Solving Low-Rank Semidefinite Programs via Manifold Optimization

Jie Wang[1] · **Liangbing Hu**[2]

Abstract

We propose a manifold optimization approach to solve linear semidefinite programs (SDP) with low-rank solutions, with an emphasis on SDP relaxations for polynomial optimization problems. This approach incorporates the inexact augmented Lagrangian method (ALM) and the Burer-Monteiro factorization, and features the self-adaptive strategies for updating the factorization size and the penalty parameter. We establish global convergence of the inexact ALM, despite the non-convexity brought by the Burer-Monteiro factorization. We further provide a practical algorithm building on the inexact ALM, and 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.

✉ Jie Wang
   wangjie212@amss.ac.cn

   Liangbing Hu
   huliangbing2000@163.com

1 Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China

2 Nanjing Research Institute of Electronics Technology, Nanjing, China

1 Introduction

In this paper, we aim to efficiently solve the following semidefinite programming (SDP) problem:

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

where $\mathcal{A} : \mathbb{S}_n \to \mathbb{R}^m, \mathcal{B} : \mathbb{S}_n \to \mathbb{R}^l$ 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 $X^\star$, i.e., $\text{rank}(X^\star) \ll n$.

Such (SDP) arises from a diverse set of fields, e.g., systems and control [33], signal processing [18, 30], optimal power flow [4], matrix completion [11], quantum steering [13], computer vision [51], to just name a few; see also the surveys [43, 49] and references therein. It also serves as tractable convex relaxations for many difficult (usually NP-hard) non-convex optimization problems, e.g., quadratically constrained quadratic programs (QCQP) [30], combinatorial optimization problems [20], polynomial optimization problems [27].

In view of the wide applications, a great deal of effort has been dedicated to solving (SDP) over the past decades and many 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, 42]. However, for large-scale SDPs interior-point methods are no longer reliable because of their extensive memory occupation and high computational cost in solving a large and dense 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 [54]. The related solver SDPNAL+ has been shown to have good scalability on *degenerate* SDPs [39]. 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) [48]. 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 [57]. Zheng et al. [58] and Garstka et al. [19] 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) [23]. 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 [55]. The property of admitting low-rank solutions can be exploited in the framework of interior-point methods [5, 21, 56] or in the framework of operator splitting methods [38]. 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 [52], which is scalable on a variety of SDPs emerging from computer vision problems [51].

Another notable approach taking the low-rank property into account is to perform a factorization $X = YY^\mathsf{T}$ 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 liter-

ature [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 for almost all cost matrices C, 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 [36, 41]. See also [22, 24, 32, 40] for very recent developments of this approach.

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 $\mathcal{N}_p = \{Y \in \mathbb{R}^{n \times p} \mid \mathcal{B}(YY^\mathsf{T}) = d\}$ is a smooth manifold [25]. 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 [34] for an extension of this approach to the regularized convex matrix optimization problem and [37] 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 [29]. Later, Zhou et al. developed an ALM for solving a class of manifold optimization problems with nonsmooth objective functions and nonlinear constraints [59]. 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 ALM subproblem [47].

Contributions. Our contributions are as follows.

- We present a manifold optimization approach for solving (SDP) by adopting the Burer-Monteiro factorization and the idea of Journée et al. More concretely, our approach employs an ALM framework to handle the linear constraints $\mathcal{A}(X) = b$ and applies the Burer-Monteiro factorization $X = YY^\mathsf{T}$ to the ALM subproblem in order to exploit the low-rank property. The ALM subproblem is then recast as a manifold optimization problem on $\mathcal{N}_p$ which could be solved with efficient manifold optimization methods. To circumvent the non-convexity introduced by the Burer-Monteiro factorization, we design an effective strategy (inspired by Journée et al.) to escape from saddle points. Then under certain conditions, we establish the global convergence of the proposed inexact ALM.
- To further enhance the practical performance of the approach, we propose self-adaptive strategies for updating the factorization size and the penalty parameter. Specifically, (i) we dynamically adjust the factorization size p so that the decision variables of the ALM subproblem is as few as possible; (ii) we propose a *simple* strategy to self-adaptively increase or decrease the penalty parameter so that it will not become too large (note that the ALM subproblem with a large penalty parameter is more difficult to solve).
- We provide a practical algorithm building on the inexact ALM, where the ALM subproblem is solved only to gain some descent (without optimality requirements) and the procedure of escaping from saddle points is performed only once within each outer iteration of the ALM. Surprisingly and intriguingly, global convergence still occurs in all numerical examples examined in this paper.
- As another main contribution, we release an open-source low-rank 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.

Although both incorporate the ALM framework, the Burer-Monteiro factorization and manifold optimization, this work differs from [47] in fourfold: (1) This work targets at solving large-scale linear SDPs, especially those arising from *the moment-SOS hierarchy* of polynomial optimization problems while [47] primarily aims to solve nonlinear and nonsmooth SDPs. Accordingly, this work adopts the Riemannian trust-region method to solve the ALM subproblem while [47] adopts a Riemannian semismooth Newton method. (2) This work introduces a *single-step* strategy for escaping from saddle points within each ALM outer iteration, yielding a practical algorithm with superior performance. (3) This work distinguishes itself from conventional ALM implementations and [47] through a self-adaptive strategy that enables reduction of the penalty parameter and improves the performance of the algorithm a lot. (4) Along with an explicit algorithm, this work releases an open-source low-rank SDP solver.

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, discuss how to escape from saddle points, and prove global convergence of the inexact ALM. In Section 5, we describe the strategies of updating the factorization size and the penalty parameter, and present the practical algorithm. Results of numerical experiments are provided in Section 6. Conclusions are made in Section 7.

2 Notation and Preliminaries

$\mathbb{R}$ (resp. $\mathbb{R}^+$, $\mathbb{N}$) denotes the set of real numbers (resp. positive real numbers, nonnegative integers). For a positive integer n, let $[n] := \{1, 2, \ldots, n\}$. Let $\mathbb{S}_n$ (resp. $\mathbb{S}_n^+$) denote the set of (resp. positive semidefinite/PSD) symmetric matrices of size n. We use $\mathrm{Tr}(A)$ (resp. A^T) to denote the trace (resp. transpose) of a matrix $A \in \mathbb{R}^{n \times m}$. For two matrices $A, B \in \mathbb{R}^{n \times m}$, the inner product is defined as $\langle A, B \rangle = \mathrm{Tr}(A^\mathsf{T} B)$. For $A \in \mathbb{R}^{n \times n}$, $\mathrm{diag}(A)$ denotes the diagonal of A, $\mathrm{Diag}(A)$ denotes the diagonal matrix with the same diagonal as A, and $\lambda_{\min}(A)$, $\lambda_{\max}(A)$ denote the smallest, largest eigenvalues of A, respectively. For a vector v, $\|v\|$ is the 2-norm of v and for a matrix A, $\|A\|$ is the Frobenius norm of A. For a function $f(x)$, we write $\nabla f(x)$ (resp. $\mathrm{grad}\, f(x)$) for the Euclidean (resp. Riemannian) gradient, and write $\nabla^2 f(x)[u]$ (resp. $\mathrm{Hess}\, f(x)[u]$) 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 $\mathcal{A}^* : \mathbb{R}^m \to \mathbb{S}_n$ 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 $\mathcal{B}^* : \mathbb{R}^l \to \mathbb{S}_n$ 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*

$$\mathcal{A}(X) = b, \tag{2.1a}$$

$$\mathcal{B}(X) = d, \tag{2.1b}$$

$$X \succeq 0, \tag{2.1c}$$

$$S := C - \mathcal{A}^*(y) - \mathcal{B}^*(z) \succeq 0, \tag{2.1d}$$

$$XS = 0. \tag{2.1e}$$

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

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). Recall that the constraints $\mathcal{B}(X) = d$ in (SDP) define a certain manifold structure (which will be rigorously defined later). Let us denote

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

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

$$\mathcal{M} = \{X \in \mathbb{R}^{n \times n} \mid X \succeq 0, \mathrm{Tr}(X) = 1\}, \tag{Unit-trace}$$

$$\mathcal{M} = \{X \in \mathbb{R}^{n \times n} \mid X \succeq 0, \mathrm{diag}(X) = \mathbf{1}\}. \tag{Unit-diagonal}$$

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

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

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.,

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

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

$$\mathcal{N}_p = \{Y \in \mathbb{R}^{n \times p}\}, \tag{Euclidean}$$

$$\mathcal{N}_p = \{Y \in \mathbb{R}^{n \times p} \mid \|Y\| = 1\}, \tag{Sphere}$$

$$\mathcal{N}_p = \{Y \in \mathbb{R}^{n \times p} \mid \|Y(i, :)\| = 1, i = 1, \ldots, n\}, \tag{Oblique}$$

where $Y(i, :)$ 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 $i = 1, \ldots, l$.

