4e-14	31.1
	#2	-	-	*	*	4.8e-08	6347	1.8e-12	1020	8.9e-14	42.8
	#3	-	-	*	*	7.0e-09	4800	8.2e-13	702	7.6e-14	61.4
60	#1	-	-	*	*	*	*	1.3e-12	2118	9.4e-14	94.3
	#2	-	-	*	*	*	*	*	*	9.5e-14	566
	#3	-	-	*	*	*	*	3.3e-12	2704	8.7e-14	150

In Figures 1 and 2, the factorization size and the maximal KKT residue per iteration in solving a random instance of (BQP) are shown for $q = 10, 20, 30, 40, 50, 60$, respectively.

To test the limit of ManiSDP, we run ManiSDP to solve the second-order moment relaxation of (BQP) with larger q . As shown in Table 5, ManiSDP can scale up to $q = 120$ for which the SDP has matrix dimension $n = 7261$ and contains $m = 17, 869, 161$

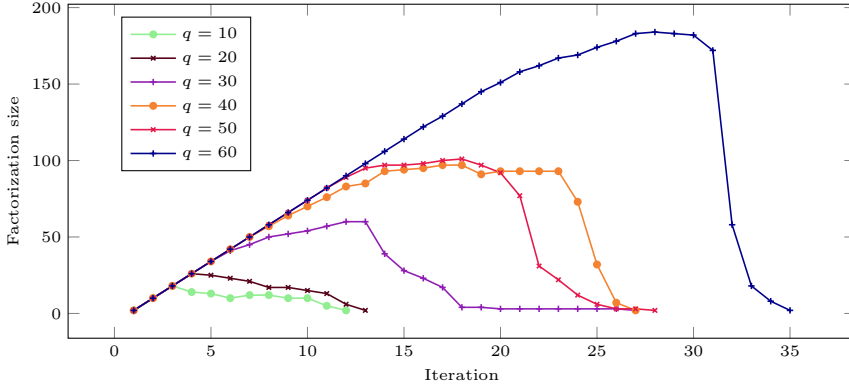


FIG. 1. The factorization size per iteration in solving (BQP).

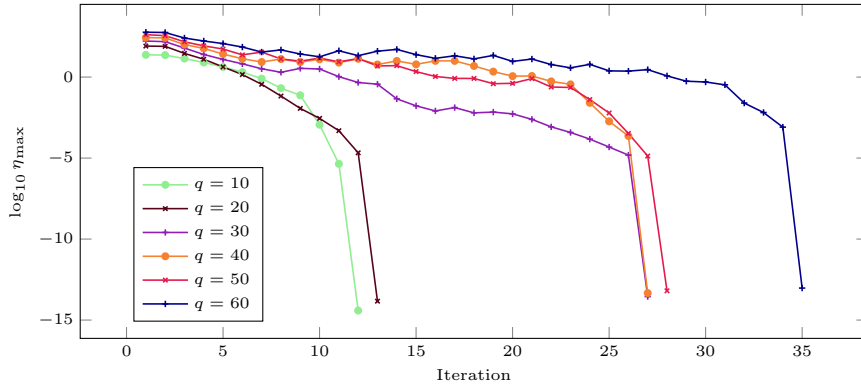


FIG. 2. The maximal KKT residue per iteration in solving (BQP).

affine constraints, far beyond the reach of other SDP solvers!

TABLE 5
Results for large binary quadratic programs via ManiSDP.

q	70	80	90	100	110	120
n	2486	3241	4096	5051	6106	7261
m	2,119,636	3,589,841	5,717,896	8,675,801	12,655,556	17,869,161
η_{\max}	1.4e-13	1.7e-13	2.2e-13	2.5e-13	3.0e-13	3.5e-13
time	1050	1132	3279	5249	7053	30801

6.4. Minimizing quartic polynomials on the unit sphere. Let us consider the problem of minimizing a quartic polynomial on the unit sphere:

$$(QS) \quad \begin{cases} \inf_{\mathbf{x} \in \mathbb{R}^q} & \mathbf{c}^\top \cdot [\mathbf{x}]_4 \\ \text{s.t.} & \sum_{i=1}^q x_i^2 = 1, \end{cases}$$

where $[\mathbf{x}]_4$ is the vector of monomials in \mathbf{x} up to degree four and $\mathbf{c} \in \mathbb{R}^{|\mathbf{x}|_4}$. As for (BQP), the second-order moment relaxation is empirically tight on randomly gener-

ated instances of (QS) [45]. Let

$$v(\mathbf{x}) := [1, x_1, \dots, x_q, x_1^2, x_1x_2, x_1x_3, \dots, x_{q-1}x_q, x_q^2]^\top$$

be the vector of monomials in \mathbf{x} up to degree two and $M := v(\mathbf{x})v(\mathbf{x})^\top$ be the corresponding moment matrix. Then the objective function of (QS) can be linearly expressed in terms of the entries of M . There are linear relationships among the entries of M consisting of all $M_{ij} = M_{kr}$. In addition, for each monomial $w \in v(\mathbf{x})$, the constraint $\sum_{i=1}^q x_i^2 = 1$ gives $w(\sum_{i=1}^q x_i^2 - 1) = 0$ which can be also linearly expressed in terms of the entries of M . Let $\mathcal{A}(X) = b$ collect all independent linear constraints obtained from these linear relationships when relaxing M to an unknown PSD matrix X . We therefore obtain the second-order moment relaxation for (QS).

