A Low-Rank Augmented Lagrangian Method for Large-Scale Semidefinite Programming Based on a Hybrid Convex-Nonconvex Approach

Renato D.C. Monteiro * Arnesh Sujanani * Diego Cifuentes †

January 22, 2024 (second version: March 15, 2024, third version: March 31, 2024)

Abstract

This paper introduces HALLaR, a new first-order method for solving large-scale semidefinite programs (SDPs) with bounded domain. HALLaR is an inexact augmented Lagrangian (AL) method where the AL subproblems are solved by a hybrid low-rank (HLR) method. The recipe behind HLR is based on two key ingredients: 1) an adaptive inexact proximal point method with inner acceleration; 2) Frank-Wolfe steps to escape from spurious local stationary points. In contrast to the low-rank method of Burer and Monteiro, HALLaR finds a near-optimal solution (with provable complexity bounds) of SDP instances satisfying strong duality. Computational results comparing HALLaR to state-of-the-art solvers on several large SDP instances arising from maximum stable set, phase retrieval, and matrix completion, show that the former finds highly accurate solutions in substantially less CPU time than the latter ones. For example, in less than 20 minutes, HALLaR can solve a maximum stable set SDP instance with dimension pair $(n, m) \approx (10^6, 10^7)$ within 10^{-5} relative precision.

Keywords: semidefinite programming, augmented Lagrangian, low-rank methods, proximal point method, Frank-Wolfe method, iteration complexity, adaptive method, global convergence rate

1 Introduction

Semidefinite programming (SDP) has many applications in engineering, machine learning, sciences, finance, among other areas. However, solving large-scale SDPs is very computationally challenging. In particular, interior point methods usually get stalled in large-scale instances due to lack of memory. This has motivated a recent surge of first-order methods for solving SDPs that scale to larger instances [18, 25, 46, 51, 56, 63, 65, 67–69].

This paper introduces HALLaR, a new first-order method for solving SDPs with bounded trace. Let $\mathbb{S}^n$ be the space of symmetric $n \times n$ matrices with Frobenius inner product $\bullet$ and with positive semidefinite partial order $\succeq$. HALLaR solves the primal/dual pair of SDPs:

$$\min_X \quad \{C \bullet X \quad : \quad \mathcal{A}X = b, \quad X \in \Delta^n\} \tag{P}$$

*Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA, 30332-0205. (Email: monteiro@isye.gatech.edu & asujanani6@gatech.edu). These authors were partially supported by AFORS Grant FA9550-22-1-0088.

†Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA, 30332-0205. (Email: diego.cifuentes@isye.gatech.edu). This author was supported partially supported by the Office of Naval Research, N00014-23-1-2631.

$$\max_{p\in\mathbb{R}^m,\theta\in\mathbb{R}} \quad \{-b^T p-\theta \quad : \quad S:=C+\mathcal{A}^* p+\theta I\succeq 0, \quad \theta\geq 0\} \tag{D}$$

where $b\in\mathbb{R}^m$, $C\in\mathbb{S}^n$, $\mathcal{A}:\mathbb{S}^n\to\mathbb{R}^m$ is a linear map, $\mathcal{A}^*:\mathbb{R}^m\to\mathbb{S}^n$ is its adjoint, and Δ^n is the spectraplex

$$\Delta^n := \{X\in\mathbb{S}^n : \operatorname{tr} X\leq 1, X\succeq 0\}. \tag{1}$$

HALLaR is based on Burer and Monteiro's *low-rank* (LR) approach [7, 8] which is described in the next paragraph.

Low-rank approach. The LR approach is motivated by SDPs often having optimal solutions with small ranks. More specifically, it is known (see [2, 48, 55]) that $r_*\leq\sqrt{2m}$, where r_* is the smallest among the ranks of all optimal solutions of (P). The LR approach consists of solving subproblems obtained by restricting (P) to matrices of rank at most r, for some integer r, or equivalently, the nonconvex smooth reformulation

$$\min_{U} \quad \left\{C\bullet UU^T \quad : \quad \mathcal{A}(UU^T)=b, \quad \|U\|_F\leq 1, \quad U\in\mathbb{R}^{n\times r}\right\}. \tag{P_r}$$

Problems (P) and (P_r) are equivalent when $r\geq r_*$, in the sense that if U_* is optimal for (P_r) then $X_*=U_*U_*^T$ is optimal for (P). The advantage of (P_r) compared to (P) is that its matrix variable U has significantly less entries than that of (P) when $r\ll n$, namely, nr instead of $n(n+1)/2$. However, as (P_r) is nonconvex, it may have stationary points which are not globally optimal. For a generic instance, the following results are known: i) if $r\geq\sqrt{2m}$ then all local minima of (P_r) are globally optimal (see [3–5, 14, 15, 49]); and ii) if $r<\sqrt{2m}$ then (P_r) may have local minima which are not globally optimal (see [59]).

Outline of HALLaR. HALLaR is an inexact augmented Lagrangian (AL) method that generates sequences $\{X_t\}$ and $\{p_t\}$ according to the recursions

$$X_t \quad \approx \quad \arg\min_X \quad \{\mathcal{L}_\beta(X;p_{t-1}) \ : \ X\in\Delta^n\}, \tag{2a}$$

$$p_t \quad = \quad p_{t-1}+\beta(\mathcal{A}X_t-b) \tag{2b}$$

where

$$\mathcal{L}_\beta(X;p) \quad := \quad C\bullet X+p^T(\mathcal{A}X-b)+\frac{\beta}{2}\|\mathcal{A}X-b\|^2. \tag{3}$$

The key part of HALLaR is an efficient method, called Hybrid Low-Rank (HLR), for finding an approximate global solution X_t of the AL subproblem (2a). The HLR method solves subproblems of the form

$$\min_{Y} \quad \left\{\mathcal{L}_\beta(YY^T;p_{t-1}) \quad : \quad \|Y\|_F\leq 1, \quad Y\in\mathbb{R}^{n\times r}\right\} \tag{L_r}$$

for some integer $r\geq 1$. Subproblem (L_r) is equivalent to the subproblem obtained by restricting X in (2a) to matrices with rank at most r. Since (L_r) is nonconvex, it may have a *spurious* (near) stationary point, i.e., a (near) stationary point Y such that YY^T is not (nearly) optimal for (2a).

More specifically, HLR finds an approximate global solution X_t of (2a) by solving a sequence of nonconvex subproblems $(L_{r_k})_{k\geq 1}$ such that $r_{k+1}\leq r_k+1$, according to following steps: i) find a near stationary point $Y=Y_k\in\mathbb{R}^{n\times r_k}$ of (L_{r_k}) using an adaptive accelerated inexact proximal point (ADAP-AIPP) method that is based on a combination of ideas developed in [13, 36, 37, 47,

57]; ii) check if $Y_k Y_k^T$ is nearly optimal for (2a) through a minimum eigenvalue computation and terminate the method if so; else iii) use the following escaping strategy to move away from the current spurious near stationary point Y_k: perform a Frank-Wolfe (FW) step from Y_k to obtain a point $\tilde{Y}_k$ with either one column in which case (the unlikely one) r_{k+1} is set to one, or with $r_k + 1$ columns in which case r_{k+1} is set to $r_k + 1$, and use $\tilde{Y}_k$ as the initial iterate for solving $\mathrm{L}_{r_{k+1}}$. The initial pair $(r_1, \tilde{Y}_0)$ for HLR is chosen by using a warm start strategy, namely, as the pair obtained at the end of the HLR call for solving the previous subproblem (2a). It is worth noting that HALLaR only stores the current iterate Y and never computes the (implicit) iterate YY^T (lying in the X-space).

Under the strong duality assumption, it is shown that HALLaR obtains an approximate primal-dual solution of (P) and (D) with provable computational complexity bounds expressed in terms of parameters associated with the SDP instance and user-specified tolerances.

Computational impact. Our computational results show that HALLaR performs very well on many large-scale SDPs such as phase retrieval, maximum-stable-set, and matrix completion. In all these applications, HALLaR efficiently obtains accurate solutions for large-scale instances, largely outperforming other state-of-the-art solvers. For example, HALLaR takes approximately 1.75 hours (resp., 13 hours) on a personal laptop to solve within 10^{-5} relative precision maximum stable set SDP instance for a Hamming graph with $n \approx 4,000,000$ and $m \approx 40,000,000$ (resp., $n \approx 16,000,000$ and $m \approx 200,000,000$). Moreover, HALLaR takes approximately 11.3 hours on a personal laptop to solve within 10^{-5} relative precision a phase retrieval SDP instance with $n \approx 3,000,000$ and $m \approx 36,000,000$. An important reason for the good computational performance of HALLaR is that the rank of the iterates X_t remain relatively small throughout the whole algorithm.

Related works. This part describes other methods for solving large-scale SDPs. SDPNAL+ [63, 68] is an AL based method that solves each AL subproblem using a semismooth Newton-CG method. Algorithms based on spectral bundle methods (more generally bundle methods) have also been proposed and studied for solving large-scale SDPs (e.g., [9, 19, 30, 31, 45]). The more recent works (e.g., [18, 25, 42, 46, 69]) propose methods for solving large-scale SDPs based on the alternating direction method of multipliers (ADMM). The remaining of this section discusses in more detail works that rely on the nonconvex LR approach and the FW method, since those works are more closely related to this paper. The reader is referred to the survey paper [43] for additional methods for solving large scale SDPs.

The nonconvex LR approach of [7, 8] has been successful in solving many relevant classes of SDPs. The SDPLR method developed in these works solves (P_r) with an AL method whose AL subproblems are solved by a limited memory BFGS method. Although SDPLR only handles equality constraints, it is possible to modify it to handle inequalities (e.g., [38]). HALLaR is also based on the AL method but it applies it directly to (P) instead of (P_r). Moreover, in contrast to SDPLR, HALLaR solves the AL subproblems using the HLR method outlined above.