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 Assumptions 3.4–3.5 are not essential and can be weakened to the assumption that the matrices $\{B_i Y\}_{i=1}^l$ are linearly independent in $\mathbb{R}^{n \times p}$ for all $Y \in \mathcal{N}_p$; we refer the reader to [8] for detailed discussions. In this paper, we utilize assumptions 3.4–3.5 to obtain a simple closed form of the dual variables z associated to the manifold constraints.

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

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

As for usual constrained optimization problems, (SDP-M) can be solved within an inexact ALM which is presented in Algorithm 3.1 (cf. [47]).

Algorithm 3.1 Inexact ALM template

Input: $\mathcal{A}, b, \mathcal{B}, d, C, \sigma_0 > 0, \sigma_{\min} > 0, \sigma_{\max} > 0$
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) inexactly to obtain an approximate minimizer X^{k+1};
4:   $y^{k+1} \leftarrow y^k - \sigma_k(\mathcal{A}(X^{k+1}) - b)$;
5:   $S^{k+1} \leftarrow \nabla\Phi_k(X^{k+1}) - \mathcal{B}^*(z)$, where z is given by (3.7) with $X := X^{k+1}$;
6:   Determine $\sigma_{k+1} \in [\sigma_{\min}, \sigma_{\max}]$ according to some policy;
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

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

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*

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

$$XS = 0, \tag{3.6b}$$

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:

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

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

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

$$X \succeq 0, \tag{3.8b}$$

$$S = \nabla\Phi_k(X) - \mathcal{B}^*(z) \succeq 0, \tag{3.8c}$$

$$XS = X(\nabla\Phi_k(X) - \mathcal{B}^*(z)) = 0. \tag{3.8d}$$

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

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

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

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

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

$$\mathrm{Tr}(B_i^2 X) = \mathrm{Tr}(Y^\mathsf{T} B_i^2 Y) = \|B_i Y\|^2 \neq 0,$$

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 the trace of (3.10). $\qquad\square$

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

$$z = \mathrm{Tr}(\nabla\Phi_k(X)X), \tag{3.11}$$

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

$$z = \mathrm{diag}(\nabla\Phi_k(X)X). \tag{3.12}$$

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 $\Psi_k(Y) = \Phi_k(YY^\mathsf{T})$ for $Y \in \mathbb{R}^{n \times p}$. Consequently, the convex subproblem (3.5) becomes the non-convex factorized subproblem:

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

Then, we solve the non-convex factorized subproblem (ALMS-k) on the manifold $\mathcal{N}_p$ (Assumption 3.3) instead of solving the convex subproblem (3.5) on $\mathcal{M}$.

4 A Manifold Optimization Approach

Since $\mathcal{N}_p$ is assumed to be a smooth manifold, we can solve the non-convex factorized subproblem (ALMS-k) by off-the-shelf efficient manifold optimization methods. Here, we choose the Riemannian trust-region method. We elaborate on the rationale behind this choice as follows. Existing work [17, 28] has demonstrated that the inexact ALM for solving linear SDPs can achieve linear convergence, provided that the ALM subproblem is solved with sufficient accuracy. This finding indicates that to retain the fast linear convergence of the inexact ALM and to obtain optimal solutions of high accuracy, it is more preferable to employ a second-order method for solving the ALM subproblem. Among second-order methods for

smooth optimization[1], we adopt the Riemannian trust-region method as it not only guarantees global convergence but also exhibits a superlinear (or even quadratic) local convergence rate [1], thereby combining robustness with exceptional computational efficiency in ALM implementations.

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

$$T_Y \mathcal{N}_p = \{U \in \mathbb{R}^{n \times p} \mid \mathcal{B}(UY^\mathsf{T}) = 0\}, \tag{4.1}$$

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

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

Lemma 4.1 *Let Assumption 3.4 hold. Let Y be a point on $\mathcal{N}_p$. The orthogonal projector $P_Y : \mathbb{R}^{n \times p} \to T_Y \mathcal{N}_p$ is given by*

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

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

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

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 the 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

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

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

Proposition 4.2 *(cf. [47], Proposition 2.3) Consider the non-convex factorized subproblem (ALMS-k). Let $X = YY^\mathsf{T}$, $\widetilde{S} = \nabla \Phi_k(X)$ and $S = \widetilde{S} - \mathcal{B}^*(z)$ with z being given in (3.7). Then the* Riemannian gradient *at Y is given by*

$$\mathrm{grad}\, \Psi_k(Y) = 2SY. \tag{4.5}$$

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

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

---

where

$$u_i = \frac{Tr(B_i^2 U Y^\mathsf{T}) z_i}{Tr(B_i^2 X)}, \text{ for } i = 1, \ldots, l. \tag{4.7}$$

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

$$Hess\, \Psi_k(Y)[U] = \widetilde{H}; \tag{4.8}$$

for $\mathcal{N}_p$ in ([Sphere](#)), we have

$$Hess\, \Psi_k(Y)[U] = \widetilde{H} - Tr(\widetilde{H} Y^\mathsf{T}) Y - 2Tr(\widetilde{S} X) U; \tag{4.9}$$

for $\mathcal{N}_p$ in ([Oblique](#)), we have

$$Hess\, \Psi_k(Y)[U] = \widetilde{H} - Diag(\widetilde{H} Y^\mathsf{T}) Y - 2Diag(\widetilde{S} X) U. \tag{4.10}$$

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

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

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

$$\mathcal{B}(\text{grad}\, \Psi_k(Y) Y^\mathsf{T}) = 2\mathcal{B}((\widetilde{S} - \mathcal{B}^*(z)) X) = 0. \tag{4.11}$$

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

By (10) of [2], it holds

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

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. □

The global optimality condition of ([ALMS-k](#)) can be characterized in terms of positive semidefiniteness of the matrix S (cf. [25, Theorem 4]).

Proposition 4.3 *Let $Y \in \mathcal{N}_p$, $X = YY^\mathsf{T} \in \mathcal{M}$, and z be given by (3.7). Let $S = \nabla \Phi_k(X) - \mathcal{B}^*(z)$. Then a stationary point Y of the non-convex factorized subproblem ([ALMS-k](#)) is a global minimizer if and only if $S \succeq 0$.*

Proof Note that Y is a global minimizer of ([ALMS-k](#)) if and only if X is a minimizer of the convex subproblem (3.5). The fact that Y is a stationary point implies grad $\Psi_k(Y) = 2SY = 0$ and, consequently, $XS = 0$. The conclusion then follows from Lemma 3.7. □

As a corollary of Proposition 4.3, we obtain the following theorem.

Theorem 4.4 *A stationary point $Y \in \mathcal{N}_p$ of the non-convex factorized subproblem (ALMS-k) provides a minimizer $X = YY^\top$ 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 This is immediate from Proposition 4.3.   □

4.2 Escaping from Saddle Points

Since the subproblem (ALMS-k) is non-convex, in order to solve (ALMS-k) to certain 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 $S = \nabla \Phi_k(X) - \mathcal{B}^*(z) \not\succeq 0$.

Lemma 4.5 *For any $U \in \mathbb{R}^{n \times p}$ satisfying $YU^\top = 0$, it holds*

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

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

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

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

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

□

Theorem 4.6 *Suppose $S = \nabla \Phi_k(X) - \mathcal{B}^*(z) \not\succeq 0$ 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 (ALMS-k) with $p := p + \delta$ at the point $Y := [Y, 0_{n \times \delta}]$, namely, U satisfies*

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

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

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

By Lemma 4.5,

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

□

Remark 4.7 The fact that an eigenvector corresponding to a negative eigenvalue of S yields a second-order descent direction of (ALMS-k) was first observed in [25] where only one eigenvector corresponding to the smallest eigenvalue was used. In [47], the authors used eigenvectors corresponding to all negative eigenvalues to obtain a descent direction.

The following theorem adapted from [25] (see also [47] 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 (ALMS-k).

Theorem 4.8 *([25], Theorem 7) A second-order critical point $Y \in \mathcal{N}_p$ of the non-convex factorized subproblem (ALMS-k) provides a minimizer $X = YY^\mathsf{T}$ of the convex subproblem (3.5) if it is rank deficient, i.e., rank $Y < p$.*

Proof Let rank $Y = r < p$. We can write $Y = Y'P^\mathsf{T}$ 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 $P^\mathsf{T}P_\perp = 0$. Let $V \in \mathbb{R}^{n \times (p-r)}$ be an arbitrary matrix and let $U = VP_\perp^\mathsf{T}$. We have $UY^\mathsf{T} = VP_\perp^\mathsf{T}P(Y')^\mathsf{T} = 0$. As Y is a second-order critical point of the non-convex factorized subproblem (ALMS-k), Hess $\Psi_k(Y)$ is positive semidefinite. Then by Lemma 4.5, we have

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

The arbitrariness of V implies that S is positive semidefinite and so by Proposition 4.3, $Y \in \mathcal{N}_p$ is a global minimizer of (ALMS-k), which further implies that $X = YY^\mathsf{T}$ is a minimizer of (3.5). □

4.3 Global Convergence

We now establish the global convergence of the inexact ALM for solving (SDP-M) assuming that the ALM subproblem (ALMS-k) is solved to certain optimality.