For each $q \in \{10, 20, 30, 40, 50, 60\}$, we generate three random instances of (QS) by taking $\mathbf{c} \in \mathbb{R}^{|\mathcal{X}|_4}$ with $c_i \sim \mathcal{N}(0, 1)$. For each instance, we solve the second-order moment relaxation using the solvers MOSEK, SDPLR, SDPNAL+, STRIDE and ManiSDP, respectively. The results are presented in Table 6, from which we can draw the following conclusions. (i) MOSEK can solve small-scale instances ($q \leq 20$) to high accuracy, but the running time significantly grows as q increases (< 1 s for $q = 10$ while ~ 50 s for $q = 20$). When $q \geq 30$, MOSEK runs out of space due to large memory consumption. (ii) SDPLR can solve all instances to medium accuracy, but the running time significantly grows as q increases. (iii) ManiSDP is the most efficient solver among the remaining three solvers. (iv) SDPNAL+ attains only medium accuracy for large-scale instances whereas STRIDE and ManiSDP can always attain high accuracy.

TABLE 6
Results for minimizing quartic polynomials on the unit sphere.

q	trial	MOSEK 10.0		SDPLR 1.03		SDPNAL+		STRIDE		ManiSDP	
		η_{\max}	time	η_{\max}	time	η_{\max}	time	η_{\max}	time	η_{\max}	time
10	#1	6.6e-11	0.79	2.0e-07	0.07	2.5e-09	0.45	3.9e-12	0.35	4.5e-09	0.18
	#2	8.8e-10	0.80	9.2e-07	0.56	1.3e-09	0.54	2.6e-12	0.52	7.0e-10	0.30
	#3	5.9e-10	0.79	9.0e-07	0.04	2.1e-09	0.54	2.3e-11	0.38	1.7e-10	0.18
20	#1	7.4e-09	42.5	3.5e-06	1.35	1.2e-09	3.27	2.5e-12	2.74	3.7e-10	0.95
	#2	4.0e-10	49.3	3.7e-07	2.34	5.6e-09	3.36	4.7e-11	2.91	5.4e-10	1.13
	#3	1.0e-08	42.9	5.2e-08	1.10	9.9e-09	3.20	8.7e-13	3.01	1.1e-09	0.82
30	#1	-	-	1.5e-06	20.8	1.2e-09	20.9	3.1e-11	18.6	1.7e-10	6.46
	#2	-	-	7.4e-07	38.4	1.5e-10	21.4	3.2e-13	19.4	8.4e-09	5.63
	#3	-	-	1.8e-07	19.6	1.1e-09	19.0	1.1e-12	22.3	3.7e-10	5.92
40	#1	-	-	3.5e-07	689	1.0e-07	45.0	2.8e-13	42.8	4.3e-09	28.7
	#2	-	-	6.2e-07	272	1.4e-07	28.8	1.0e-12	39.1	4.2e-09	19.4
	#3	-	-	1.4e-07	261	3.9e-06	24.9	4.2e-11	39.3	8.9e-09	20.1
50	#1	-	-	8.0e-07	1588	5.2e-07	68.9	2.3e-12	115	4.8e-09	61.4
	#2	-	-	8.2e-08	1183	2.9e-06	69.0	8.4e-11	105	2.6e-09	49.9
	#3	-	-	3.9e-07	2350	1.1e-06	71.4	5.1e-11	124	4.7e-09	57.1
60	#1	-	-	2.4e-07	4167	5.7e-07	177	2.6e-12	194	3.9e-09	109
	#2	-	-	1.0e-08	7229	3.4e-07	237	3.6e-13	288	6.5e-10	116
	#3	-	-	3.6e-08	7752	4.7e-07	195	4.4e-13	209	2.1e-09	173

In Figures 3 and 4, the factorization size and the maximal KKT residue per iteration in solving a random instance of (QS) are shown for $q = 10, 20, 30, 40, 50, 60$,

respectively.

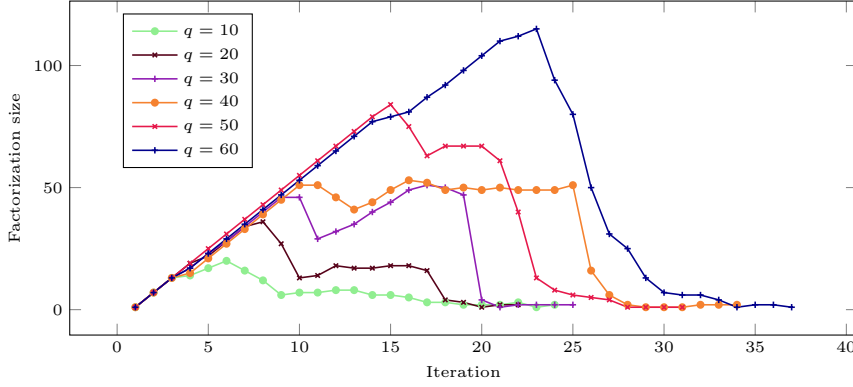


FIG. 3. The factorization size per iteration in solving (QS).