This paragraph describes works that solve (possibly a sequence of) (P_r) without using the AL method. Approaches that use interior point methods for solving (P_r) have been pursued for example in [52]. In the context of MaxCut SDPs, several specialized methods have been proposed which solve (P_r) using optimization techniques which preserves feasibility (e.g., [6, 21, 32, 35, 44]). Finally, Riemannian optimization methods have been used to solve special classes of SDPs where the feasible sets for (P_r) are smooth manifolds (e.g., [33, 35, 44, 53]).

The FW method minimizes a convex function $g(X)$ over a compact convex domain (e.g., the spectraplex Δ^n). It is appealing when a sparse solution is desired, where the notion of sparsity is

broad (e.g., small cardinality and/or rank). The FW method has been used (e.g., [29, 34, 56]) for solving SDP feasibility problems by minimizing $g(X) = \phi(\mathcal{A}X - b)$ where ϕ is either the squared norm function $\|\cdot\|^2$ or the function $\mathrm{LSE}(y) = \log(\sum_i \exp y_i)$. Several papers (e.g., [23, 28, 39, 50]) introduce variants of the FW method for general convex optimization problems.

In the remaining of the "Related works" part, we discuss AL-based SDP low-rank methods directly applicable to (P) and hence which (approximately) solves subproblems in the form of (2a). An interesting method for solving (P) in this manner is CGAL of [65, 66] which generates its iterates by performing only FW steps for the AL subproblems (2a). As HALLaR, the method of [66] only generates iterates in the Y-space. Its Lagrange multiplier update policy though differs from (2b) in that it updates the Lagrange multiplier in a more conservative way, i.e., with β in (2b) replaced by a usually much smaller $\gamma_t > 0$, and does so only when the size of the new tentative multiplier is not too large. Moreover, instead of using a pure FW method to solve (2a), an iteration of the subroutine HLR invoked by HALLaR to solve (2a) consists of an ADAP-AIPP call applied to (L_r) and, if HLR does not terminate, also a FW step (which generally increases the rank of the iterate by one). As demonstrated by our computational results, the use of ADAP-AIPP calls significantly reduces the number of FW steps performed by HALLaR, and, as a by-product, keeps the ranks of its iterates considerably smaller than those of the CGAL iterates.

The CGAL method was enhanced in [67] to derive a low-storage variant, namely, Sketchy-CGAL. Instead of explicitly storing its most recent Y-iterate as CGAL does, this variant computes a certain approximation of the above iterates lying in $\mathbb{R}^{n\times r}$ where $r \in \{1, \ldots, n-1\}$ is a specified threshold value whose purpose is to limit the rank of the stored approximation. It is shown in [67] that Sketchy-CGAL has $O(m + nr)$ memory storage, and that it outputs an $O(r^*/(r - r^* - 1))$-approximate solution of (P) (constructed using the sketch) under the assumption that $r > r^* + 1$, where r^* is the largest among the ranks of all optimal solutions of (P). In contrast to either CGAL or HALLaR, a disadvantage of Sketchy-CGAL is that the accuracy of its output primal approximate solution is often low and degrades further as r decreases, and can even be undetermined if $r \leq r^*+1$. Finally, alternative methods for solving SDPs with $O(m + nr^*)$ memory storage are presented in [20, 56, 60].

Finally, more recent SDP low-rank solvers have been proposed which approximately solve (2a) by solving a sequence of nonconvex subproblems in the form (L_r) using either a Riemannian trust-region approach [61] or a Riemannian semi-smooth Newton method [62].

Structure of the paper. This paper is organized into four sections. Section 2 discusses the HLR method for solving the AL subproblem (2a) and, more generally, smooth convex optimization problems over the spectraplex Δ^n. It also presents complexity bounds for HLR, given in Theorem 2.5. Section 3 presents HALLaR for solving the pair of SDPs (P) and (D) and presents the main complexity result of this paper, namely Theorem 3.2, which provides complexity bounds for HALLaR. Finally, Section 4 presents computational experiments comparing HALLaR with various solvers in a large collection of SDPs arising from stable set, phase retrieval, and matrix completion problems.

1.1 Basic Definitions and Notations

Let $\mathbb{R}^n$ be the space of n dimensional vectors, $\mathbb{R}^{n\times r}$ the space of $n \times r$ matrices, and $\mathbb{S}^n$ the space of $n \times n$ symmetric matrices. Let $\mathbb{R}^n_{++}$ ($\mathbb{R}^n_+$) be the convex cone in $\mathbb{R}^n$ of vectors with positive (nonnegative) entries, and let $\mathbb{S}^n_{++}$ ($\mathbb{S}^n_+$) be the convex cone in $\mathbb{S}^n$ of positive (semi)definite matrices. Let $\langle\cdot,\cdot\rangle$ and $\|\cdot\|$ be the Euclidean inner product and norm on $\mathbb{R}^n$, and let $\bullet$ and $\|\cdot\|_F$ be the Frobenius inner product and norm on $\mathbb{S}^n$. The minimum eigenvalue of a matrix $Q \in \mathbb{S}^n$ is denoted

by $\lambda_{\min}(Q)$, and $v_{\min}(Q)$ denotes a corresponding eigenvector of unit norm. For any $t > 0$ and $a \geq 0$, let $\log_a^+(t) := \max\{\log t, a\}$.

For a given closed convex set $C \subseteq \mathbb{R}^n$, its boundary is denoted by ∂C and the distance of a point $z \in \mathbb{R}^n$ to C is denoted by $\operatorname{dist}(z, C)$. The diameter of C, denoted D_C, is defined as

$$D_C := \sup\{\|Z - Z'\| : Z, Z' \in C\}. \tag{4}$$

The indicator function of C, denoted by δ_C, is defined by $\delta_C(z) = 0$ if $z \in C$, and $\delta_C(z) = +\infty$ otherwise. The domain of a function $h : \mathbb{R}^n \to (-\infty, \infty]$ is the set $\operatorname{dom} h := \{x \in \mathbb{R}^n : h(x) < +\infty\}$. Moreover, h is said to be proper if $\operatorname{dom} h \neq \emptyset$. The ϵ-subdifferential of a proper convex function $h : \mathbb{R}^n \to (-\infty, \infty]$ is defined by

$$\partial_\epsilon h(z) := \{u \in \mathbb{R}^n : h(z') \geq h(z) + \langle u, z' - z\rangle - \epsilon, \quad \forall z' \in \mathbb{R}^n\} \tag{5}$$

for every $z \in \mathbb{R}^n$. The classical subdifferential, denoted by $\partial h(\cdot)$, corresponds to $\partial_0 h(\cdot)$. Recall that, for a given $\epsilon \geq 0$, the ϵ-normal cone of a closed convex set C at $z \in C$, denoted by $N_C^\epsilon(z)$, is

$$N_C^\epsilon(z) := \{\xi \in \mathbb{R}^n : \langle \xi, u - z\rangle \leq \epsilon, \quad \forall u \in C\}.$$

The normal cone of a closed convex set C at $z \in C$ is denoted by $N_C(z) = N_C^0(z)$.

Given a differentiable function $\psi : \mathbb{R}^n \to \mathbb{R}$, its affine approximation at a point $\bar{z} \in \mathbb{R}^n$ is

$$\ell_\psi(z; \bar{z}) := \psi(\bar{z}) + \langle \nabla\psi(\bar{z}), z - \bar{z}\rangle \quad \forall z \in \mathbb{R}^n. \tag{6}$$

The function ψ is L-smooth on a set $\Omega \subseteq \mathbb{R}^n$ if its gradient is L-Lipschitz continuous on Ω, i.e.,

$$\|\nabla\psi(x') - \nabla\psi(x)\| \leq L\|x' - x\| \quad \forall x, x' \in \Omega. \tag{7}$$

The set of L-smooth functions on Ω is denoted by $\mathcal{C}^1(\Omega; L)$.

2 Hybrid Low-Rank Method

This section introduces a Hybrid Low-Rank (HLR) method which, as outlined in the introduction, uses a combination of the ADAP-AIPP method and Frank-Wolfe steps for approximately solving convex problems of the form as in (2a). This section consists of three subsections. The first subsection introduces the main problem that the HLR method considers and introduces a notion of the type of approximate solution that it aims to find. The second subsection presents the ADAP-AIPP method and its complexity results. The third subsection states the complete HLR method and establishes its total complexity.

2.1 Problem of Interest and Solution Type

Let $g : \mathbb{S}^n \to \mathbb{R}$ be a convex and differentiable function. The HLR method is developed in the context of solving the problem

$$g_* := \min\{g(Z) : Z \in \Delta^n\} \tag{8}$$

where Δ^n is the spectraplex as in (1) and g is L_g-smooth on Δ^n, i.e., there exists $L_g \geq 0$ such that

$$\|\nabla g(Z') - \nabla g(Z)\|_F \leq L_g\|Z' - Z\|_F \quad \forall Z, Z' \in \Delta^n. \tag{9}$$

The goal of the HLR method is to find a near-optimal solution of (8) whose definition is given immediately after the next result.

Lemma 2.1. *Let $Z \in \Delta^n$ be given and define*

$$\theta(Z) := \max\{-\lambda_{\min}(\nabla g(Z)), 0\}. \tag{10}$$

Then:

a) there hold

$$\theta(Z) \geq 0, \quad \nabla g(Z) + \theta(Z) I \succeq 0; \tag{11}$$

b) for any $\epsilon > 0$, the inclusion holds

$$0 \in \nabla g(Z) + \partial_\epsilon \delta_{\Delta^n}(Z) \tag{12}$$

if and only if

$$\nabla g(Z) \bullet Z + \theta(Z) \leq \epsilon. \tag{13}$$

Proof. (a) The result is immediate from the definition of $\theta(Z)$ in (10).