Theorem 4.9 *Let $\{\varepsilon_k\}_{k \in \mathbb{N}}, \{\tau_k\}_{k \in \mathbb{N}} \subseteq \mathbb{R}^+$ satisfy $\sum_{i=0}^\infty \varepsilon_k < \infty$ and $\sum_{i=0}^\infty \tau_k < \infty$. Suppose that we use the following stopping criteria for the ALM subproblem (ALMS-k):*

$$\|\text{grad } \Psi_k(Y)\| \leq \varepsilon_k \text{ and } \lambda_{\min}(S) \geq -\tau_k, \tag{4.15}$$

where S is the dual variable to X defined as in Proposition 4.2. Assume that the sequence $\{Y^k\}_{k \in \mathbb{N}}$ is bounded. Let $(\hat{X}, \hat{y}, \hat{S})$ be a limit point of $\{(X^k, y^k, S^k)\}_{k \geq 1}$. Then $(\hat{X}, \hat{y}, \hat{S})$ is a KKT point of (SDP-M), i.e., it satisfies (2.1) with $\hat{z}$ being given by

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

For the proof, we need the following two lemmas.

Lemma 4.10 *Suppose that Y^{k+1} is an approximate minimizer of the ALM subproblem (ALMS-k) so that the criteria (4.15) is fulfilled. Let $X^{k+1} := Y^{k+1}(Y^{k+1})^\mathsf{T}$ and $(X^\star, y^\star, S^\star)$ be a KKT point of (SDP-M). Then*

$$\langle y^k - y^\star, \mathcal{A}(X^{k+1}) - b \rangle \geq \sigma_k \|\mathcal{A}(X^{k+1}) - b\|^2 - \tau_k Tr(X^\star) - \frac{\varepsilon_k}{2}\|Y^{k+1}\|. \tag{4.16}$$

Proof By Proposition 4.2, we have $S^{k+1}Y^{k+1} = \frac{1}{2} \cdot \text{grad } \Psi_k(Y^{k+1})$ and so it holds $\langle S^{k+1}Y^{k+1}, Y^{k+1} \rangle \leq \frac{1}{2} \cdot \|\text{grad } \Psi_k(Y^{k+1})\| \cdot \|Y^{k+1}\|$. Thus from the stopping criteria (4.15), we obtain

$$\langle S^{k+1}, X^\star - X^{k+1} \rangle = \langle S^{k+1}, X^\star \rangle - \langle S^{k+1}Y^{k+1}, Y^{k+1} \rangle \geq -\tau_k \text{Tr}(X^\star) - \frac{\varepsilon_k}{2}\|Y^{k+1}\|. \tag{4.17}$$

Substituting $C - \mathcal{A}^*(y^k - \sigma_k(\mathcal{A}(X^{k+1}) - b)) - \mathcal{B}^*(z^{k+1})$ for S^{k+1} in (4.17) where z^{k+1} is given by (3.7) with $X := X^{k+1}$, we obtain

$$\langle C - \mathcal{A}^*(y^k - \sigma_k(\mathcal{A}(X^{k+1}) - b)) - \mathcal{B}^*(z^{k+1}), X^\star - X^{k+1} \rangle$$

$$= \langle C, X^\star - X^{k+1} \rangle - \langle y^k - \sigma_k(\mathcal{A}(X^{k+1}) - b), \mathcal{A}(X^\star) - \mathcal{A}(X^{k+1}) \rangle$$

$$\quad - \langle z^{k+1}, \mathcal{B}(X^\star) - \mathcal{B}(X^{k+1}) \rangle$$

$$= \langle C, X^\star - X^{k+1} \rangle + \langle y^k - \sigma_k(\mathcal{A}(X^{k+1}) - b), \mathcal{A}(X^{k+1}) - b \rangle$$

$$\geq -\tau_k \mathrm{Tr}(X^\star) - \frac{\varepsilon_k}{2} \|Y^{k+1}\|, \tag{4.18}$$

where the second equality is because $\mathcal{A}(X^\star) = b$ and $\mathcal{B}(X^\star) = \mathcal{B}(X^{k+1}) = d$. On the other hand, we have

$$\langle S^\star, X^{k+1} - X^\star \rangle = \langle C - \mathcal{A}^*(y^\star) - \mathcal{B}^*(z^\star), X^{k+1} - X^\star \rangle = \langle S^\star, X^{k+1} \rangle \geq 0,$$

which gives

$$\langle C, X^{k+1} - X^\star \rangle - \langle y^\star, \mathcal{A}(X^{k+1}) - b \rangle \geq 0. \tag{4.19}$$

Summing (4.18) and (4.19) gives (4.16) as desired.   □

Lemma 4.11 *Assume that $\{Y^k\}_{k \in \mathbb{N}}$ is bounded. The sequence $\{\mathcal{A}(X^k)\}_{k \geq 1}$ converges to b.*

Proof Let $(X^\star, y^\star, S^\star)$ be a KKT point of (SDP-M). For all $k \geq 1$, by invoking Lemma 4.10, we have

$$\|y^{k+1} - y^\star\|^2 = \|y^k - y^\star\|^2 - 2\sigma_k \langle y^k - y^\star, \mathcal{A}(X^{k+1}) - b \rangle + \sigma_k^2 \|\mathcal{A}(X^{k+1}) - b\|^2$$

$$\leq \|y^k - y^\star\|^2 + \sigma_k(2\tau_k \mathrm{Tr}(X^\star) + \varepsilon_k \|Y^{k+1}\|) - \sigma_k^2 \|\mathcal{A}(X^{k+1}) - b\|^2.$$

For arbitrary $N \geq 1$, summing the above inequality for $k = 1, \ldots, N$, we have

$$0 \leq \sum_{k=1}^{N} \left(\|y^k - y^\star\|^2 - \|y^{k+1} - y^\star\|^2 \right) - \sum_{k=1}^{N} \sigma_k^2 \|\mathcal{A}(X^{k+1}) - b\|^2$$

$$+ \sum_{k=1}^{N} \sigma_k \left(2\tau_k \mathrm{Tr}(X^\star) + \varepsilon_k \|Y^{k+1}\| \right)$$

$$\leq \|y^1 - y^\star\|^2 - \|y^{N+1} - y^\star\|^2 - \sigma_{\min}^2 \sum_{k=1}^{N} \|\mathcal{A}(X^{k+1}) - b\|^2$$

$$+ 2\sigma_{\max} \mathrm{Tr}(X^\star) \sum_{k=1}^{N} \tau_k + \sigma_{\max} D \sum_{k=1}^{N} \varepsilon_k,$$

where $D = \max\{\|Y^{k+1}\| : k \geq 1\}$. It follows

$$\sigma_{\min} \sum_{k=1}^{N} \|\mathcal{A}(X^{k+1}) - b\|^2 \leq \|y^1 - y^\star\|^2 + 2\sigma_{\max} \mathrm{Tr}(X^\star) \sum_{k=1}^{N} \tau_k + \sigma_{\max} D \sum_{k=1}^{N} \varepsilon_k. \tag{4.20}$$

Now because the right-hand side of (4.20) is bounded and N is arbitrary, we see that $\mathcal{A}(X^k)$ must converge to b.   □

Proof of Theorem 4.9. Let $(X^\star, y^\star, S^\star)$ be a KKT point of (SDP-M). Assume $\hat{X} = \lim_{i \to \infty} X^{k_i}$. By Lemma 4.11, $\hat{X}$ is feasible to (SDP-M). Substituting k_i for k in (4.18) and

then letting $i \to \infty$, by virtue of Lemma 4.11 we obtain $\langle C, X^\star - \hat{X} \rangle \geq 0$, which implies $\langle C, \hat{X} \rangle \leq \langle C, X^\star \rangle$. Therefore, $\hat{X}$ is an optimal solution of (SDP-M). Because $\lim_{k \to \infty} \tau_k = 0$, we must have $\hat{S} \succeq 0$. Moreover, noting $\langle S^k, X^k \rangle = |\langle S^k Y^k, Y^k \rangle| \leq \frac{\varepsilon_{k-1}}{2} \cdot \|Y^k\|$, since $\lim_{k \to \infty} \varepsilon_k = 0$ and $\{Y^k\}_{k \in \mathbb{N}}$ is bounded, we have $\langle \hat{X}, \hat{S} \rangle = 0$. It follows that $(\hat{X}, \hat{y}, \hat{S})$ is a KKT point of (SDP-M). □

Remark 4.12 It could be seen from the above proof that Theorem 4.9 remains valid with a fixed penalty parameter σ. However, in the next section we will describe an adaptive strategy for updating the penalty parameter in order to improve the performance of the algorithm.