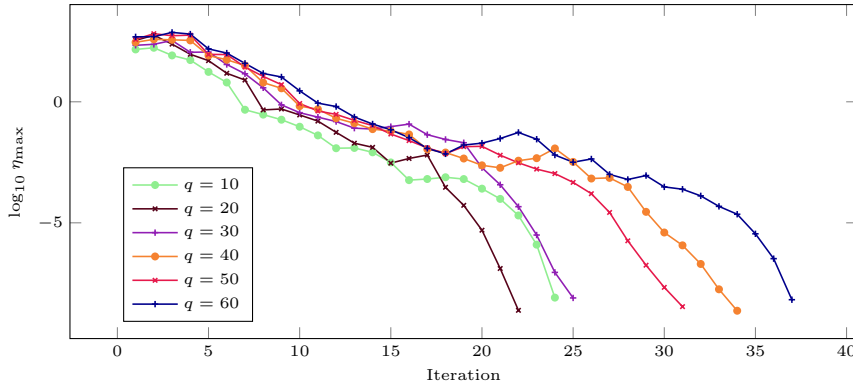


FIG. 4. The maximal KKT residue per iteration in solving (QS).

6.5. The robust rotation search problem. The robust rotation search problem (also known as the Wahba problem with outliers) is to search for the best 3D rotation to align two sets of 3D points while explicitly tolerating outliers, which can be formulated as the nonlinear optimization problem:

$$(6.4) \quad \min_{\|q\|=1} \sum_{i=1}^N \min \left\{ \frac{\|\tilde{z}_i - q \circ \tilde{w}_i \circ q^{-1}\|^2}{\beta_i^2}, 1 \right\},$$

where q is the unit quaternion parametrization of a 3D rotation, $(z_i \in \mathbb{R}^3, w_i \in \mathbb{R}^3)_{i=1}^N$ are given N pairs of 3D points, $\tilde{z} := [z^\top, 0]^\top \in \mathbb{R}^4$, $\tilde{w} := [w^\top, 0]^\top \in \mathbb{R}^4$, $q^{-1} := [-q_1, -q_2, -q_3, q_4]^\top$ is the inverse quaternion, “ \circ ” denotes the quaternion product, $\beta_i > 0$ is a given threshold that determines the maximum inlier residual. Problem (6.4) is a fundamental problem in aerospace, robotics and computer vision [43, 46]. By introducing N binary variables $\{\theta_i\}_{i=1}^N$, Problem (6.4) can be equivalently refor-

mulated as a polynomial optimization problem:

$$(RRS) \quad \min_{\substack{\|q\|=1, \\ \theta_i \in \{-1,1\}, i=1,\dots,N}} \sum_{i=1}^N \frac{1 + \theta_i \|\tilde{z}_i - q \circ \tilde{w}_i \circ q^{-1}\|^2}{2\beta_i^2} + \frac{1 - \theta_i}{2}.$$

Each θ_i is used to decide whether the i -th pair of 3D points (z_i, w_i) is an inlier or an outlier.

Yang and Carlone [43] proposed an SDP relaxation for (RRS) that was empirically shown to be tight. Let $\mathbf{x} = [q^\top, \theta_1, \dots, \theta_N]^\top \in \mathbb{R}^{N+4}$ be the decision variables of (RRS), and let

$$(6.5) \quad v(\mathbf{x}) = [q^\top, \theta_1 q^\top, \dots, \theta_N q^\top]^\top \in \mathbb{R}^{4N+4}$$

be the sparse set of monomials in \mathbf{x} of degree up to two. We build $M = v(\mathbf{x})v(\mathbf{x})^\top$ as the sparse moment matrix. Then the objective function of (RRS) can be linearly expressed in terms of the entries of M . There are linear relationships among the entries of M : (1) the diagonal 4×4 blocks of M are all identical ($\theta_i^2 qq^\top = qq^\top$); (2) the off-diagonal 4×4 blocks are symmetric ($\theta_i \theta_j qq^\top \in \mathbb{S}_4$). Let $\mathcal{A}(X) = b$ collect all independent linear constraints obtained from these linear relationships when relaxing M to an unknown PSD matrix X . In addition, because of the unit quaternion constraint, M satisfies $\text{Tr}(M) = N + 1$ and so we let $\mathcal{B}(X) = d$ impose the trace constraint on X . Consequently, this leads to an SDP relaxation of size

$$(6.6) \quad n = 4N + 4, \quad m = 3N^2 + 13N + 1.$$

For each $N \in \{50, 100, 150, 200, 300, 500\}$, we generate three random instances of (RRS). For each instance, we solve the above SDP relaxation using the solvers MOSEK, SDPLR, SDPNAL+, STRIDE and ManiSDP, respectively. The results are presented in Table 7. The following conclusions can be drawn from the table. (i) MOSEK can solve small-scale instances ($N \leq 100$) to high accuracy, but the running time significantly grows as N increases (< 20 s for $N = 10$ while ~ 600 s for $N = 100$). When $N \geq 150$, MOSEK runs out of space due to large memory consumption. (ii) Both SDPLR and SDPNAL+ fail in solving these SDPs to even medium accuracy. (iii) Both STRIDE and ManiSDP can solve all instances to high accuracy while ManiSDP is faster than STRIDE by a factor of $3 \sim 10$.