(b) It is easy to see that $\partial_\epsilon \delta_{\Delta^n}(Z) = N^\epsilon_{\Delta^n}(Z)$. Statement (b) then follows immediately from this observation, the definition of $\theta(Z)$ in (10), and Proposition A.2(b) with $G = \nabla g(Z)$. □

Relation (13) provides an easily verifiable condition for checking whether Z satisfies inclusion (12). Moreover, (13) is equivalent to the "complementary slackness" condition

$$(1 - \operatorname{tr} Z)\, \theta(Z) + [\nabla g(Z) + \theta(Z) I] \bullet Z \leq \epsilon. \tag{14}$$

Definition 2.2. *An ϵ-optimal solution of* (8) *is a matrix $Z \in \Delta^n$ satisfying relation* (12) *or* (13).

The next lemma shows that the objective value of an ϵ-optimal solution of (8) is within ϵ of the optimal value of (8).

Lemma 2.3. *An ϵ-optimal solution Z of* (8) *satisfies that $g(Z) - g_* \leq \epsilon$.*

Proof. Let Z_* be an optimal solution of (8). Relation (12) implies that $-\nabla g(Z) \in N^\epsilon_{\Delta^n}(Z)$ and hence that $\langle -\nabla g(Z), Z_* - Z\rangle \leq \epsilon$. It then follows from this relation and the fact that g is convex that

$$g(Z_*) - g(Z) \geq \langle \nabla g(Z), Z_* - Z\rangle \geq -\epsilon,$$

which immediately implies the result. □

2.2 The ADAP-AIPP Method

As already mentioned in the introduction, one iteration of the HLR method consists of a call to the ADAP-AIPP method followed by a FW step. The purpose of this subsection is to describe the details of the ADAP-AIPP method.

For a given integer s, consider the the subproblem obtained by restricting (8) to matrices Z of rank at most s, or equivalently, the reformulation

$$\min\{\tilde{g}(U) := g(UU^T) \ : \ U \in \bar{B}^s_1\}, \tag{15}$$

where

$$\bar{B}^s_r := \{U \in \mathbb{R}^{n\times s} : \|U\|_F \leq r\} \tag{16}$$

denotes the Frobenius ball of radius r in $\mathbb{R}^{n\times s}$. In this subsection, the above set will be denoted by $\bar{B}_r$ since the column dimension s remains constant throughout its presentation.

The goal of the ADAP-AIPP method is to find an approximate stationary solution of (15) as described in Proposition 2.4(a) below. Briefly, ADAP-AIPP is an inexact proximal point method which attempts to solve its (potentially nonconvex) prox subproblems using an accelerated composite gradient method, namely, ADAP-FISTA, whose description is given in Appendix B. A rough description of the j-th iteration of ADAP-AIPP is as follows: given $W_{j-1} \in \bar{B}_1$ and a positive scalar λ_{j-1}, ADAP-AIPP calls the ADAP-FISTA method to attempt to find a suitable approximate solution of the possibly nonconvex proximal subproblem

$$\min_{U \in \bar{B}_1} \left\{ \lambda \tilde{g}(U) + \frac{1}{2}\|U - W_{j-1}\|_F^2 \right\}, \tag{17}$$

where the first call made is always performed with $\lambda = \lambda_{j-1}$. If ADAP-FISTA successfully finds such a solution, ADAP-AIPP sets this solution as its next iterate W_j and sets λ as its next prox stepsize λ_j. If ADAP-FISTA is unsuccessful, ADAP-AIPP invokes it again to attempt to solve (17) with $\lambda = \lambda/2$. This loop always terminates since ADAP-FISTA is guaranteed to terminate with success when the objective in (17) becomes strongly convex, which occurs when λ is sufficiently small.

The formal description of the ADAP-AIPP method is presented below. For the sake of simplifying the input lists of the algorithms stated throughout this paper, the parameters σ and χ are considered universal ones (and hence not input parameters).

ADAP-AIPP Method

Universal Parameters: $\sigma \in (0, 1/2)$ and $\chi \in (0, 1)$.
Input: quadruple $(\tilde{g}, \lambda_0, \underline{W}, \bar{\rho}) \in (\mathcal{C}^1(\Delta^n; L_g), \mathbb{R}_{++}, \bar{B}_1, \mathbb{R}_{++})$.

0. set $W_0 = \underline{W}$, $j = 1$, and

$$\lambda = \lambda_0, \quad \bar{M}_0 = 1; \tag{18}$$

1. choose $\underline{\mathrm{M}}_j \in [1, \bar{M}_{j-1}]$ and call the ADAP-FISTA method in Appendix B with universal input (σ, χ) and inputs

$$x_0 = W_{j-1}, \quad (\mu, L_0) = (1/2, \underline{\mathrm{M}}_j), \tag{19}$$

$$\psi_s = \lambda \tilde{g} + \frac{1}{2}\|\cdot - W_{j-1}\|_F^2, \quad \psi_n = \lambda \delta_{\bar{B}_1}; \tag{20}$$

2. if ADAP-FISTA fails or its output (W, V, L) (if it succeeds) does not satisfy the inequality

$$\lambda \tilde{g}(W_{j-1}) - \left[\lambda \tilde{g}(W) + \frac{1}{2}\|W - W_{j-1}\|_F^2 \right] \geq V \bullet (W_{j-1} - W), \tag{21}$$

then set $\lambda = \lambda/2$ and go to step 1; else, set $(\lambda_j, \bar{M}_j) = (\lambda, L)$, $(W_j, V_j) = (W, V)$, and

$$R_j := \frac{V_j + W_{j-1} - W_j}{\lambda_j} \tag{22}$$

and go to step 3;

3. if $\|R_j\|_F \leq \bar{\rho}$, then stop with success and output $(\overline{W}, \overline{R}) = (W_j, R_j)$; else, go to step 4;

4. set $j \leftarrow j + 1$ and go to step 1.

Several remarks about ADAP-AIPP are now given. First, at each iteration, steps 1 and 2 successively call the ADAP-FISTA method with inputs given by (19) and (20) to obtain a prox stepsize $\lambda_j \leq \lambda_{j-1}$ and a pair (W_j, V_j) satisfying (21) and

$$\|V_j\|_F \leq \sigma \|W_j - W_{j-1}\|_F, \quad V_j \in \lambda_j \left[\nabla \tilde{g}(W_j) + \partial \delta_{\bar{B}_1}(W_j)\right] + (W_j - W_{j-1}) \tag{23}$$

where σ is part of the input of ADAP-AIPP. Such a pair (W_j, V_j) can be viewed as an approximate stationary solution of prox subproblem (17) with $\lambda = \lambda_j$, where the residual V_j is relaxed from being zero to a quantity that is now relatively bounded as in (23). Second, it follows immediately from the inclusion in relation (23) and the definition of R_j in (22) that the pair (W_j, R_j) computed in step 2 of ADAP-AIPP satisfies the inclusion $R_j \in \nabla \tilde{g}(W_j) + \partial \delta_{\bar{B}_1}(W_j)$ for every iteration $j \geq 1$. As a consequence, if ADAP-AIPP terminates in step 3, then the pair $(\overline{W}, \overline{R}) = (W_j, R_j)$ output by this step is a $\bar{\rho}$-approximate stationary solution of (15), i.e., it satisfies

$$\overline{R} \in \nabla \tilde{g}(\overline{W}) + \partial \delta_{\bar{B}_1}(\overline{W}), \quad \|\overline{R}\|_F \leq \bar{\rho}. \tag{24}$$

Finally, it is interesting to note that ADAP-AIPP is a universal method in that it requires no knowledge of any parameters (such as objective function curvatures) underlying problem (15).

Before stating the main complexity result of the ADAP-AIPP method, the following quantities are introduced

$$\bar{G} := \sup\{\|\nabla g(UU^T)\|_F : U \in \bar{B}_3\}, \quad L_{\tilde{g}} := 2\bar{G} + 36L_g, \tag{25}$$

$$\underline{\lambda} := \min\{\lambda_0, 1/(4L_{\tilde{g}})\}, \quad C_\sigma = \frac{2(1-\sigma)^2}{1-2\sigma}, \tag{26}$$

where λ_0 is the initial prox stepsize of ADAP-AIPP and L_g and $\bar{B}_3$ are as in (9) and (16), respectively. Observe that C_σ is well-defined and positive due to the fact that $\sigma \in (0, 1/2)$.

The main complexity result of ADAP-AIPP is stated in the proposition below. Its proof is in Appendix C.

Proposition 2.4. *The following statements about ADAP-AIPP hold:*

(a) ADAP-AIPP terminates with a pair $(\overline{W}, \overline{R})$ that is a $\bar{\rho}$-approximate stationary solution of (15) and its last iteration index l satisfies

$$1 \leq l \leq \mathcal{T} := 1 + \frac{C_\sigma}{\underline{\lambda}\bar{\rho}^2}\left[\tilde{g}(\underline{W}) - \tilde{g}(\overline{W})\right], \tag{27}$$

where $\bar{\rho} > 0$ is an input tolerance, $\underline{W}$ is the initial point, and C_σ and $\underline{\lambda}$ are as in (26);

(b) the total number of ADAP-FISTA calls performed by ADAP-AIPP is no more than

$$\mathcal{T} + \lceil \log_0^+(\lambda_0/\underline{\lambda})/\log 2 \rceil \tag{28}$$

where λ_0 is the initial prox stepsize and $\mathcal{T}$ is as in (27).