Remark 4.13 It is clear that if the manifold N_p is bounded (e.g., a sphere or oblique manifold), then $\{Y^k\}$ must be bounded. For the general case, it was established in [35] that the dual iterate by the ALM coincides with the proximal update on the dual function by the proximal point method, from which the boundedness of the primal sequence $\{X^k\}$ (and hence $\{Y^k\}$) can be deduced under mild conditions (see Proposition 2(c) of [28]).

5 The Practical Algorithm

Before giving the practical algorithm, we first elaborate the strategies for adjusting the factorization size p and the penalty parameter σ which are crucial to enhancing the practical performance of the algorithm.

5.1 Dynamically Adjusting the Factorization Size p

The computational complexity of the ALM subproblem (ALMS-k) heavily depends on the factorization size p. To minimize the computational burden, we propose an effective strategy for dynamically adjusting the value of p inspired by [47]. The recipe behind the strategy is based on two key ingredients: (1) increasing p to escape from saddle points as discussed in Section 5.1; (2) decreasing p by estimating ranks so that the computational burden of the ALM subproblem is as low as possible. More specifically, suppose that $\{s_i\}_i$ are the singular values of Y sorted from large to small. Then the rank of Y is estimated by

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

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

$$Y = WDV^\mathsf{T},$$

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

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

where n_{ne} is the number of negative eigenvalues of S and $\delta_{\mathrm{ne}} \in \mathbb{N}$ 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 $U = [0_{n \times r}, v_1, \cdots, v_\delta]$ where $v_1, \ldots, v_\delta$ are the eigenvectors corresponding to the δ smallest eigenvalues of S. To summarize, the size updating strategy operates through dual complementary mechanisms: (1) size reduction via truncated singular value decomposition on the matrix Y (this makes Y be of full rank); (2) size expansion when $S \nsucceq 0$ in order to escape from saddle points. Therefore, when $S \nsucceq 0$, we can compute a descent direction via size expansion; when $S \succeq 0$, the ALM iterations drive $\|\mathcal{A}(X) - b\| \to 0$ and $XS \to 0$ so that the converging point Y^* provides an optimal solution $(X^* = Y^*(Y^*)^\mathsf{T})$ of the original SDP.

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} should be chosen to balance these two aspects.

Remark 5.2 The above strategy for adjusting the factorization size is adapted from [47] with two distinctions: (1) we rely on a different procedure to estimate ranks; (2) we introduce the parameter δ_{ne} to control the maximum increment of the factorization size at each step.

5.2 Self-adaptively Updating the Penalty Parameter σ

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

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

where $\gamma > 1, \tau > 0, \sigma_{\min}, \sigma_{\max} > 0$ are constants. The intuition behind (5.2) is the following: the inequality $\|\mathcal{A}(X^{k+1}) - b\|/(1 + \|b\|) > \tau\|\mathrm{grad}\,\Psi_k(Y^{k+1})\|$ indicates that the progress of feasibility is not satisfactory and hence we increase the penalty parameter by setting $\sigma_{k+1} = \min\{\gamma\sigma_k, \sigma_{\max}\}$; otherwise, the progress of feasibility is satisfactory and we may decrease the penalty parameter by setting $\sigma_{k+1} = \max\{\sigma_k/\gamma, \sigma_{\min}\}$. In doing so, the penalty parameter will not become too large through the iterations of the algorithm. We point out that a large penalty parameter makes the ALM subproblem (ALMS-k) more difficult to solve, and thus preventing the penalty parameter from becoming large would improve the performance of the algorithm.

5.3 The Algorithm

Our practical algorithm is presented below in Algorithm 5.1.

We now make a few remarks on Algorithm 5.1.

- The initial value p_0 of the factorization size p is typically set to 1 or 2. However, for large-scale SDPs, setting a larger p_0 could be more advantageous. The value of γ is typically set to 2. The optimal setting of the other parameters is highly problem-dependent.
- The values of y^0 and/or Y^0 can be provided according to some initial guess of optimal solutions for warm-starting which may further improve the performance of the algorithm.
- At Step 3, the ALM subproblem is inexactly solved by performing a fixed number of iterations with the Riemannian Trust-Region method along the descent direction U computed at Step 7. So it is not guaranteed that the approximate minimizer of the ALM

Algorithm 5.1 ManiSDP

Input: $\mathcal{A}, b, \mathcal{B}, d, C, \sigma_0 > 0, \sigma_{\min} > 0, \sigma_{\max} > 0, p_0 \in \mathbb{N}, \gamma > 1, \tau > 0, \delta_{\text{ne}} \in \mathbb{N}, \theta \in (0, 1)$
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 (ALMS-k) inexactly with U to obtain an approximate minimizer Y^{k+1};
4:     $X^{k+1} \leftarrow Y^{k+1}(Y^{k+1})^\mathsf{T}, y^{k+1} \leftarrow y^k - \sigma_k(\mathcal{A}(X^{k+1}) - b)$;
5:     $z_i \leftarrow \dfrac{\text{Tr}\left(B_i \nabla \Phi_k\left(X^{k+1}\right) X^{k+1}\right)}{\text{Tr}\left(B_i^2 X^{k+1}\right)}, i = 1, \ldots, l$;
6:     $S^{k+1} \leftarrow C - \mathcal{A}^*(y^{k+1}) - \mathcal{B}^*(z)$;
7:     Compute a descent direction U from S^{k+1} according to Theorem 4.6;
8:     Update p as described in Section 5.1;
9:     **if** $\|\mathcal{A}(X^{k+1}) - b\|/(1 + \|b\|) > \tau \|\text{grad}\, \Psi_k(Y^{k+1})\|$ **then**
10:        $\sigma_{k+1} \leftarrow \min\{\gamma \sigma_k, \sigma_{\max}\}$;
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);

---

    subproblem at step 3 would fulfill the stopping criteria (4.15). In other words, we solve the ALM subproblem without imposing optimality requirements. Intriguingly, global convergence is still observed in all numerical experiments presented in Section 6.

- Most computation of the algorithm could be performed with Y^{k+1} to avoid forming the big matrix $X^{k+1} = Y^{k+1}(Y^{k+1})^\mathsf{T}$.

- At Step 7, we need to perform an eigenvalue decomposition at each outer iteration of the ALM. As mentioned earlier, the ALM can enjoy fast linear convergence when the Riemannian trust-region method is adopted to solve the ALM subproblem. For the numerical examples tested in this paper, we perform *full* eigenvalue decomposition and the ALM typically returns an approximately optimal solution with KKT residues <1e-8 in a few tens of outer iterations, implying that only a few tens of eigenvalue decompositions are required. Therefore, the computational cost of (full) eigenvalue decomposition is manageable (at least for the SDP size investigated in this study). In its current form, the algorithm may not be suitable for large-scale problems where performing full eigenvalue decomposition is computationally prohibitive. Nonetheless, we could employ partial eigenvalue decomposition to further decrease the computational cost when tackling SDPs of larger size, and we leave its numerical implementation and detailed analysis in future work.

    The following theorem provides a posterior guarantee of global optimality for the output of Algorithm 5.1.

Theorem 5.3 *If the sequence* $\{(X^k, y^k, S^k)\}_{k \in \mathbb{N}}$ *generated by Algorithm 5.1 converges so that* $\lim_{k \to \infty} X^k = X^\star$, $\lim_{k \to \infty} y^k = y^\star$, *and* $\lim_{k \to \infty} S^k = S^\star$. *Then* $(X^\star, y^\star, S^\star)$ *is a KKT point of (SDP-M), i.e., it satisfies (2.1) with* $z^\star$ *being given by*

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

Proof By the updating rule $y^{k+1} = y^k - \sigma_k(\mathcal{A}(X^{k+1}) - b)$ and the convergence of $\{y^k\}_{k \in \mathbb{N}}$, we must have $\mathcal{A}(X^\star) = b$ and thus $X^\star$ is a feasible solution to (SDP-M). If $X^\star$ is not optimal, then by Theorem 4.4, we have $S^\star \not\succeq 0$. So there exists a second-order descent direction for

(ALMS-k) starting from $Y^\star$ due to Theorem 4.6, which contradicts to the fact that $X^\star$ is the limit point of $\{X^k\}_{k\in\mathbb{N}}$. Thus $X^\star$ is optimal and $S^\star \succeq 0$. Moreover, as $S^\star$ is the dual variable to $X^\star$, we have $\langle X^\star, S^\star \rangle = 0$ by strong duality.   □

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 (ALMS-k). `ManiSDP` is freely available at

<div align="center">

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

</div>

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. It would be very interesting to compare also with the solver `SDPDAL` of [47], which is, however, currently impossible as `SDPDAL` is not publicly available. 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 $10,000$s; "∗∗" 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:

$$\eta_p = \frac{\|\mathcal{A}(X) - b\|}{1 + \|b\|}, \ \eta_d = \frac{\max\{0, -\lambda_{\min}(S)\}}{1 + |\lambda_{\max}(S)|}, \ \eta_g = \frac{|\langle C, X \rangle - b^\mathsf{T} y|}{1 + |\langle C, X \rangle| + |b^\mathsf{T} y|}. \quad (6.1)$$

Given a tolerance tol > 0, the SDP solver terminates when $\eta_{\max} := \max\{\eta_p, \eta_d, \eta_g\} \leq$ tol, and we set tol $=$ 1e-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 $V =$

$\{1, \ldots, N\}$ 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:

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

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

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

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 $m = 0$ and $\mathcal{M}$ matching (Unit-diagonal).