In Figures 5 and 6, the factorization size and the maximal KKT residue per iteration in solving a random instance of (RRS) are displayed for $N = 50, 100, 150, 200, 300, 500$, respectively.

6.6. Nearest structured rank deficient matrices. Let us consider the problem of finding the nearest structured rank deficient matrix:

$$(6.7) \quad \min_{u \in \mathbb{R}^N} \left\{ \|u - \theta\|^2 \left| L_0 + \sum_{i=1}^N u_i L_i \text{ is rank deficient} \right. \right\},$$

where $L_i \in \mathbb{R}^{s \times t}$ ($s \leq t$), $i = 0, \dots, N$ and $\theta \in \mathbb{R}^N$ are given. Applications of Problem (6.7) (also known as the structured total least squares problem) could be found in [27]. We can reformulate (6.7) as the following polynomial optimization problem:

$$(NSRD) \quad \min_{z \in \mathbb{R}^s, u \in \mathbb{R}^N} \left\{ \|u - \theta\|^2 \left| z^\top \left(L_0 + \sum_{i=1}^N u_i L_i \right) = 0, \|z\| = 1 \right. \right\}.$$

TABLE 7
Results for the robust rotation search problem.

N	trial	MOSEK 10.0		SDPLR 1.03		SDPNAL+		STRIDE		ManiSDP	
		η_{\max}	time	η_{\max}	time	η_{\max}	time	η_{\max}	time	η_{\max}	time
50	#1	4.7e-10	16.4	9.8e-03	12.5	1.1e-02	106	2.8e-09	18.3	6.6e-09	3.02
	#2	7.9e-10	19.3	3.1e-02	22.0	1.0e-02	96.3	7.3e-09	15.4	7.2e-10	2.93
	#3	1.1e-10	15.2	2.8e-03	19.4	1.1e-02	119	9.5e-09	15.4	5.5e-10	3.59
100	#1	2.0e-11	622	3.6e-04	106	7.1e-02	642	3.1e-09	73.0	1.0e-09	22.9
	#2	1.8e-10	653	8.1e-04	78.1	3.8e-02	631	1.6e-09	67.4	3.6e-10	20.3
	#3	7.3e-12	590	2.9e-03	67.2	7.8e-02	597	4.8e-09	69.0	5.4e-10	18.2
150	#1	-	-	2.0e-03	291	8.0e-02	1691	4.3e-11	249	1.6e-09	33.5
	#2	-	-	1.2e-03	233	6.4e-02	804	7.9e-09	171	4.7e-09	36.5
	#3	-	-	1.5e-01	416	1.2e-01	1491	9.9e-09	162	2.6e-09	33.8
200	#1	-	-	3.1e-02	459	8.3e-02	2799	1.4e-09	254	6.3e-10	65.3
	#2	-	-	1.6e-01	761	6.5e-02	1653	2.9e-09	306	9.2e-10	66.2
	#3	-	-	3.8e-03	894	6.3e-02	2171	3.2e-11	220	8.5e-10	67.9
300	#1	-	-	1.1e-03	1264	5.2e-02	3528	4.1e-10	1176	1.1e-09	188
	#2	-	-	7.3e-03	1787	4.9e-02	3421	8.0e-09	1458	3.6e-09	190
	#3	-	-	2.2e-03	1734	6.0e-02	4260	2.9e-09	868	1.2e-09	203
500	#1	-	-	*	*	*	*	7.1e-09	5627	5.2e-10	601
	#2	-	-	5.4e-02	9574	*	*	4.5e-10	4884	1.9e-09	801
	#3	-	-	*	*	*	*	3.4e-09	7878	5.0e-09	1055

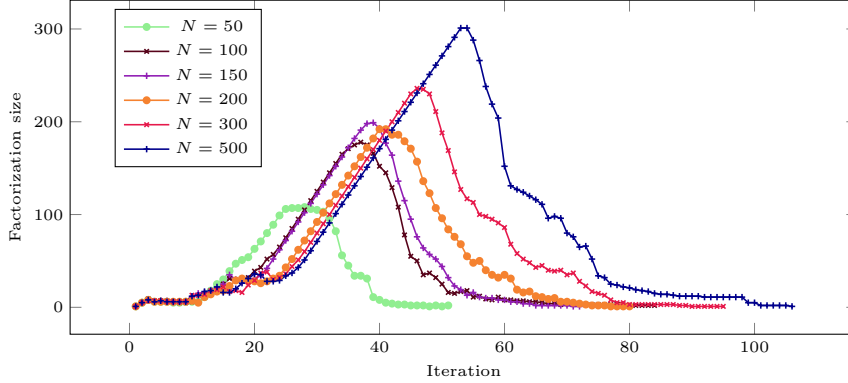


FIG. 5. The factorization size per iteration in solving (RRS).