Some remarks about Proposition 2.4 are now in order. First, it follows from statement (a) that $\tilde{g}(\overline{W}) \leq \tilde{g}(\underline{W})$. Second, recall that each ADAP-AIPP iteration may perform more than a single ADAP-FISTA call. Statement (b) implies that the total number of ADAP-FISTA calls performed is at most the total number of ADAP-AIPP iterations performed plus a logarithmic term.

2.3 HLR Method

The goal of this subsection is to describe the HLR method for solving problem (8).

An iteration of HLR consists of a call to ADAP-AIPP followed by an FW step (if it does not terminate after the ADAP-AIPP call). Hence, HLR is a hybrid method that combines two types of approaches, i.e., a nonconvex optimization one (ADAP-AIPP) and a convex optimization one (FW method). The formal description of the HLR method is presented below.

HLR Method

Input: A quintuple $(\bar{Y}_0, g, \bar{\epsilon}, \bar{\rho}, \lambda_0) \in \bar{B}_1^{s_0} \times \mathcal{C}^1(\Delta^n; L_g) \times \mathbb{R}^3_{++}$ for some $s_0 \geq 1$.
Output: $\bar{Y} \in \bar{B}_1^s$ for some $s \geq 1$ such that $\bar{Y}\bar{Y}^T$ is an $\bar{\epsilon}$-optimal solution of (8).

0. set $\tilde{Y}_0 = \bar{Y}_0$, $s = s_0$, and $k = 1$.

1. call ADAP-AIPP with quadruple $(g, \lambda_0, \bar{\rho}, \underline{W}) = (g, \lambda_0, \bar{\rho}, \tilde{Y}_{k-1})$ and let $(Y_k, \mathcal{R}_k) \in \bar{B}_1^s \times \mathbb{R}^{n\times s}$ denote its output pair $(\overline{W}, \overline{R})$;

2. compute

$$\theta_k = \max\{-\lambda_{\min}(G_k), 0\}, \qquad y_k = \begin{cases} v_{\min}(G_k) & \text{if } \theta_k > 0, \\ 0 & \text{otherwise,} \end{cases} \tag{29}$$

where

$$G_k := \nabla g(Y_k Y_k^T) \in \mathbb{S}^n \tag{30}$$

and $(\lambda_{\min}(G_k), v_{\min}(G_k)) \in \mathbb{R} \times \mathbb{R}^n$ is a minimum eigenpair of G_k;

3. set

$$\epsilon_k := (G_k Y_k) \bullet Y_k + \theta_k. \tag{31}$$

If

$$\epsilon_k \leq \bar{\epsilon} \tag{32}$$

then **stop** and output pair $(\bar{Y}, \bar{\theta}) = (Y_k, \theta_k)$; else go to step 4;

4. compute

$$\alpha_k = \arg\min_{\alpha} \left\{ g\left(\alpha y_k (y_k)^T + (1-\alpha) Y_k Y_k^T\right) : \alpha \in [0,1] \right\} \tag{33}$$

and set

$$(\tilde{Y}_k, s) = \begin{cases} (y_k, 1) & \text{if } \alpha_k = 1 \\ \left(\left[\sqrt{1-\alpha_k} Y_k, \sqrt{\alpha_k} y_k\right], s+1\right) & \text{otherwise;} \end{cases} \tag{34}$$

5. set $k \leftarrow k+1$ and go to step 1.

Remarks about each of the steps in HLR are now given. First, step 1 of the k-th iteration of HLR calls ADAP-AIPP with initial point $\tilde{Y}_{k-1} \in \bar{B}_1^s$ to produce an iterate $Y_k \in \bar{B}_1^s$ that is an $\bar{\rho}$-approximate stationary solution of nonconvex problem (15). Second, step 2 performs a minimum eigenvector (MEV) computation to compute the quantity θ_k in (29), which is needed for the termination check performed in step 3. Third, the definition of $\theta(\cdot)$ in (10), and relations (30), (29), and (31), imply that

$$\theta_k = \theta(Y_k Y_k^T), \quad \epsilon_k = \nabla g(Y_k Y_k^T) \bullet (Y_k Y_k^T) + \theta(Y_k Y_k^T).$$

Hence, it follows from Lemma 2.1(b) that termination criterion (32) in step 3 is equivalent to checking if $Y_k Y_k^T$ is a $\bar{\epsilon}$-optimal solution of (8). Fourth, if the termination criterion in step 3 is not satisfied, a FW step at $Y_k Y_k^T$ for (8) is taken in step 4 to produce an iterate $\tilde{Y}_k \tilde{Y}_k^T$ as in (33) and (34). The computation of $\tilde{Y}_k$ is entirely performed in the Y-space to avoid forming matrices of the form YY^T, and hence to save storage space. The reason for performing this FW step is to make HLR escape from the spurious near stationary point Y_k of (15) (see the end of the paragraph containing (L_r) in the Introduction). Finally, the quantity s as in (34) keeps track of the column dimension of the most recently generated $\tilde{Y}_k$. It can either increase by one or be set to one after the update (34) is performed.

The complexity of the HLR method is described in the result below whose proof is given in the next subsection.

Theorem 2.5. *The following statements about the HLR method hold:*

(a) the HLR method outputs a point $\bar{Y}$ such that $\bar{Z} = \bar{Y}\bar{Y}^T$ is a $\bar{\epsilon}$-optimal solution of (8) in at most

$$\mathcal{S}(\bar{\epsilon}) := \left\lceil 1 + \frac{4 \max\left\{ g(\bar{Z}_0) - g_*, \sqrt{4L_g(g(\bar{Z}_0) - g_*)}, 4L_g \right\}}{\bar{\epsilon}} \right\rceil \tag{35}$$

iterations (and hence MEV computations) where $\bar{Z}_0 = \bar{Y}_0 \bar{Y}_0^T$, L_g is as in (9), and g_* is the optimal value of (8);

(b) the total number of ADAP-FISTA calls performed by the HLR method is no more than

$$(1 + \mathcal{Q})\mathcal{S}(\bar{\epsilon}) + \frac{C_\sigma \max\{8\bar{G} + 144L_g, 1/\lambda_0\}}{\bar{\rho}^2} \left[g(\bar{Z}_0) - g(\bar{Z}) \right] \tag{36}$$

where λ_0 is given as input to the HLR method, and $\bar{G}$, C_σ, and $\mathcal{Q}$ are as in (25), (26), and (50), respectively.

Two remarks about Theorem 2.5 are now given. First, it follows from statement (a) that HLR performs $\mathcal{O}(1/\bar{\epsilon})$ iterations (and hence MEV computations). Second, statement (b) implies that the total number of ADAP-FISTA calls performed by HLR is $\mathcal{O}\left(1/\bar{\epsilon} + 1/\bar{\rho}^2\right)$ where $\bar{\rho}$ is the input tolerance for each ADAP-AIPP call in step 1 of HLR.

Recall from the Introduction that an iteration of the HALLaR method described in (2a) and (2b) has to approximately solve subproblem (2a), which is accomplished by the HLR method specialized to the case where $g(\cdot) = \mathcal{L}_\beta(\cdot; p)$. In the following, we discuss how to specialize some of the steps of the HLR to this case. Step 1 of HLR calls the ADAP-AIPP method whose steps can be easily implemented if it is known how to project a matrix onto the unit ball $\bar{B}_1$ and how to compute $\nabla \tilde{g}(Y)$ where $\tilde{g}(Y)$ is as in (15). Projecting a matrix onto the unit ball is easy as it just involves dividing the matrix by its norm. Define the quantities

$$q(Y; p) := p + \beta(\mathcal{A}(YY^T) - b), \quad \tilde{\theta}(Y; p) := \max\{-\lambda_{\min}[C + \mathcal{A}^*(q(Y; p))], 0\}. \tag{37}$$

When $g(\cdot) = \mathcal{L}_\beta(\cdot; p)$, the matrix $\nabla \tilde{g}(Y)$ can be explicitly computed as

$$\nabla \tilde{g}(Y) = 2\nabla g(YY^T)Y, \quad \nabla g(YY^T) = C + \mathcal{A}^*(q(Y; p)). \tag{38}$$

Also, the stepsize α_k in step 4 of HLR has a closed form expression given by

$$\alpha_k = \min\left\{ \frac{CY_k \bullet Y_k + (\mathcal{A}^* q_k) Y_k \bullet Y_k + \tilde{\theta}(Y_k; p)}{\beta \|\mathcal{A}(Y_k Y_k^T) - \mathcal{A}(y_k (y_k)^T)\|^2}, \ 1 \right\}$$

where

$$q_k := q(Y_k; p), \quad y_k := \begin{cases} v_{\min}\left(\nabla g(Y_k Y_k^T)\right) & \text{if } \tilde{\theta}(Y_k; p) > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{39}$$

2.4 Proof of Theorem 2.5

This subsection provides the proof of Theorem 2.5. The following proposition establishes important properties of the iterates generated by HLR.

Proposition 2.6. *For every $k \geq 1$, define*

$$Z_k = Y_k Y_k^T, \quad Z_k^F = y_k y_k^T, \quad D_k := Z_k - Z_k^F, \quad \tilde{Z}_k = \tilde{Y}_k \tilde{Y}_k^T, \tag{40}$$

where y_k is as in (31)*. Then, for every $k \geq 1$, the following relations hold:*

$$Z_k^F \in \arg\min_U \{\ell_g(U; Z_k) : U \in \Delta^n\}; \tag{41}$$

$$\tilde{Z}_k = Z_k - \alpha_k D_k; \tag{42}$$

$$\epsilon_k = G_k \bullet D_k; \tag{43}$$

$$\alpha_k = \arg\min_{\alpha \in [0,1]} g(Z_k - \alpha D_k)\} \tag{44}$$

$$g(Z_{k+1}) \leq g(\tilde{Z}_k) \leq g(Z_k) \tag{45}$$

where ϵ_k and α_k are as in (29) *and* (33)*, respectively. Moreover, for every $k \geq 1$, it holds that*

$$\theta_k \geq 0, \quad G_k + \theta_k I \succeq 0, \quad G_k \bullet Z_k + \theta_k = \epsilon_k. \tag{46}$$

Proof. Relation (41) follows immediately from the way y_k is computed in (31), the definitions of Z_k and Z_k^F in (40), and Proposition A.1 with $G = G_k$ and $(Z^F, \theta^F) = (Z_k^F, \theta_k)$.