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, 47] for similar experiments on these graphs.

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 formulized as the convex optimization problem:

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

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**

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

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

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	5.3e-09	**7.87**
	#2	550,565	-	-	1.3e-06	14.7	4.5e-09	131	3.2e-10	**7.92**
	#3	550,590	-	-	8.6e-07	15.3	4.1e-09	143	3.7e-10	**8.32**
3000	#1	930,328	-	-	1.8e-06	51.4	3.1e-08	238	9.1e-11	**21.6**
	#2	929,882	-	-	6.7e-07	49.4	3.2e-08	217	1.0e-10	**22.5**
	#3	930,080	-	-	3.6e-06	45.4	3.1e-08	216	4.1e-10	**22.2**
4000	#1	1,318,563	-	-	1.0e-06	88.7	4.8e-08	532	3.2e-10	**48.3**
	#2	1,318,488	-	-	1.6e-06	99.1	1.9e-09	548	2.9e-10	**47.2**
	#3	1,318,885	-	-	2.2e-06	96.8	4.7e-08	519	2.7e-10	**49.5**
5000	#1	1,711,980	-	-	1.2e-06	157	1.4e-09	1143	1.5e-10	**86.3**
	#2	1,711,445	-	-	1.0e-06	166	1.6e-09	1084	2.4e-10	**86.8**
	#3	1,711,660	-	-	1.1e-06	177	1.3e-09	1111	1.7e-10	**90.4**
6000	#1	2,107,303	-	-	2.2e-07	272	2.1e-09	1883	4.7e-09	**139**
	#2	2,106,628	-	-	1.1e-06	260	2.2e-09	2001	1.5e-10	**145**
	#3	2,106,039	-	-	1.3e-06	271	2.5e-09	1979	2.0e-10	**145**
8000	#1	2,900,179	-	-	2.1e-06	498	1.5e-08	3417	5.2e-11	**210**
	#2	2,900,585	-	-	3.4e-06	449	3.0e-08	4374	2.2e-10	**213**
	#3	2,900,182	-	-	3.2e-06	490	3.5e-08	4307	2.0e-10	**209**
10000	#1	3,695,929	-	-	1.1e-06	800	1.4e-09	8370	1.9e-10	**369**
	#2	3,696,602	-	-	2.1e-06	789	8.6e-09	8849	2.1e-10	**363**
	#3	3,696,604	-	-	1.1e-06	798	7.2e-09	8502	2.5e-10	**354**
12000	#1	4,493,420	-	-	7.8e-07	1310	*	*	8.3e-11	**568**
	#2	4,494,532	-	-	7.1e-07	1291	*	*	1.8e-10	**578**
	#3	4,493,391	-	-	3.5e-07	1330	*	*	4.9e-10	**590**

where E_{ij} is a $s \times t$ matrix with 1 at its (i, j)-position and 0 otherwise. Note also that (6.3) fits in (SDP-M) with $\mathcal{M}$ being the Euclidean manifold ($l = 0$). 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 $M = M_1 M_2^{\mathsf{T}}$, 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. In particular, ManiSDP is twice faster than SDPLR, and is faster than SDPNAL+ by a order of magnitude.

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

6.3 Binary Quadratic Programs

Let us consider the binary quadratic program given by

$$\begin{cases} \inf\limits_{\mathbf{x}\in\mathbb{R}^q} & \mathbf{x}^\mathsf{T} Q\mathbf{x} + \mathbf{c}^\mathsf{T}\mathbf{x} \\ \text{s.t.} & x_i^2 = 1, \quad i = 1,\ldots,q, \end{cases} \tag{BQP}$$

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 hierarchy[2] [27]. 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 generated instances as observed in [26, 52]. In the following we outline the ingredients of the second-order moment relaxation for (BQP). Let

$$v(\mathbf{x}) := [1, x_1, \ldots, x_q, x_1 x_2, x_1 x_3, \ldots, x_{q-1} x_q]^\mathsf{T}$$

be the vector of monomials in $\mathbf{x}$ up to degree two (excluding $x_i^2, i = 1, \ldots, q$) and $M := v(\mathbf{x})v(\mathbf{x})^\mathsf{T}$ 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 $x_i^2 = 1, i = 1, \ldots, q$, 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), which fits in (SDP-M) with $\mathcal{M}$ matching (Unit-diagonal).

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 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 (< 1s for $q = 10$ while ~ 50s 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

---

[2] Under mild conditions, the optima of the hierarchy converge to the optimum of the polynomial optimization problem.

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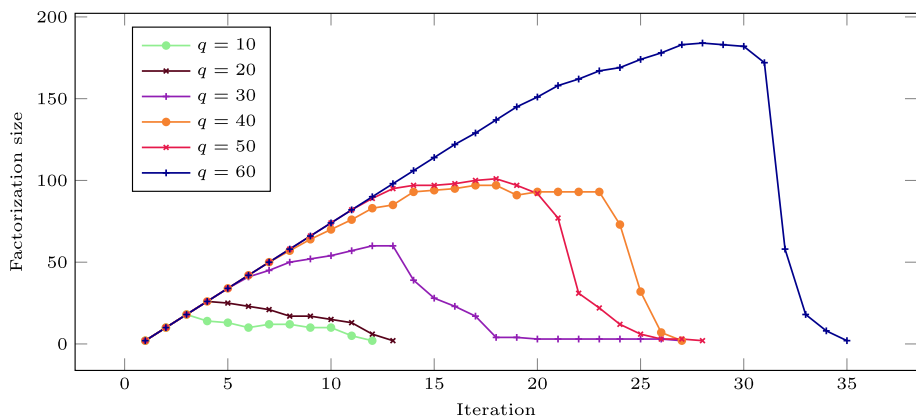
Fig. 1 The factorization size per iteration in solving (BQP)

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 $2 \sim 35$.

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 $q = 120$
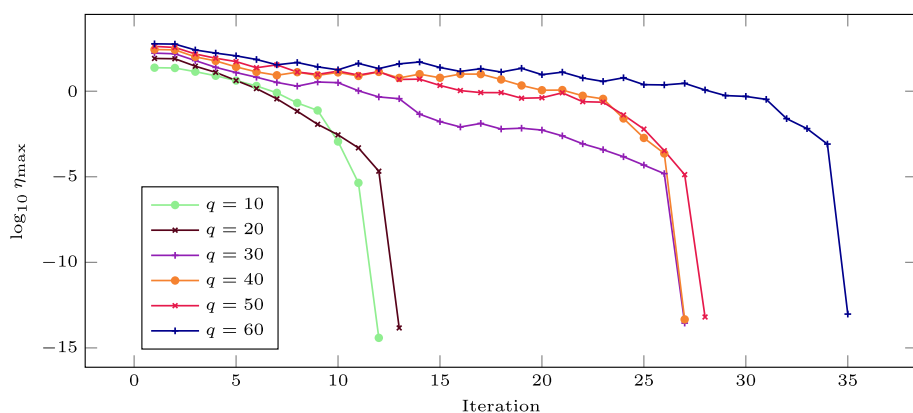
Fig. 2 The maximal KKT residue per iteration in solving (BQP)

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

for which the SDP has matrix dimension $n = 7261$ and contains $m = 17,869,161$ affine constraints, far beyond the reach of other SDP solvers!

6.4 Minimizing Quartic Polynomials on the Unit Sphere

Let us consider the problem of minimizing a quartic polynomial on the unit sphere:

$$\begin{cases} \inf\limits_{\mathbf{x} \in \mathbb{R}^q} & \mathbf{c}^\mathsf{T} \cdot [\mathbf{x}]_4 \\ \text{s.t.} & \sum_{i=1}^{q} x_i^2 = 1, \end{cases} \tag{QS}$$

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 generated instances of (QS) [52]. Let

$$v(\mathbf{x}) := [1, x_1, \ldots, x_q, x_1^2, x_1 x_2, x_1 x_3, \ldots, x_{q-1} x_q, x_q^2]^\mathsf{T}$$

be the vector of monomials in $\mathbf{x}$ up to degree two and $M := v(\mathbf{x}) v(\mathbf{x})^\mathsf{T}$ 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), which fits in (SDP-M) with $\mathcal{M}$ being the Euclidean manifold ($l = 0$).