Note that the unit vector z in (NSRD) serves as a witness of rank deficiency. (NSRD) is non-convex and Cifuentes proposed an SDP relaxation for (NSRD) [14] which is guaranteed to be tight under a low-noise assumption [16]. Let $\mathbf{x} = [z^\top, u^\top]^\top \in \mathbb{R}^{s+N}$ be the vector of variables involved in (NSRD), and let

$$(6.8) \quad v(\mathbf{x}) = [z^\top, u_1 z^\top, \dots, u_N z^\top]^\top \in \mathbb{R}^{s(N+1)}$$

be the sparse set of monomials in \mathbf{x} of degree up to two. We build $M = v(\mathbf{x})v(\mathbf{x})^\top$ as the sparse moment matrix. Then the objective function of (NSRD) can be linearly expressed in terms of the entries of M . There are linear relationships among the entries of M : (1) all off-diagonal $s \times s$ blocks $u_i u_j z z^\top$ are symmetric; (2) each of

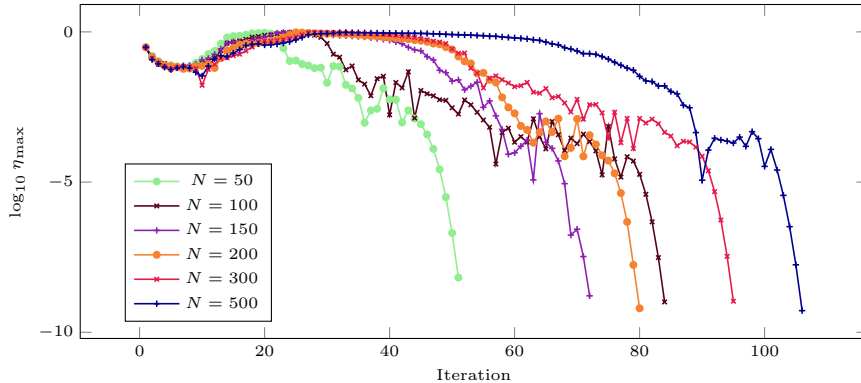


FIG. 6. The maximal KKT residue per iteration in solving (RRS).

the first t equality constraint in (NSRD), say $g = 0$, gives rise to $wg = 0$ for each monomial $w \in v(\mathbf{x})$; (3) the unit norm of z implies that the trace of the leading $s \times s$ block of M is equal to 1. Let $\mathcal{A}(X) = b$ collect all independent linear constraints obtained from these linear relationships when relaxing M to an unknown PSD matrix X . Consequently, we obtain an SDP relaxation of size

$$n = s(N + 1), \quad m = 1 + st(N + 1) + \frac{s(s - 1)N(N + 1)}{4}.$$

For each $s \in \{10, 15, 20, 25, 30, 40\}$, we generate three random instances of (6.7) with $s = t$ and $N = 2s - 1$. For each instance, we solve the above SDP relaxation using the solvers MOSEK, SDPLR, SDPNAL+, STRIDE and ManiSDP, respectively. The results are presented in Table 8 from which we can make the following conclusions. (i) MOSEK can solve small-scale instances ($s \leq 15$) to high accuracy, but the running time significantly grows as s increases (~ 20 s for $s = 10$ while ~ 1500 s for $s = 15$). When $s \geq 20$, MOSEK runs out of space due to large memory consumption. (ii) SDPLR can only solve small-scale instances to medium accuracy, and becomes unreliable when $s \geq 20$ for returning numerical errors. (iii) SDPNAL+ is much slower than STRIDE and ManiSDP, and can only obtain low/medium accuracy solutions. (iv) Both STRIDE and ManiSDP can solve the instances to high accuracy (occasionally STRIDE returns low/medium accuracy solutions) while ManiSDP is faster than STRIDE by a factor of $2 \sim 10$.

In Figures 7 and 8, the factorization size and the maximal KKT residue per iteration in solving a random instance of (NSRD) are shown for $s = 10, 15, 20, 25, 30, 40$, respectively.

7. Conclusions. We have presented a manifold optimization approach to solve linear SDPs with low-rank solutions by integrating the ALM and the Burer-Monteiro factorization. Global convergence is guaranteed despite of the non-convexity brought by the Burer-Monteiro factorization. Diverse numerical experiments demonstrate the superior performance of this approach. It has been shown that our solver ManiSDP is able to solve linear SDPs with millions of equality constraints to high accuracy in reasonable time.

We emphasize that ManiSDP is still at an early stage of development and the strength of the approach has not been fully revealed yet. Among others, we list several directions in enhancing the approach: (1) warm-starting; (2) using Riemannian line

TABLE 8
Results for nearest structured rank deficient matrices.

s	trial	MOSEK 10.0		SDPLR 1.03		SDPNAL+		STRIDE		ManiSDP	
		η_{\max}	time	η_{\max}	time	η_{\max}	time	η_{\max}	time	η_{\max}	time
10	#1	3.0e-11	22.9	8.4e-07	6.49	7.2e-08	64.1	3.5e-12	8.97	6.8e-10	1.28
	#2	4.2e-11	20.1	6.4e-05	3.04	1.8e-06	32.5	3.4e-12	4.74	4.4e-10	1.29
	#3	4.2e-09	15.3	6.1e-06	3.87	2.6e-05	15.5	1.3e-10	6.14	4.7e-09	0.90
15	#1	4.9e-11	1623	1.5e-05	236	4.1e-06	233	4.4e-11	41.4	7.1e-09	12.7
	#2	3.5e-09	1436	5.0e-05	369	2.9e-03	256	1.5e-10	33.0	7.2e-09	14.5
	#3	4.6e-10	1558	1.1e-05	32.5	1.8e-06	151	6.2e-11	35.5	6.5e-10	5.97
20	#1	-	-	**	**	3.8e-03	894	3.0e-10	174	9.7e-09	55.9
	#2	-	-	**	**	1.6e-02	1336	3.1e-11	125	7.9e-09	37.5
	#3	-	-	8.6e-06	1055	4.4e-03	1474	2.2e-10	149	7.5e-09	40.1
25	#1	-	-	**	**	6.1e-03	8457	3.3e-06	4398	7.4e-09	781
	#2	-	-	**	**	4.4e-07	3907	5.8e-10	429	7.9e-09	50.8
	#3	-	-	**	**	1.3e-01	5153	2.6e-10	445	6.3e-09	75.5
30	#1	-	-	*	*	*	*	4.2e-10	1812	4.9e-09	697
	#2	-	-	*	*	*	*	4.2e-01	2484	7.2e-09	263
	#3	-	-	*	*	3.8e-07	9616	3.0e-11	1042	3.9e-09	108
40	#1	-	-	*	*	*	*	*	*	4.4e-09	1984
	#2	-	-	*	*	*	*	*	*	4.0e-09	2493
	#3	-	-	*	*	*	*	*	*	3.3e-09	1279