To show relation (43), it is first necessary to show that

$$\theta_k = -G_k \bullet Z_k^F. \tag{47}$$

Consider two possible cases. For the first case, suppose that $\theta_k = 0$, which in view of the definitions of y_k and Z_k^F in (29) and (40), respectively implies that $Z_k^F = 0$. Hence, relation (47) immediately follows. For the other case suppose that $\theta_k > 0$ and hence $(\theta_k, y_k) = (-\lambda_{\min}(G_k), v_{\min}(G_k))$. This observation and the fact that $(\lambda_{\min}(G_k), v_{\min}(G_k))$ is an eigenpair of G_k imply that $G_k y_k \bullet y_k = -\theta_k$, which in view of the definition of Z_k^F in (40) immediately implies relation (47).

It is now easy to see that the definition of ϵ_k in (29), relation (47), the update rule for $\tilde{Y}_k$ in (34), and the definitions of Z_k, Z_k^F, and D_k in (40) imply that

$$\begin{aligned} \tilde{Z}_k &= Z_k - \alpha_k D_k \\ \epsilon_k &= G_k \bullet (Z_k - Z_k^F) = G_k \bullet D_k \\ \alpha_k &= \arg\min_{\alpha} \{g(Z_k - \alpha D_k) : \alpha \in [0,1]\}\} \end{aligned}$$

and hence relations (42), (43), and (44) follow.

Relation (27) and the fact that HLR during its k-th iteration calls ADAP-AIPP with initial point $\underline{W} = \tilde{Y}_{k-1}$ and outputs point $\overline{W} = Y_k$ imply that $\tilde{g}(Y_k) \leq \tilde{g}(\tilde{Y}_{k-1})$. This fact together with the definition of $\tilde{g}$ in (15) and the definitions of Z_k and $\tilde{Z}_k$ in (40) imply the first inequality in (45).

Now the first inequality in (45) and relations (41), (42), (43), and (44) imply that $Z_k = Y_k Y_k^T$ and $\tilde{Z}_k = \tilde{Y}_k \tilde{Y}_k^T$ can be viewed as iterates of the k-th iteration of the RFW method of Appendix D. The second inequality in (45) is then an immediate consequence of this observation and the second relation in (138).

The first two relations in (46) follow directly from the definitions of G_k and θ_k in (30) and (29), respectively. The definitions of ϵ_k and Z_k in (31) and (40), respectively, immediately imply the third relation in (46). □

Since step 1 of the HLR method consists of a call to the ADAP-AIPP method developed in Subsection 2.2, the conclusion of Proposition 2.4 applies to this step. The following result, which will be useful in the analysis of the HLR method, translates the conclusion of Proposition 2.4 to the current setting.

Proposition 2.7. *The following statements about step 1 of the k-th iteration of the HLR method hold:*

(a) the ADAP-AIPP call terminates with a pair $(Y_k, \mathcal{R}_k)$ satisfying

$$\mathcal{R}_k \in \nabla \tilde{g}(Y_k) + \partial \delta_{\bar{B}_1}(Y_k), \quad \|\mathcal{R}_k\|_F \leq \bar{\rho}, \tag{48}$$

and the number l_k of ADAP-AIPP iterations performed by the ADAP-AIPP call in step 1 satisfies

$$1 \leq l_k \leq \mathcal{T}_k := 1 + \frac{C_\sigma \max\{8\bar{G} + 144L_g, 1/\lambda_0\}}{\bar{\rho}^2} \left[\tilde{g}(\tilde{Y}_{k-1}) - \tilde{g}(Y_k)\right] \tag{49}$$

where λ_0 and $\bar{\rho}$ are given as input to the HLR method and L_g, $\tilde{g}$, $\bar{G}$, and C_σ are as in (9), (15), (25), and (26) respectively;

(b) the number of ADAP-FISTA calls performed by the ADAP-AIPP call in step 1 is no more than $\mathcal{T}_k + \mathcal{Q}$ where

$$\mathcal{Q} := \left\lceil \log_0^+ \left(\lambda_0 \max\{8\bar{G} + 144L_g, 1/\lambda_0\}\right) / \log 2 \right\rceil. \tag{50}$$

Proof. (a) The first statement is immediate from relation (24) and the fact that ADAP-AIPP outputs pair $(Y_k, \mathcal{R}_k) = (\overline{W}, \overline{R})$. To prove the second statement, suppose that the ADAP-AIPP call made in step 1 terminates after performing l_k iterations. It follows immediately from Proposition 2.4(a) and the fact that the HLR method during its k-th iteration calls ADAP-AIPP with initial point $\underline{W} = \tilde{Y}_{k-1}$ and outputs point $\overline{W} = Y_k$ that l_k satisfies

$$1 \leq l_k \overset{(27)}{\leq} 1 + \frac{C_\sigma}{\underline{\lambda}\bar{\rho}^2} \left[\tilde{g}(\tilde{Y}_{k-1}) - \tilde{g}(Y_k)\right]. \tag{51}$$

The result then follows from the definitions of $\underline{\lambda}$ and $L_{\tilde{g}}$ in (26) and (25), respectively.

(b) The result follows immediately from (a), the fact that the number of times λ is divided by 2 in step 2 of ADAP-AIPP is at most $\lceil \log_0^+(\lambda_0/\underline{\lambda})/\log 2 \rceil$, and the definitions of $\underline{\lambda}$ and $L_{\tilde{g}}$ in (26) and (25), respectively. □

We are now ready to give the proof of Theorem 2.5.

Proof of Theorem 2.5. (a) Consider the matrix $\bar{Z} = \bar{Y}\bar{Y}^T$ where $\bar{Y}$ is the output of the HLR method. The definition of G_k in (30), the fact that the definitions of θ_k and $\theta(\cdot)$ in (29) and (10), respectively, imply that $\theta_k = \theta(Y_k Y_k^T)$, relation (46), and the stopping criterion (32) in step 3 of the HLR method immediately imply that the pair $(\bar{Z}, \theta(\bar{Z}))$ satisfies relation (13) in Lemma 2.1(b) with $\epsilon = \bar{\epsilon}$ and hence $\bar{Z}$ is an $\bar{\epsilon}$-optimal solution of (8). To show that the number of iterations that the HLR method performs to find such an $\bar{\epsilon}$-optimal solution is at most the quantity in (35), observe that Proposition 2.6 establishes that the HLR method generates iterates Y_k and $\tilde{Y}_k$ during its k-th iteration such that $Z_k = Y_k Y_k^T$ and $\tilde{Z}_k = \tilde{Y}_k \tilde{Y}_k^T$ can be viewed as iterates of the k-th iteration of the RFW method of Appendix D. The result then immediately follows from this observation, Theorem D.1, and the fact that the diameter of Δ^n is at most 2.

(b) Suppose that the HLR method terminates at an iteration index $\bar{K}$. It follows from relations (50) and (35) and the definition of $\mathcal{T}_k$ in (49) that the HLR method performs at most

$$(1 + \mathcal{Q})\mathcal{S}(\bar{\epsilon}) + \frac{C_\sigma \max\{8\bar{G} + 144 L_g, 1/\lambda_0\}}{\bar{\rho}^2} \sum_{k=1}^{\bar{K}} \left[\tilde{g}(\tilde{Y}_{k-1}) - \tilde{g}(Y_k)\right] \tag{52}$$

ADAP-FISTA calls. Now using the fact that $\tilde{g}(\tilde{Y}_k) \leq \tilde{g}(Y_k)$, it is easy to see that the last term in (52) is summable:

$$\begin{aligned}\sum_{k=1}^{\bar{K}} \tilde{g}(\tilde{Y}_{k-1}) - \tilde{g}(Y_k) &= \tilde{g}(\tilde{Y}_{\bar{K}-1}) - \tilde{g}(Y_{\bar{K}}) + \sum_{k=1}^{\bar{K}-1} \tilde{g}(\tilde{Y}_{k-1}) - \tilde{g}(Y_k) \\ &\leq \tilde{g}(\tilde{Y}_{\bar{K}-1}) - \tilde{g}(Y_{\bar{K}}) + \sum_{k=1}^{\bar{K}-1} \tilde{g}(\tilde{Y}_{k-1}) - \tilde{g}(\tilde{Y}_k) = \tilde{g}(\tilde{Y}_0) - \tilde{g}(Y_{\bar{K}}).\end{aligned} \tag{53}$$

The result then follows from relations (52) and (53), the facts that $Y_{\bar{K}} = \bar{Y}$, $\bar{Z} = \bar{Y}\bar{Y}^T$, $\tilde{Y}_0 = \bar{Y}_0$, $\bar{Z}_0 = \bar{Y}_0 \bar{Y}_0^T$, and the definition of $\tilde{g}$ in (15). □

3 HALLaR

This section presents an inexact AL method for solving the pair of primal-dual SDPs (P) and (D), namely, HALLaR, whose outline is given in the introduction. It contains two subsections. Subsection 3.1 formally states HALLaR and presents its main complexity result, namely Theorem 3.2. Subsection 3.2 is devoted to the proof of Theorem 3.2.