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**

For each $q \in \{10, 20, 30, 40, 50, 60\}$, we generate three random instances of (QS) by taking $\mathbf{c} \in \mathbb{R}^{|[\mathbf{x}]_4|}$ 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 (< 1s for $q = 10$ while ~ 50s 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.

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.

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:

$$\min_{\|q\|=1} \sum_{i=1}^{N} \min \left\{ \frac{\|\tilde{z}_i - q \circ \tilde{w}_i \circ q^{-1}\|^2}{\beta_i^2}, 1 \right\}, \tag{6.4}$$
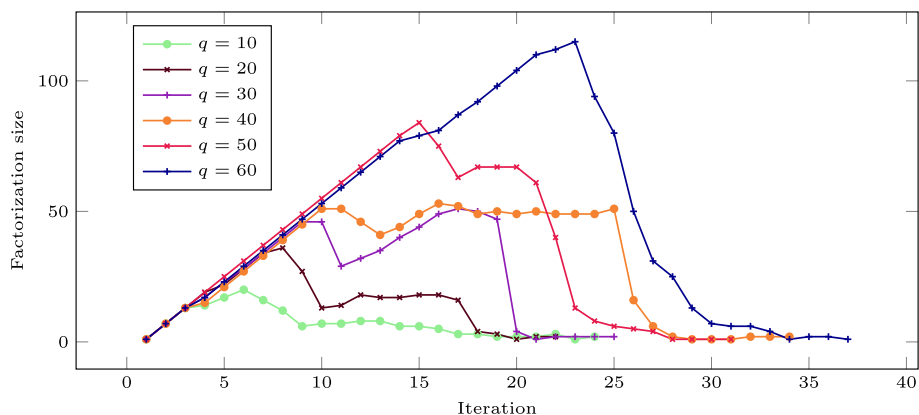
Fig. 3 The factorization size per iteration in solving (QS)
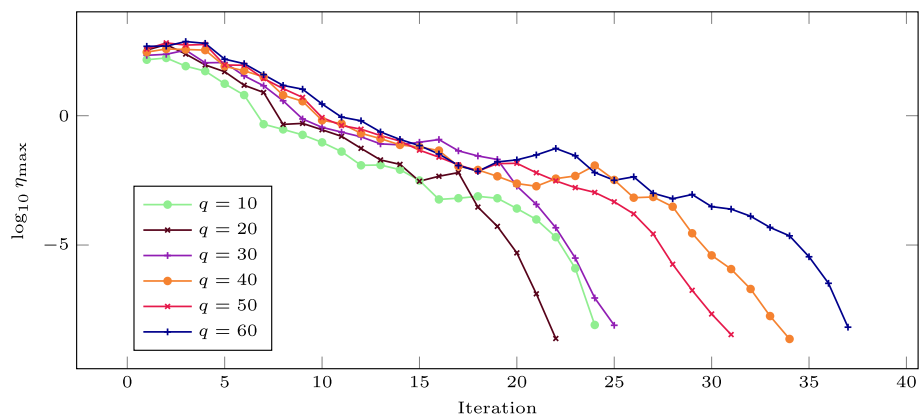

Fig. 4 The maximal KKT residue per iteration in solving (QS)

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, "∘" 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 [50, 53]. By introducing N binary variables $\{\theta_i\}_{i=1}^N$, Problem (6.4) can be equivalently reformulated as a polynomial optimization problem:

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

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 [50] 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

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

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**

be the sparse set of monomials in $\mathbf{x}$ of degree up to two. We build $M = v(\mathbf{x})v(\mathbf{x})^\mathsf{T}$ 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 ($\theta_i^2 qq^\mathsf{T} = qq^\mathsf{T}$); (2) the off-diagonal 4×4 blocks are symmetric ($\theta_i \theta_j qq^\mathsf{T} \in \mathbb{S}_4$). 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 $\mathrm{Tr}(M) = N + 1$ and so we let $\mathcal{B}(X) = d$ impose the trace constraint on X. Consequently, this leads to an SDP relaxation of size

$$n = 4N + 4, \quad m = 3N^2 + 13N + 1, \tag{6.6}$$

which fits in (SDP-M) with $\mathcal{M}$ matching (Unit-trace) after scaling X by $\frac{1}{N+1}$.

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 ($< 20s$ for $N = 10$ while $\sim 600s$ 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.
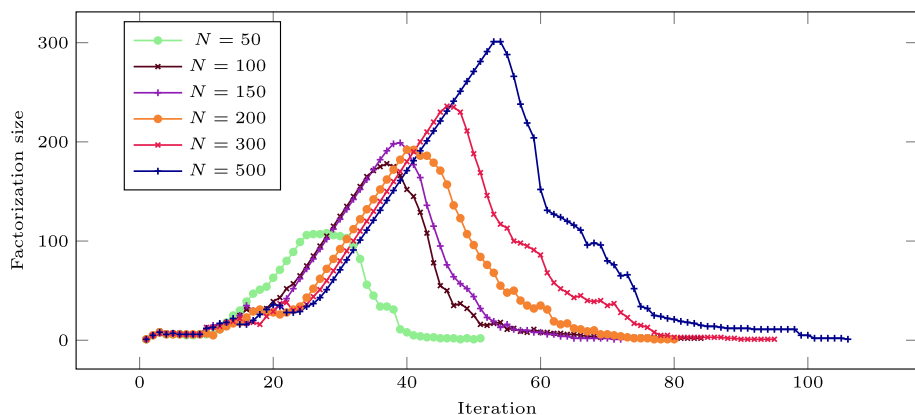
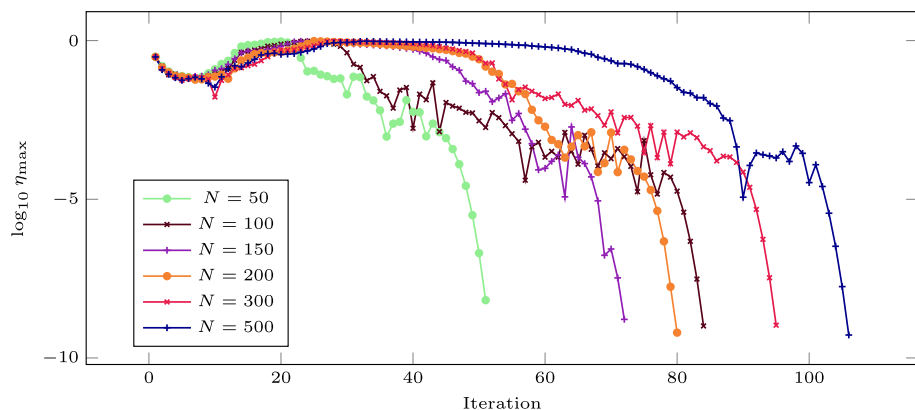**Fig. 5** The factorization size per iteration in solving (RRS)

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

6.6 Nearest Structured Rank Deficient Matrices

Let us consider the problem of finding the nearest structured rank deficient matrix:

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

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 [31]. We can reformulate (6.7) as the following polynomial optimization problem:

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

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 $\mathbf{x} = [z^\mathsf{T}, u^\mathsf{T}]^\mathsf{T} \in \mathbb{R}^{s+N}$ be the vector of
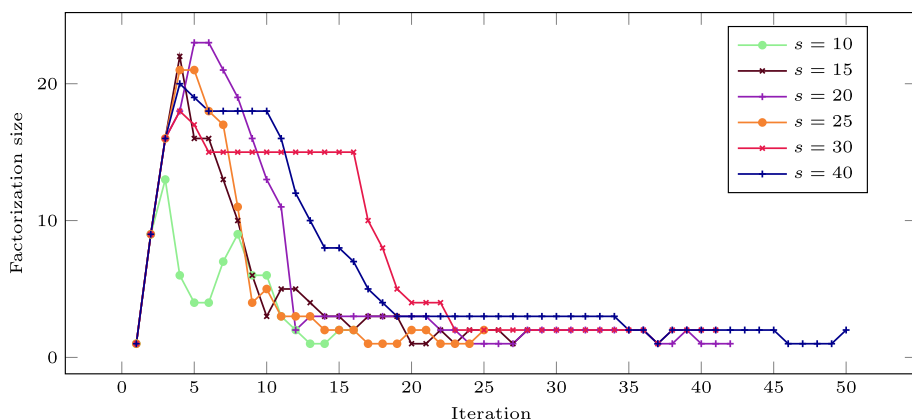
Fig. 7 The factorization size per iteration in solving (NSRD)

variables involved in (NSRD), and let

$$v(\mathbf{x}) = [z^{\mathsf{T}}, u_1 z^{\mathsf{T}}, \ldots, u_N z^{\mathsf{T}}]^{\mathsf{T}} \in \mathbb{R}^{s(N+1)} \tag{6.8}$$

be the sparse set of monomials in $\mathbf{x}$ of degree up to two. We build $M = v(\mathbf{x})v(\mathbf{x})^{\mathsf{T}}$ 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 $u_i u_j zz^{\mathsf{T}}$ are symmetric; (2) each of 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

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

which fits in (SDP-M) with $\mathcal{M}$ being the Euclidean manifold ($l = 0$).