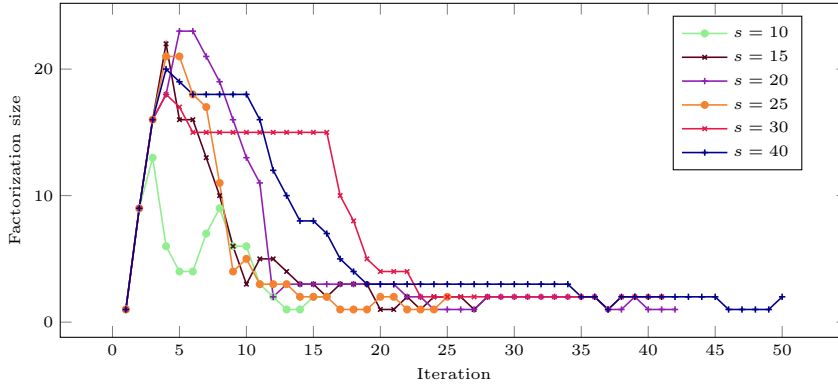


FIG. 7. The factorization size per iteration in solving (NSRD).

search to determine the step size for escaping from saddle points; (3) preconditioning for the Riemannian Hessian; (4) handling SDPs with inequality constraints. Moreover, as SDPs may contain multiple PSD blocks (e.g., SDP relaxations for sparse polynomial optimization problems [37, 38, 39]), it is also worth extending ManiSDP to handle multi-block SDPs. We believe that all of these efforts will eventually lead to a more powerful SDP solver, which makes large-scale low-rank SDPs even more tractable and hence allows to tackle hard application problems in real world.

Acknowledgments. The authors would like to thank Heng Yang for kindly providing the scripts for running STRIDE and for generating random instances of the robust rotation search problem and the problem of nearest structured rank deficient

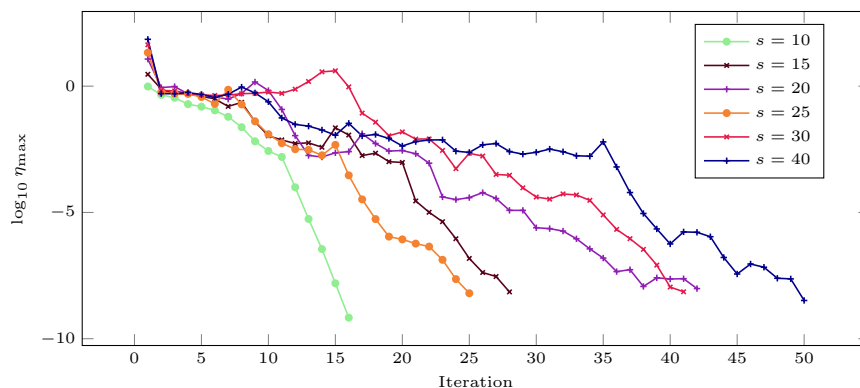


FIG. 8. The maximal KKT residue per iteration in solving (NSRD).

matrices.