Throughout this section, it is assumed that (P) and (D) have optimal solutions X_* and (p_*, θ_*), respectively, and that both (P) and (D) have the same optimal value. It is well-known that such an assumption is equivalent to the existence of a triple (X_*, p_*, θ_*) satisfying the optimality conditions:

$$\begin{aligned}&\text{(primal feasibility)} && \mathcal{A}(X_*) - b = 0, \quad \operatorname{tr}(X_*) \leq 1, \quad X_* \succeq 0, \\ &\text{(dual feasibility)} && S_* := C + \mathcal{A}^* p_* + \theta_* I \succeq 0, \quad \theta_* \geq 0, \\ &\text{(complementarity)} && \langle X_*, S_* \rangle = 0, \quad \theta_*(1 - \operatorname{tr} X_*) = 0.\end{aligned} \tag{54}$$

This section studies the complexity of HALLaR for finding an $(\epsilon_{\rm p}, \epsilon_{\rm c})$-solution of (54), i.e., a triple $(\bar{X}, \bar{p}, \bar{\theta})$ that satisfies

$$\begin{aligned}&\text{($\epsilon_{\rm p}$-primal feasibility)} && \|\mathcal{A}(\bar{X}) - b\| \leq \epsilon_{\rm p}, \quad \operatorname{tr}(\bar{X}) \leq 1, \quad \bar{X} \succeq 0, \\ &\text{(dual feasibility)} && \bar{S} := C + \mathcal{A}^* \bar{p} + \bar{\theta} I \succeq 0, \quad \bar{\theta} \geq 0, \\ &\text{($\epsilon_{\rm c}$-complementarity)} && \langle \bar{X}, S \rangle + \bar{\theta}(1 - \operatorname{tr} \bar{X}) \leq \epsilon_{\rm c}.\end{aligned} \tag{55}$$

3.1 Description of HALLaR and Main Theorem

The formal description of HALLaR is presented next.

HALLaR Method

Input: Initial points $(U_0, p_0) \in \bar{B}_1^{s_0} \times \mathbb{R}^m$, tolerance pair $(\epsilon_{\rm c}, \epsilon_{\rm p}) \in \mathbb{R}^2_{++}$, penalty parameter $\beta \in \mathbb{R}_{++}$, and ADAP-AIPP parameters $(\bar{\rho}, \lambda_0) \in \mathbb{R}^2_{++}$.
Output: $(\bar{X}, \bar{p}, \bar{\theta}) \in \Delta^n \times \mathbb{R}^m \times \mathbb{R}_+$, an $(\epsilon_{\rm p}, \epsilon_{\rm c})$-solution of (54)

0. set $t = 1$ and

$$\bar{\epsilon} = \min\{\epsilon_{\rm c},\ \epsilon_{\rm p}^2\beta/6\}; \tag{56}$$

1. call HLR with input $(g, \bar{Y}_0, \lambda_0, \bar{\epsilon}, \bar{\rho}) = (\mathcal{L}_\beta(\cdot; p_{t-1}), U_{t-1}, \lambda_0, \bar{\epsilon}, \bar{\rho})$ where $U_{t-1} \in \bar{B}_1^{s_{t-1}}$, and let $U_t \in \bar{B}_1^{s_t}$ denote its output $\bar{Y}$;

2. set

$$p_t = p_{t-1} + \beta(\mathcal{A}(U_t U_t^T) - b); \tag{57}$$

3. if $\|\mathcal{A}(U_t U_t^T) - b\| \le \epsilon_{\rm p}$, then set $T = t$ and **return** $(\bar{X}, \bar{p}, \bar{\theta}) = (U_T U_T^T, p_T, \tilde{\theta}(U_T U_T^T; p_{T-1}))$ where $\tilde{\theta}(\cdot;\cdot)$ is as in (37);

4. set $t = t + 1$ and **go to** step **1.**

Some remarks about each of the steps in HALLaR are now given. First, step 1 invokes HLR to obtain an $\bar{\epsilon}$-optimal solution $U_t U_t^T$ of subproblem (2a) using the previous $U_{t-1} \in \bar{B}_1^{s_{t-1}}$ as initial point. Second, step 2 updates the multiplier p_t according to a full Lagrange multiplier update. Third, it is shown in Lemma 3.1 below that the triple $(U_t U_t^T, p_t, \tilde{\theta}(U_t U_t^T; p_{t-1}))$ always satisfies the dual feasibility and $\epsilon_{\rm c}$-complementarity conditions in (55) where $\tilde{\theta}(\cdot;\cdot)$ is as in (37). Finally, step 3 checks if $U_t U_t^T$ is an $\epsilon_{\rm p}$-primal feasible solution. It then follows from the above remarks that if this condition is satisfied, then the triple $(U_t U_t^T, p_t, \tilde{\theta}(U_t U_t^T; p_{t-1}))$ is an $(\epsilon_{\rm p}, \epsilon_{\rm c})$-solution of (55).

Lemma 3.1. *For every iteration index t, the triple $(U_t U_t^T, p_t, \tilde{\theta}(U_t U_t^T; p_{t-1}))$ satisfies the dual feasibility and $\epsilon_{\rm c}$-complementarity conditions in* (55) *where $\tilde{\theta}(\cdot;\cdot)$ is as in* (37)*.*

Proof. The definitions of $q(\cdot;\cdot)$ and $\tilde{\theta}(\cdot;\cdot)$ in (37), the second relation in (38) with $Y = U_t$ and $p = p_{t-1}$, and the update rule for p_t in (57) imply that

$$\nabla\mathcal{L}_\beta(U_t U_t^T; p_{t-1}) = C + \mathcal{A}^* p_t, \quad \tilde{\theta}(U_t U_t^T; p_{t-1}) = \max\{-\lambda_{\min}(C + \mathcal{A}^* p_t), 0\}.$$

It is then easy to see from the above relation that the triple $(U_t U_t^T, p_t, \tilde{\theta}(U_t U_t^T; p_{t-1}))$ always satisfies the dual feasibility condition in (55).

The fact that the definition of $\bar{\epsilon}$ in (56) implies that $\bar{\epsilon} \le \epsilon_{\rm c}$, the fact that $U_t U_t^T$ is an $\bar{\epsilon}$ solution of (2a), and the formula for $\nabla\mathcal{L}_\beta(U_t U_t^T; p_{t-1})$ above imply that

$$0 \in C + \mathcal{A}^* p_t + \partial_{\epsilon_{\rm c}}\delta_{\Delta^n}(U_t U_t^T).$$

It then follows immediately from the above inclusion and relation (14) with $Z = U_t U_t^T$, $g = \mathcal{L}_\beta(\cdot; p_{t-1})$, and $\theta(\cdot) = \tilde{\theta}(\cdot; p_{t-1})$ that the triple $(U_t U_t^T, p_t, \tilde{\theta}(U_t U_t^T; p_{t-1}))$ always satisfies the $\epsilon_{\rm c}$-complementarity condition in (55). □

Before stating the complexity of HALLaR, the following quantities are first introduced:

$$\bar{\mathcal{F}}_\beta := \frac{\beta}{2}\|\mathcal{A}X_0 - b\|^2 + \frac{5\|p_*\|^2 + \|p_*\|\sqrt{3\|p_*\|^2 + 2\beta\bar{\epsilon}}}{\beta} + 3\bar{\epsilon} \tag{58}$$

$$\bar{\mathcal{G}}_\beta := \|C\|_F + \|\mathcal{A}\|\left(9\beta\|\mathcal{A}\| + \sqrt{4\|p_*\|^2 + 2\beta\bar{\epsilon}} + \beta\|b\|\right) \tag{59}$$

$$\bar{\kappa}_\beta := \left\lceil \log_0^+\left(\lambda_0 \max\{8\bar{\mathcal{G}}_\beta + 144\beta\|A\|^2, 1/\lambda_0\}\right)/\log 2\right\rceil \tag{60}$$

where p_* is an optimal dual solution and $\bar{\epsilon}$ is as in (56).

The main result of this paper is now stated.

Theorem 3.2. *The following statements hold:*

a) HALLaR terminates with an $(\epsilon_{\rm p}, \epsilon_{\rm c})$-solution $(\bar{X}, \bar{p}, \bar{\theta})$ of the pair of SDPs (P) *and* (D) *in at most*

$$\mathcal{J} := \left\lceil \frac{3\|p_*\|^2}{\beta^2\epsilon_{\rm p}^2} \right\rceil \tag{61}$$

iterations where p_* is an optimal solution to (D)*;*

(b) HALLaR performs at most

$$\mathcal{J} \cdot \bar{\mathcal{P}}_\beta(\epsilon_{\rm p}, \epsilon_{\rm c}) \tag{62}$$

and

$$\mathcal{J}\left[(1+\bar{\kappa}_\beta)\bar{\mathcal{P}}_\beta(\epsilon_{\rm p}, \epsilon_{\rm c})\right] + \frac{C_\sigma\bar{\mathcal{F}}_\beta \max\{8\bar{\mathcal{G}}_\beta + 144\beta\|A\|^2, 1/\lambda_0\}}{\bar{\rho}^2} \tag{63}$$

total HLR iterations (and hence MEV computations) and total ADAP-FISTA calls, respectively, where

$$\bar{\mathcal{P}}_\beta(\epsilon_{\rm p}, \epsilon_{\rm c}) := \left\lceil 1 + \frac{4\max\left\{\bar{\mathcal{F}}_\beta, \sqrt{4\beta\|A\|^2\bar{\mathcal{F}}_\beta}, 4\beta\|A\|^2\right\}}{\min\{\epsilon_{\rm c},\ \epsilon_{\rm p}^2\beta/6\}} \right\rceil, \tag{64}$$

$\bar{\rho}$ is an input parameter to HALLaR, and $\bar{\mathcal{F}}_\beta$, C_σ, $\bar{\mathcal{G}}_\beta$, and $\bar{\kappa}_\beta$ are as in (58), (25), (59), *and* (60)*, respectively.*

Two remarks about Theorem 3.2 are now given. First, it follows from statement (a) that HALLaR performs $\mathcal{O}\left(1/(\beta^2\epsilon_p^2)\right)$ iterations. Second, statement (b) and the definitions of $\mathcal{J}$, $\bar{\mathcal{P}}_\beta(\epsilon_{\rm p}, \epsilon_{\rm c})$, and $\bar{\mathcal{F}}_\beta$ in (61), (64), and (58), respectively imply that HALLaR performs

$$\mathcal{O}\left(\frac{1}{\beta\epsilon_p^2 \min\{\epsilon_{\rm c}, \epsilon_p^2\beta\}}\right) \tag{65}$$

and

$$\mathcal{O}\left(\frac{1}{\beta\epsilon_p^2 \min\{\epsilon_{\rm c}, \epsilon_p^2\beta\}} + \frac{\beta^2}{\bar{\rho}^2}\right) \tag{66}$$

total HLR iterations (and hence MEV computations) and ADAP-FISTA calls, respectively. In contrast to the case where $\beta = \mathcal{O}(1)$, the result below shows that the bounds (65) and (66) can be improved when $\beta = \mathcal{O}(1/\epsilon_{\rm p})$.