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 ($\sim$ 20s for $s = 10$ while $\sim$ 1500s 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.

The numerical experiments indicate that ManiSDP typically outperforms SDPLR even in the case of $l = 0$ in which there is no non-trivial manifold structure to exploit. This could be explained as follows: (1) The Riemannian trust-region method enjoys superlinear (or

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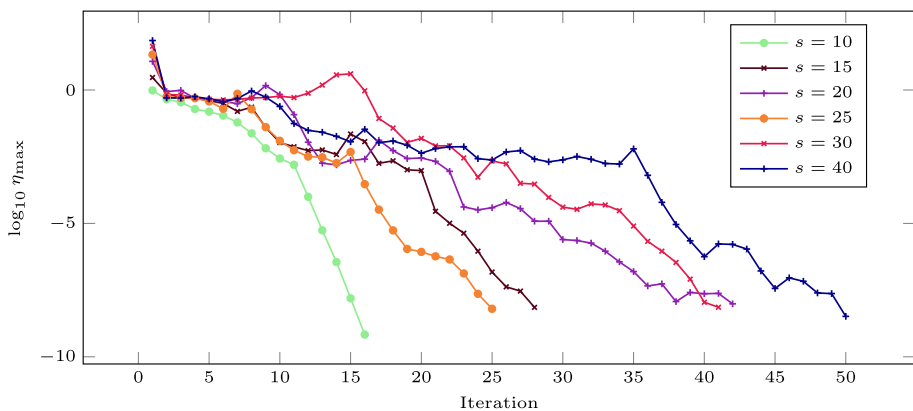
Fig. 8 The maximal KKT residue per iteration in solving (NSRD)

even quadratic) convergence [1] which may guarantee fast linear convergence of the ALM whereas the ALM with L-BFGS (implemented in SDPLR) cannot achieve linear convergence; (2) ManiSDP implements the adaptive strategy of updating the factorization size that improves the performance a lot whereas SDPLR utilizes a fixed factorization size ($\sim \sqrt{2m}$); (3) ManiSDP allows to decrease the penalty parameter (note that a large penalty parameter makes the ALM subproblem more difficult to solve) whereas SDPLR does not.

Table 9 Running time for different choices of the initial factorization size p_0

	p_0	1	5	10	15	20	25
(MC) ($n = 4000$)	time	28.4	25.6	25.8	30.9	30.0	28.1
(BQP) ($q = 50$)	p_0	2	10	20	30	40	50
	time	40.7	41.5	38.4	39.7	38.3	37.9
(QS) ($q = 50$)	p_0	1	10	20	30	40	50
	time	36.8	45.9	71.0	96.9	122	129
(RRS) ($N = 150$)	p_0	1	10	20	30	40	50
	time	28.7	30.7	30.5	32.9	31.2	31.2
(NSRD) ($s = 20$)	p_0	1	10	20	30	40	50
	time	36.8	38.0	44.9	31.8	47.0	42.9

Table 10 Comparison of running time of `ManiSDP` with and without the adaptive strategies. `ManiSDP 1`: `ManiSDP` with the adaptive strategies; `ManiSDP 2`: `ManiSDP` without the adaptive strategies

(MC)	n	2000	4000	6000	8000	10000	12000
	`ManiSDP 1`	8.03	48.0	143	210	362	578
	`ManiSDP 2`	9.66	51.7	151	286	430	652
(BQP)	q	10	20	30	40	50	60
	`ManiSDP 1`	0.19	0.74	4.78	8.87	48.2	331
	`ManiSDP 2`	0.56	1.65	12.1	35.1	156	1906
(QS)	q	10	20	30	40	50	60
	`ManiSDP 1`	0.23	0.64	7.95	12.6	38.3	89.0
	`ManiSDP 2`	0.89	2.17	27.4	73.7	286	622
(RRS)	N	50	100	150	200	300	500
	`ManiSDP 1`	3.18	20.4	34.6	66.4	193	819
	`ManiSDP 2`	12.0	113	163	337	765	4721
(NSRD)	s	10	15	20	25	30	40
	`ManiSDP 1`	1.15	11.0	44.5	302	356	1918
	`ManiSDP 2`	2.67	40.1	242	2074	2238	5144

6.7 Influence of the Initial Factorization Size p_0

In this subsection, we test the performance of `ManiSDP` under different choices of the initial factorization size p_0 on the problems described in Sections 6.2–6.6. For each problem of a fixed size and different choices of p_0, we run three random instances and then take the average running time. The results are displayed in Table 9. It can be seen that except Problem (QS), different choices of p_0 do not make big difference on the running time, while for (QS), a larger p_0 leads to increment of the running time.

6.8 Comparison of `ManiSDP` With and Without the Adaptive Strategies

In this subsection, we compare the performance of `ManiSDP` with and without the adaptive strategies introduced in Sections 5.1–5.2 on the problems described in Sections 6.2–6.6. For each problem of each size, we run three random instances and then take the average running time. The results are displayed in Table 10, from which we see that the adaptive strategies

(significantly) enhance the performance of `ManiSDP` and speed up the algorithm typically by several (up to 7.5) times.

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 under certain conditions despite the non-convexity brought by the Burer-Monteiro factorization. A practical algorithm is provided and diverse numerical experiments demonstrate its superior performance. It has been shown that our solver `ManiSDP` is capable of solving linear SDPs with millions of equality constraints to a very high precision in a reasonable time.

More research is required to achieve a comprehensive understanding of Algorithm 5.1. In particular, we believe that a global convergence result could be established under much weaker conditions. Another interesting point is the fast convergence rate of the algorithm that we empirically observed. These issues will be pursued in our future work.

We emphasize that `ManiSDP` is still in 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) designing a line search method to determine the step size for escaping from saddle points; (2) preconditioning for the Riemannian Hessian; (3) more efficiently escaping from saddle points; (4) handling SDPs with inequality constraints. Moreover, as SDPs may contain multiple PSD blocks (e.g., SDP relaxations for sparse polynomial optimization problems [44–46]), 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.

Data Availability The authors confirm that all data generated or analysed during this study are included in this article.

Declarations

References

1. Absil, P.-A., Baker, C.G., Gallivan, K.A.: Trust-region methods on Riemannian manifolds. Found. Comput. Math. **7**, 303–330 (2007). https://doi.org/10.1007/s10208-005-0179-9
2. Absil, P.A., Mahony, R., Trumpf, J.: An extrinsic look at the Riemannian Hessian, in Geometric Science of Information: First International Conference, GSI: Paris, France, August 28–30, 2013. Proceedings, Springer **2013**, 361–368 (2013)
3. Andersen, E.D., Roos, C., Terlaky, T.: On implementing a primal-dual interior-point method for conic quadratic optimization. Math. Program. **95**, 249–277 (2003)