REFERENCES

- [1] P.-A. ABSIL, C. G. BAKER, AND K. A. GALLIVAN, *Trust-region methods on Riemannian manifolds*, Found. Comput. Math., 7 (2007), pp. 303–330, <https://doi.org/10.1007/s10208-005-0179-9>.
- [2] P. A. ABSIL, R. MAHONY, AND J. TRUMPF, *An extrinsic look at the Riemannian Hessian*, in Geometric Science of Information: First International Conference, GSI 2013, Paris, France, August 28–30, 2013. Proceedings, Springer, 2013, pp. 361–368.
- [3] E. D. ANDERSEN, C. ROOS, AND T. TERLAKY, *On implementing a primal-dual interior-point method for conic quadratic optimization*, Mathematical Programming, 95 (2003), pp. 249–277.
- [4] X. BAI, H. WEI, K. FUJISAWA, AND Y. WANG, *Semidefinite programming for optimal power flow problems*, International Journal of Electrical Power & Energy Systems, 30 (2008), pp. 383–392.
- [5] S. BELLAVIA, J. GONDZIO, AND M. PORCELLI, *A relaxed interior point method for low-rank semidefinite programming problems with applications to matrix completion*, Journal of Scientific Computing, 89 (2021), pp. 1–36.
- [6] N. BOUMAL, B. MISHRA, P.-A. ABSIL, AND R. SEPULCHRE, *Manopt, a Matlab toolbox for optimization on manifolds*, Journal of Machine Learning Research, 15 (2014), pp. 1455–1459, <https://www.manopt.org>.
- [7] N. BOUMAL, V. VORONINSKI, AND A. BANDEIRA, *The non-convex burer-monteiro approach works on smooth semidefinite programs*, Advances in Neural Information Processing Systems, 29 (2016).
- [8] N. BOUMAL, V. VORONINSKI, AND A. S. BANDEIRA, *Deterministic guarantees for burer-monteiro factorizations of smooth semidefinite programs*, Communications on Pure and Applied Mathematics, 73 (2020), pp. 581–608.
- [9] S. BURER AND R. D. MONTEIRO, *A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization*, Mathematical Programming, 95 (2003), pp. 329–357.
- [10] S. BURER AND R. D. MONTEIRO, *Local minima and convergence in low-rank semidefinite programming*, Mathematical programming, 103 (2005), pp. 427–444.
- [11] E. CANDÈS AND B. RECHT, *Exact matrix completion via convex optimization*, Communications of the ACM, 55 (2012), pp. 111–119.
- [12] E. J. CANDÈS AND T. TAO, *The power of convex relaxation: Near-optimal matrix completion*, IEEE Transactions on Information Theory, 56 (2010), pp. 2053–2080.
- [13] D. CAVALCANTI AND P. SKRZYPCZYK, *Quantum steering: a review with focus on semidefinite programming*, Reports on Progress in Physics, 80 (2016), p. 024001.
- [14] D. CIFUENTES, *A convex relaxation to compute the nearest structured rank deficient matrix*, SIAM Journal on Matrix Analysis and Applications, 42 (2021), pp. 708–729.
- [15] D. CIFUENTES, *On the burer–monteiro method for general semidefinite programs*, Optimization Letters, (2021), pp. 1–11.

- [16] D. CIFUENTES, S. AGARWAL, P. A. PARRILO, AND R. R. THOMAS, *On the local stability of semidefinite relaxations*, *Mathematical Programming*, 193 (2022), pp. 629–663.
- [17] A. DE MAIO, S. DE NICOLA, Y. HUANG, Z.-Q. LUO, AND S. ZHANG, *Design of phase codes for radar performance optimization with a similarity constraint*, *IEEE Transactions on Signal Processing*, 57 (2008), pp. 610–621.
- [18] M. GARSTKA, M. CANNON, AND P. GOULART, *Cosmo: A conic operator splitting method for convex conic problems*, *Journal of Optimization Theory and Applications*, 190 (2021), pp. 779–810.
- [19] M. X. GOEMANS, *Semidefinite programming in combinatorial optimization*, *Mathematical Programming*, 79 (1997), pp. 143–161.
- [20] S. HABIBI, A. KAVAND, M. KOCVARA, AND M. STINGL, *Barrier and penalty methods for low-rank semidefinite programming with application to truss topology design*, arXiv preprint arXiv:2105.08529, (2021).
- [21] C. HELMBERG AND F. RENDL, *A spectral bundle method for semidefinite programming*, *SIAM Journal on Optimization*, 10 (2000), pp. 673–696.
- [22] M. JOURNÉE, F. BACH, P.-A. ABSIL, AND R. SEPULCHRE, *Low-rank optimization on the cone of positive semidefinite matrices*, *SIAM Journal on Optimization*, 20 (2010), pp. 2327–2351.
- [23] J. B. LASSERRE, *An explicit exact sdp relaxation for nonlinear 0-1 programs*, in *Integer Programming and Combinatorial Optimization: 8th International IPCO Conference Utrecht, The Netherlands, June 13–15, 2001 Proceedings 8*, Springer, 2001, pp. 293–303.
- [24] J.-B. LASSERRE, *Global Optimization with Polynomials and the Problem of Moments*, *SIAM Journal on Optimization*, 11 (2001), pp. 796–817.
- [25] C. LIU AND N. BOUMAL, *Simple algorithms for optimization on riemannian manifolds with constraints*, *Applied Mathematics & Optimization*, 82 (2020), pp. 949–981.
- [26] Z.-Q. LUO, W.-K. MA, A. M.-C. SO, Y. YE, AND S. ZHANG, *Semidefinite relaxation of quadratic optimization problems*, *IEEE Signal Processing Magazine*, 27 (2010), pp. 20–34.
- [27] I. MARKOVSKY, *Structured low-rank approximation and its applications*, *Automatica*, 44 (2008), pp. 891–909.
- [28] P. A. PARRILO AND S. LALL, *Semidefinite programming relaxations and algebraic optimization in control*, *European Journal of Control*, 9 (2003), pp. 307–321.
- [29] C. PEI LEE, L. LIANG, T. TANG, AND K.-C. TOH, *Accelerating nuclear-norm regularized low-rank matrix optimization through Burer-Monteiro decomposition*, 2022, <https://arxiv.org/abs/2204.14067>. arXiv:2204.14067v2.
- [30] D. M. ROSEN, *Scalable low-rank semidefinite programming for certifiably correct machine perception*, in *Algorithmic Foundations of Robotics XIV: Proceedings of the Fourteenth Workshop on the Algorithmic Foundations of Robotics 14*, Springer International Publishing, 2021, pp. 551–566.
- [31] D. M. ROSEN, L. CARLONE, A. S. BANDEIRA, AND J. J. LEONARD, *Se-sync: A certifiably correct algorithm for synchronization over the special euclidean group*, *The International Journal of Robotics Research*, 38 (2019), pp. 95–125.
- [32] M. SOUTO, J. D. GARCIA, AND Á. VEIGA, *Exploiting low-rank structure in semidefinite programming by approximate operator splitting*, *Optimization*, 71 (2022), pp. 117–144.
- [33] D. SUN, K.-C. TOH, Y. YUAN, AND X.-Y. ZHAO, *SDPNAL+: A Matlab software for semidefinite programming with bound constraints (version 1.0)*, *Optimization Methods and Software*, 35 (2020), pp. 87–115.
- [34] T. TANG AND K.-C. TOH, *Solving graph equipartition SDPs on an algebraic variety*, 2021, <https://arxiv.org/abs/2112.04256>. arXiv:2112.04256v2.
- [35] K.-C. TOH, M. J. TODD, AND R. H. TÜTÜNCÜ, *Sdpt3—a matlab software package for semidefinite programming, version 1.3*, *Optimization methods and software*, 11 (1999), pp. 545–581.
- [36] L. VANDENBERGHE AND S. BOYD, *Semidefinite programming*, *SIAM review*, 38 (1996), pp. 49–95.
- [37] J. WANG, V. MAGRON, AND J.-B. LASSERRE, *Chordal-TSSOS: a moment-SOS hierarchy that exploits term sparsity with chordal extension*, *SIAM Journal on Optimization*, 31 (2021), pp. 114–141.
- [38] J. WANG, V. MAGRON, AND J.-B. LASSERRE, *TSSOS: A moment-SOS hierarchy that exploits term sparsity*, *SIAM Journal on Optimization*, 31 (2021), pp. 30–58.
- [39] J. WANG, V. MAGRON, J.-B. LASSERRE, AND N. H. A. MAI, *CS-TSSOS: Correlative and term sparsity for large-scale polynomial optimization*, arXiv:2005.02828, (2020).
- [40] Y. WANG, K. DENG, H. LIU, AND Z. WEN, *A decomposition augmented lagrangian method for low-rank semidefinite programming*, arXiv preprint arXiv:2109.11707, (2021).
- [41] Z. WEN, D. GOLDFARB, AND W. YIN, *Alternating direction augmented lagrangian methods for*