Corollary 3.3. *If $\beta = \mathcal{O}\left(1/\epsilon_{\rm p}\right)$ and $\bar{\rho} = \beta\min\{\epsilon_{\rm c}, \epsilon_p^2\beta/6\}$, then HALLaR performs at most*

$$\mathcal{O}\left(\frac{1}{\epsilon_{\rm p}^2} + \frac{1}{\epsilon_{\rm c}^2}\right) \tag{67}$$

total HLR iterations (and hence MEV computations) and total ADAP-FISTA calls.

Proof. The conclusion of the corollary immediately follows from relations (65) and (66) together with the assumptions that $\beta = \mathcal{O}(1/\epsilon_{\rm p})$ and $\bar{\rho} = \beta \min\{\epsilon_{\rm c}, \epsilon_p^2 \beta/6\}$. □

It can be shown that each ADAP-FISTA call performs at most

$$\mathcal{O}_1\left(\sqrt{2\left[1+\lambda_0(2\bar{\mathcal{G}}_\beta + 36\beta\|\mathcal{A}\|^2)\right]}\log_1^+\left(1+\lambda_0\left[2\bar{\mathcal{G}}_\beta + 36\beta\|\mathcal{A}\|^2\right]\right)\right) \tag{68}$$

iterations/resolvent evaluations.[1] This is an immediate consequence of Lemma C.2(a) in Appendix C, the definition of $L_{\tilde{g}}$ in (25), the fact that $\mathcal{L}_\beta(X; p_{t-1})$ is $(\beta\|\mathcal{A}\|^2)$-smooth, and Lemma 3.7 which is developed in the next subsection.

3.2 Proof of Theorem 3.2

Since HALLaR calls the HLR method at every iteration, the next proposition specializes Theorem 2.5, which states the complexity of HLR, to the specific case of SDPs. Its statement uses the following quantities associated with an iteration t of HALLaR:

$$\mathcal{F}_\beta^{(t)} := \mathcal{L}_\beta(X_{t-1}, p_{t-1}) - \min_{X\in\Delta^n} \mathcal{L}_\beta(X, p_{t-1}), \tag{69}$$

$$\mathcal{G}_\beta^{(t)} := \sup\{\|\nabla\mathcal{L}_\beta(UU^T, p_{t-1})\| : U \in \bar{B}_3^{s_{t-1}}\}, \tag{70}$$

$$\kappa_\beta^{(t)} = \left\lceil \log_0^+\left(\lambda_0 \max\{8\mathcal{G}_\beta^{(t)} + 144\beta\|A\|^2, 1/\lambda_0\}\right)/\log 2\right\rceil, \tag{71}$$

Proposition 3.4. *The following statements about the HLR call in step 1 of the t-th iteration of the HALLaR hold:*

(a) it outputs U_t such that $X_t = U_t U_t^T$ is an $\bar{\epsilon}$-optimal solution of

$$\min_{X\in\Delta^n} \mathcal{L}_\beta(X; p_{t-1})$$

by performing at most

$$\mathcal{P}_\beta^{(t)}(\bar{\epsilon}) := \left\lceil 1 + \frac{4\max\left\{\mathcal{F}_\beta^{(t)}, \sqrt{4\beta\|\mathcal{A}\|^2\mathcal{F}_\beta^{(t)}}, 4\beta\|\mathcal{A}\|^2\right\}}{\bar{\epsilon}}\right\rceil \tag{72}$$

iterations (and hence MEV computations) where $\mathcal{F}_\beta^{(t)}$ and $\bar{\epsilon}$ are as in (69) and (56), respectively;

(b) the total number of ADAP-FISTA calls within such call is no more than

$$(1+\kappa_\beta^{(t)})\mathcal{P}_\beta^{(t)}(\bar{\epsilon}) + \frac{C_\sigma \max\{8\mathcal{G}_\beta^{(t)} + 144\beta\|\mathcal{A}\|^2, 1/\lambda_0\}}{\bar{\rho}^2}\left[\mathcal{L}_\beta(X_{t-1}, p_{t-1}) - \mathcal{L}_\beta(X_t, p_{t-1})\right] \tag{73}$$

where λ_0 is given as input to HLR, and C_σ, $\mathcal{G}_\beta^{(t)}$, and $\kappa_\beta^{(t)}$ are as in (26), (70), and (71), respectively.

[1] A resolvent evaluation of h is an evaluation of $(I+\gamma\partial h)^{-1}(\cdot)$ for some $\gamma > 0$.

Proof. (a) Recall that each HLR call during the t-th iteration of HALLaR is made with $(g, \bar{Y}_0, \lambda_0, \bar{\epsilon}, \bar{\rho}) = (\mathcal{L}_\beta(\cdot; p_{t-1}), U_{t-1}, \lambda_0, \bar{\epsilon}, \bar{\rho})$. The result then immediately follows from Theorem 2.5(a), the definition of $\mathcal{F}_\beta^{(t)}$ in (69), and the fact that $\mathcal{L}_\beta(X; p_{t-1})$ is $(\beta\|\mathcal{A}\|^2)$-smooth.

(b) The proof follows directly from Theorem 2.5(b), statement(a), the fact that $\mathcal{L}_\beta(\cdot; p_{t-1})$ is $(\beta\|\mathcal{A}\|^2)$-smooth, and the definitions of $\mathcal{G}_\beta^{(t)}$ and $\kappa_\beta^{(t)}$ in (70) and (71), respectively. □

The following Lemma establishes key bounds which will be used later to bound quantities $\mathcal{F}_\beta^{(t)}$, $\mathcal{G}_\beta^{(t)}$, and $\kappa_\beta^{(t)}$ that appear in Proposition 3.4.

Lemma 3.5. *The following relations hold:*

$$\max_{t\in\{0,\dots T\}} \|p_t\| \leq \|p_*\| + \sqrt{3\|p_*\|^2 + 2\beta\bar{\epsilon}}, \tag{74}$$

$$\beta \sum_{t=1}^{T} \|\mathcal{A}X_t - b\|^2 \leq \frac{3\|p_*\|^2}{\beta} + 2\bar{\epsilon}, \tag{75}$$

where T is the last iteration index of HALLaR, p_* is an optimal Lagrange multiplier, and $\bar{\epsilon}$ is as in (56).

Proof. It follows immediately from Proposition 3.4(a) and the definition of $\bar{\epsilon}$-optimal solution that the HLR call made in step 1 of the t-th iteration of HALLaR outputs U_t such that $X_t = U_t U_t^T$ satisfies

$$0 \in \nabla\mathcal{L}_\beta(X_t; p_{t-1}) + \partial_{\bar{\epsilon}}\delta_{\Delta^n}. \tag{76}$$

It is then easy to see that HALLaR is an instance of the AL Framework in Appendix E since the HLR method implements relation (152) in the Blackbox AL with $(\hat{\epsilon}_c, \hat{\epsilon}_d) = (\bar{\epsilon}, 0)$. The proof of relations (74) and (75) now follows immediately from relations (157) and (158) and the fact that $p_0 = 0$. □

Lemma 3.6. *For every iteration index t of HALLaR, the following relations hold:*

$$\mathcal{F}_\beta^{(t)} \leq \bar{\mathcal{F}}_\beta \tag{77}$$

$$\sum_{l=1}^{t} [\mathcal{L}_\beta(X_{l-1}, p_{l-1}) - \mathcal{L}_\beta(X_l, p_{l-1})] \leq \bar{\mathcal{F}}_\beta \tag{78}$$

where $X_t = U_t U_t^T$ and $\mathcal{F}_\beta^{(t)}$ and $\bar{\mathcal{F}}_\beta$ are as in (69) *and* (58)*, respectively.*

Proof. Let t be an iteration index. We first show that

$$\mathcal{L}_\beta(X_{t-1}, p_{t-1}) \leq \lambda_{\min}(C) + \frac{\beta}{2}\|\mathcal{A}X_0 - b\|^2 + \frac{3\|p_*\|^2}{\beta} + 2\bar{\epsilon}. \tag{79}$$

holds. It follows immediately from the definition of $\mathcal{L}_\beta(X; p)$ in (3), the fact that $X_0 = \arg\min_{X\in\Delta^n} C\bullet X$, the Cauchy-Schwarz inequality, and the fact that $p_0 = 0$, that

$$\mathcal{L}_\beta(X_0, p_0) \overset{(3)}{\leq} C \bullet X_0 + \frac{\beta}{2}\|\mathcal{A}(X_0) - b\|^2 = \lambda_{\min}(C) + \frac{\beta}{2}\|\mathcal{A}(X_0) - b\|^2. \tag{80}$$