4. Bai, X., Wei, H., Fujisawa, K., Wang, Y.: Semidefinite programming for optimal power flow problems. International Journal of Electrical Power & Energy Systems **30**, 383–392 (2008)
5. Bellavia, S., Gondzio, J., Porcelli, M.: A relaxed interior point method for low-rank semidefinite programming problems with applications to matrix completion. J. Sci. Comput. **89**, 1–36 (2021)
6. Boumal, N., Mishra, B., Absil, P.-A., Sepulchre, R.: Manopt, a Matlab toolbox for optimization on manifolds. J. Mach. Learn. Res. **15**, 1455–1459 (2014). (https://www.manopt.org)
7. Boumal, N., Voroninski, V., Bandeira, A.: The non-convex burer-monteiro approach works on smooth semidefinite programs, Adv. Neural Inf. Process Syst., 29 (2016)
8. Boumal, N., Voroninski, V., Bandeira, A.S.: Deterministic guarantees for burer-monteiro factorizations of smooth semidefinite programs. Commun. Pure Appl. Math. **73**, 581–608 (2020)
9. Burer, S., Monteiro, R.D.: A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. Math. Program. **95**, 329–357 (2003)
10. Burer, S., Monteiro, R.D.: Local minima and convergence in low-rank semidefinite programming. Math. Program. **103**, 427–444 (2005)
11. Candes, E., Recht, B.: Exact matrix completion via convex optimization. Commun. ACM **55**, 111–119 (2012)
12. Candès, E.J., Tao, T.: The power of convex relaxation: Near-optimal matrix completion. IEEE Trans. Inf. Theory **56**, 2053–2080 (2010)
13. Cavalcanti, D., Skrzypczyk, P.: Quantum steering: A review with focus on semidefinite programming. Rep. Prog. Phys. **80**, 024001 (2016)
14. Cifuentes, D.: A convex relaxation to compute the nearest structured rank deficient matrix. SIAM J. Matrix Anal. Appl. **42**, 708–729 (2021)
15. Cifuentes, D.: On the burer–monteiro method for general semidefinite programs, Optim. Lett., pp. 1–11 (2021)
16. Cifuentes, D., Agarwal, S., Parrilo, P.A., Thomas, R.R.: On the local stability of semidefinite relaxations. Math. Program. **193**, 629–663 (2022)
17. Cui, Y., Sun, D., Toh, K.-C.: On the r-superlinear convergence of the kkt residuals generated by the augmented lagrangian method for convex composite conic programming. Math. Program. **178**, 381–415 (2019)
18. De Maio, A., De Nicola, S., Huang, Y., Luo, Z.-Q., Zhang, S.: Design of phase codes for radar performance optimization with a similarity constraint. IEEE Trans. Signal Process. **57**, 610–621 (2008)
19. Garstka, M., Cannon, M., Goulart, P.: Cosmo: A conic operator splitting method for convex conic problems. J. Optim. Theory Appl. **190**, 779–810 (2021)
20. Goemans, M.X.: Semidefinite programming in combinatorial optimization. Math. Program. **79**, 143–161 (1997)
21. Habibi, S., Kavand, A., Kocvara, M., Stingl, M.: Barrier and penalty methods for low-rank semidefinite programming with application to truss topology design, arXiv preprint arXiv:2105.08529, (2021)
22. Han, Q., Li, C., Lin, Z., Chen, C., Deng, Q., Ge, D., Liu, H., Ye, Y.: A low-rank admm splitting approach for semidefinite programming, arXiv preprint arXiv:2403.09133, (2024)
23. Helmberg, C., Rendl, F.: A spectral bundle method for semidefinite programming. SIAM J. Optim. **10**, 673–696 (2000)
24. Huang, Y., Gleich, D.F.: Suboptimality bounds for trace-bounded sdps enable a faster and scalable low-rank sdp solver sdplr+, arXiv preprint arXiv:2406.10407, (2024)
25. Journée, M., Bach, F., Absil, P.-A., Sepulchre, R.: Low-rank optimization on the cone of positive semidefinite matrices. SIAM J. Optim. **20**, 2327–2351 (2010)
26. 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: Proceedings 8. Springer **2001**, 293–303 (2001)
27. Lasserre, J.-B.: Global Optimization with Polynomials and the Problem of Moments. SIAM J. Optim. **11**, 796–817 (2001)
28. Liao, F.-Y., Ding, L., Zheng, Y.: Inexact augmented lagrangian methods for conic optimization: Quadratic growth and linear convergence. Adv. Neural. Inf. Process. Syst. **37**, 41013–41050 (2025)
29. Liu, C., Boumal, N.: Simple algorithms for optimization on riemannian manifolds with constraints. Applied Mathematics & Optimization **82**, 949–981 (2020)
30. Luo, Z.-Q., Ma, W.-K., So, A.M.-C., Ye, Y., Zhang, S.: Semidefinite relaxation of quadratic optimization problems. IEEE Signal Process. Mag. **27**, 20–34 (2010)
31. Markovsky, I.: Structured low-rank approximation and its applications. Automatica **44**, 891–909 (2008)
32. Monteiro, R.D., Sujanani, A., Cifuentes, D.: A low-rank augmented lagrangian method for large-scale semidefinite programming based on a hybrid convex-nonconvex approach, arXiv preprint arXiv:2401.12490, (2024)

33. Parrilo, P.A., Lall, S.: Semidefinite programming relaxations and algebraic optimization in control. Eur. J. Control. **9**, 307–321 (2003)
34. Lee, C.P., Liang, L., Tang, T., Toh, K.-C.: Accelerating nuclear-norm regularized low-rank matrix optimization through Burer-Monteiro decomposition, Journal of Machine Learning Research, **25**(379), 1–52 (2024)
35. Rockafellar, R.T.: Augmented lagrangians and applications of the proximal point algorithm in convex programming. Math. Oper. Res. **1**, 97–116 (1976)
36. Rosen, D.M.: 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, pp. 551–566 (2021)
37. Rosen, D.M., Carlone, L., Bandeira, A.S., Leonard, J.J.: Se-sync: A certifiably correct algorithm for synchronization over the special euclidean group. Int. J. Robot. Res. **38**, 95–125 (2019)
38. Souto, M., Garcia, J.D., Veiga, Á.: Exploiting low-rank structure in semidefinite programming by approximate operator splitting. Optimization **71**, 117–144 (2022)
39. Sun, D., Toh, K.-C., Yuan, Y., Zhao, X.-Y.: SDPNAL+: A Matlab software for semidefinite programming with bound constraints (version 1.0). Optim. Methods Softw. **35**, 87–115 (2020)
40. Tang, T., Toh, K.-C.: A feasible method for general convex low-rank sdp problems, SIAM Journal on Optimization, **34**(3), 2169–2200 (2024)
41. Tang, T., Toh, K.-C.: Solving graph equipartition SDPs on an algebraic variety. Math. Program. **204**, 299–347 (2024)
42. Toh, K.-C., Todd, M.J., Tütüncü, R.H.: Sdpt3-a matlab software package for semidefinite programming, version 1.3. Optim. Methods Softw. **11**, 545–581 (1999)
43. Vandenberghe, L., Boyd, S.: Semidefinite programming. SIAM Rev. **38**, 49–95 (1996)
44. Wang, J., Magron, V., Lasserre, J.-B.: Chordal-TSSOS: A moment-SOS hierarchy that exploits term sparsity with chordal extension. SIAM J. Optim. **31**, 114–141 (2021)
45. Wang, J., Magron, V., Lasserre, J.-B.: TSSOS: A moment-SOS hierarchy that exploits term sparsity. SIAM J. Optim. **31**, 30–58 (2021)
46. Wang, J., Magron, V., Lasserre, J.-B., Mai, N.H.A.: CS-TSSOS: Correlative and term sparsity for largescale polynomial optimization, ACM Transactions on Mathematical Software, **48**(4), 1–26 (2022)
47. Wang, Y., Deng, K., Liu, H., Wen, Z.: A decomposition augmented lagrangian method for low-rank semidefinite programming. SIAM J. Optim. **33**, 1361–1390 (2023)
48. Wen, Z., Goldfarb, D., Yin, W.: Alternating direction augmented lagrangian methods for semidefinite programming. Math. Program. Comput. **2**, 203–230 (2010)
49. Wolkowicz, H., Saigal, R., Vandenberghe, L.: Handbook of semidefinite programming: theory, algorithms, and applications, vol. 27, Springer Science & Business Media, (2012)
50. Yang, H., Carlone, L.: A quaternion-based certifiably optimal solution to the wahba problem with outliers, in Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 1665–1674 (2019)
51. Yang, H., Carlone, L.: Certifiably optimal outlier-robust geometric perception: Semidefinite relaxations and scalable global optimization, IEEE Transactions on Pattern Analysis and Machine Intelligence, (2022)
52. Yang, H., Liang, L., Carlone, L., Toh, K.-C.: An inexact projected gradient method with rounding and lifting by nonlinear programming for solving rank-one semidefinite relaxation of polynomial optimization, Math. Program., pp. 1–64 (2022)
53. Yang, H., Shi, J., Carlone, L.: Teaser: Fast and certifiable point cloud registration. IEEE Trans. Rob. **37**, 314–333 (2020)
54. Yang, L., Sun, D., Toh, K.-C.: SDPNAL+: a majorized semismooth Newton-CG augmented lagrangian method for semidefinite programming with nonnegative constraints. Math. Program. Comput. **7**, 331–366 (2015)
55. Yurtsever, A., Tropp, J.A., Fercoq, O., Udell, M., Cevher, V.: Scalable semidefinite programming. SIAM Journal on Mathematics of Data Science **3**, 171–200 (2021)
56. Zhang, R.Y., Lavaei, J.: Modified interior-point method for large-and-sparse low-rank semidefinite programs, in 2017 IEEE 56th Annual Conference on Decision and Control (CDC), IEEE, pp. 5640–5647 (2017)
57. Zhang, R.Y., Lavaei, J.: Sparse semidefinite programs with guaranteed near-linear time complexity via dualized clique tree conversion. Math. Program. **188**, 351–393 (2021)
58. Zheng, Y., Fantuzzi, G., Papachristodoulou, A., Goulart, P., Wynn, A.: Chordal decomposition in operator-splitting methods for sparse semidefinite programs. Math. Program. **180**, 489–532 (2020)
59. Zhou, Y., Bao, C., Ding, C., Zhu, J.: A semismooth newton based augmented lagrangian method for nonsmooth optimization on matrix manifolds. Math. Program. **201**, 1–61 (2022)