- semidefinite programming*, Mathematical Programming Computation, 2 (2010), pp. 203–230.
- [42] H. WOLKOWICZ, R. SAIGAL, AND L. VANDENBERGHE, *Handbook of semidefinite programming: theory, algorithms, and applications*, vol. 27, Springer Science & Business Media, 2012.
- [43] H. YANG AND L. CARLONE, *A quaternion-based certifiably optimal solution to the wahba problem with outliers*, in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 1665–1674.
- [44] H. YANG AND L. CARLONE, *Certifiably optimal outlier-robust geometric perception: Semidefinite relaxations and scalable global optimization*, IEEE Transactions on Pattern Analysis and Machine Intelligence, (2022).
- [45] H. YANG, L. LIANG, L. CARLONE, AND K.-C. TOH, *An inexact projected gradient method with rounding and lifting by nonlinear programming for solving rank-one semidefinite relaxation of polynomial optimization*, Mathematical Programming, (2022), pp. 1–64.
- [46] H. YANG, J. SHI, AND L. CARLONE, *Teaser: Fast and certifiable point cloud registration*, IEEE Transactions on Robotics, 37 (2020), pp. 314–333.
- [47] L. YANG, D. SUN, AND K.-C. TOH, *SDPNAL+: a majorized semismooth Newton-CG augmented lagrangian method for semidefinite programming with nonnegative constraints*, Mathematical Programming Computation, 7 (2015), pp. 331–366.
- [48] A. YURTSEVER, J. A. TROPP, O. FERCOQ, M. UDELL, AND V. CEVHER, *Scalable semidefinite programming*, SIAM Journal on Mathematics of Data Science, 3 (2021), pp. 171–200.
- [49] R. Y. ZHANG AND J. LAVAEI, *Modified interior-point method for large-and-sparse low-rank semidefinite programs*, in 2017 IEEE 56th Annual Conference on Decision and Control (CDC), IEEE, 2017, pp. 5640–5647.
- [50] R. Y. ZHANG AND J. LAVAEI, *Sparse semidefinite programs with guaranteed near-linear time complexity via dualized clique tree conversion*, Mathematical programming, 188 (2021), pp. 351–393.
- [51] Y. ZHENG, G. FANTUZZI, A. PAPACHRISTODOULOU, P. GOULART, AND A. WYNN, *Chordal decomposition in operator-splitting methods for sparse semidefinite programs*, Mathematical Programming, 180 (2020), pp. 489–532.
- [52] Y. ZHOU, C. BAO, C. DING, AND J. ZHU, *A semismooth newton based augmented lagrangian method for nonsmooth optimization on matrix manifolds*, Mathematical Programming, (2022), pp. 1–61.