Hence, relation (79) holds with $t = 1$. Suppose now that $t \geq 2$ and let l be an iteration index such that $l < t$. Relation (45), the fact that HALLaR during its l-th iteration calls the HLR method

with input $g = \mathcal{L}_\beta(X, p_{l-1})$ and initial point $\bar{Y}_0 = U_{l-1}$, and the fact that $X_l = U_l U_l^T$ imply that $\mathcal{L}_\beta(X_l, p_{l-1}) \leq \mathcal{L}_\beta(X_{l-1}, p_{l-1})$ and hence that

$$\mathcal{L}_\beta(X_l, p_l) - \mathcal{L}_\beta(X_{l-1}, p_{l-1}) \leq \mathcal{L}_\beta(X_l, p_l) - \mathcal{L}_\beta(X_l, p_{l-1}) \overset{(57),(3)}{=} \beta \|\mathcal{A}(X_l) - b\|^2 \tag{81}$$

in view of the update rule (57) and the definition of $\mathcal{L}_\beta(\cdot;\cdot)$ in (3). Summing relation (81) from $l = 1$ to $t-1$ and using relation (75) gives

$$\mathcal{L}_\beta(X_{t-1}, p_{t-1}) - \mathcal{L}_\beta(X_0, p_0) \overset{(81)}{\leq} \beta \sum_{l=1}^{t-1} \|\mathcal{A}(X_l) - b\|^2 \overset{(75)}{\leq} \frac{3\|p_*\|^2}{\beta} + 2\bar{\epsilon}. \tag{82}$$

Relation (79) now follows by combining relations (80) and (82).

Now, relation (74), the fact that $\min_{X \in \Delta^n} C \bullet X = \lambda_{\min}(C)$, and the definition of $\mathcal{L}_\beta(\cdot;\cdot)$ in (3), imply that for any $t = 0, \ldots, T$,

$$\begin{aligned} \mathcal{L}_\beta(X, p_t) - \lambda_{\min}(C) \geq \mathcal{L}_\beta(X, p_t) - C \bullet X &= \frac{1}{2} \left\| \frac{p_t}{\sqrt{\beta}} + \sqrt{\beta}(AX - b) \right\|^2 - \frac{\|p_t\|^2}{2\beta} \\ &\overset{(74)}{\geq} -\frac{2\|p_*\|^2 + \beta\bar{\epsilon} + \|p_*\|\sqrt{3\|p_*\|^2 + 2\beta\bar{\epsilon}}}{\beta} \qquad \forall X \in \Delta^n \end{aligned} \tag{83}$$

where T is the last iteration index of HALLaR. Relations (79) and (83) together with the definition of $\bar{\mathcal{F}}_\beta$ in (58) then imply that $\mathcal{L}_\beta(X_{t-1}, p_{t-1}) - \mathcal{L}_\beta(X, p_{t-1}) \leq \bar{\mathcal{F}}_\beta$ for every $X \in \Delta^n$ and iteration index t. Relation (77) then follows immediately from this conclusion and the definition of $\mathcal{F}_\beta^{(t)}$ in (69).

To show relation (78), observe that relations (81) and (75) imply that for any iteration index t the following relations hold:

$$\begin{aligned} &\sum_{l=1}^{t} \mathcal{L}_\beta(X_{l-1}, p_{l-1}) - \mathcal{L}_\beta(X_l, p_{l-1}) \\ &= \sum_{l=1}^{t} [\mathcal{L}_\beta(X_{l-1}, p_{l-1}) - \mathcal{L}_\beta(X_l, p_l)] + \sum_{l=1}^{t} [\mathcal{L}_\beta(X_l, p_l) - \mathcal{L}_\beta(X_l, p_{l-1})] \\ &\overset{(81)}{=} \mathcal{L}_\beta(X_0, p_0) - \mathcal{L}_\beta(X_t, p_t) + \beta \sum_{l=1}^{t} \|\mathcal{A}(X_l) - b\|^2 \overset{(75)}{\leq} \mathcal{L}_\beta(X_0, p_0) - \mathcal{L}_\beta(X_t, p_t) + \frac{3\|p_*\|^2}{\beta} + 2\bar{\epsilon}. \end{aligned}$$

Relation (78) then follows immediately from the above relation, relation (83) with $X = X_t$, relation (80), and the definition of $\bar{\mathcal{F}}_\beta$ in (58). □

Lemma 3.7. *For every iteration t of HALLaR, we have $\mathcal{G}_\beta^{(t)} \leq \bar{\mathcal{G}}_\beta$ where $\mathcal{G}_\beta^{(t)}$ and $\bar{\mathcal{G}}_\beta$ are as in* (70) *and* (59)*, respectively.*

Proof. Let t be an iteration index of HALLaR and suppose that $U \in \bar{B}_3^{s_{t-1}}$. It is easy to see from the definition of $\mathcal{L}_\beta(\cdot;\cdot)$ in (3) that

$$\nabla \mathcal{L}_\beta(UU^T; p_{t-1}) = C + \mathcal{A}^* p_{t-1} + \beta \mathcal{A}^*(\mathcal{A}(UU^T) - b). \tag{84}$$

It then follows from the fact that $U \in \bar{B}_3^{s_{t-1}}$, Cauchy-Schwarz inequality, triangle inequality, relation (84), and bound (74) that

$$\begin{aligned}\|\nabla \mathcal{L}_\beta(UU^T; p_{t-1})\|_F &\overset{(84)}{=} \|C + \mathcal{A}^* p_{t-1} + \beta \mathcal{A}^*(\mathcal{A}(UU^T) - b)\|_F \\ &\leq \|C\|_F + \|\mathcal{A}\|\|p_{t-1}\| + \beta\|\mathcal{A}\|^2\|UU^T\|_F + \beta\|\mathcal{A}\|\,\|b\| \\ &\leq \|C\|_F + \|\mathcal{A}\|\|p_{t-1}\| + 9\beta\|\mathcal{A}\|^2 + \beta\|\mathcal{A}\|\,\|b\| \\ &\overset{(74)}{\leq} \|C\|_F + \|\mathcal{A}\|\left(9\beta\|\mathcal{A}\| + \|p_*\| + \sqrt{3\|p_*\|^2 + 2\beta\bar{\epsilon}} + \beta\|b\|\right)\end{aligned}$$

which immediately implies the result of the lemma in view of the definitions of $\mathcal{G}_\beta^{(t)}$ and $\bar{\mathcal{G}}_\beta$ in (70) and (59), respectively. □

We are now ready to prove Theorem 3.2.

Proof of Theorem 3.2. (a) It follows immediately from Lemma 3.1 that output $(\bar{X}, \bar{p}, \bar{\theta})$ satisfies the the dual feasibility and $\epsilon_{\rm c}$-complementarity conditions in (55). It then remains to show that the triple $(\bar{X}, \bar{p}, \bar{\theta})$ satisfies the $\epsilon_{\rm p}$-primal feasibility condition in (55) in at most $\mathcal{J}$ iterations where $\mathcal{J}$ is as in (61). To show this, it suffices to show that HALLaR is an instance of the AL framework analyzed in Appendix E. Observe first that (P) is a special case of (148) with $f(X) = C \bullet X$ and $h(X) = \delta_{\Delta^n}(X)$. It is also easy to see that the call to the HLR in step 1 of HALLaR is a special way of implementing step 1 of the AL framework, i.e., the Blackbox AL. Indeed, Proposition 3.4(a) implies that the output U_t of HLR satisfies $0 \in \nabla \mathcal{L}_\beta(U_t U_t^T; p_{t-1}) + \partial_{\bar{\epsilon}} \delta_{\Delta^n}(U_t U_t^T)$ and hence HLR implements relation (152) in the Blackbox AL with $(\hat{X}, \hat{R}) = (U_t U_t^T, 0)$, $(\hat{\epsilon}_{\rm c}, \hat{\epsilon}_{\rm d}) = (\bar{\epsilon}, 0)$, $g = \mathcal{L}_\beta(\cdot; p_{t-1})$, and $h = \delta_{\Delta^n}(\cdot)$.

In view of the facts that HALLaR is an instance of the AL framework and $p_0 = 0$, it then follows immediately from Theorem E.1 that HALLaR terminates within the number of iterations in (61) and that the output $(\bar{X}, \bar{p}, \bar{\theta})$ satisfies the $\epsilon_{\rm p}$-primal feasibility condition in (55).

(b) Consider the quantity $\mathcal{P}_\beta^{(t)}(\bar{\epsilon})$ as in (72) where t is an iteration index of HALLaR. It follows immediately from relations (56) and (77) and the definition of $\bar{\mathcal{P}}_\beta(\epsilon_{\rm p}, \epsilon_{\rm c})$ in (64) that $\mathcal{P}_\beta^{(t)}(\bar{\epsilon}) \leq \bar{\mathcal{P}}_\beta(\epsilon_{\rm p}, \epsilon_{\rm c})$. Hence, it follows from Proposition 3.4(a) that each HLR call made in step 1 of HALLaR performs at most $\bar{\mathcal{P}}_\beta(\epsilon_{\rm p}, \epsilon_{\rm c})$ iterations/MEV computations. The result then follows from this conclusion and part (a).

(c) Let t be an iteration index of HALLaR. Lemma 3.7 implies that $\mathcal{G}_\beta^{(t)} \leq \bar{\mathcal{G}}_\beta$ and $\kappa_\beta^{(t)} \leq \bar{\kappa}_\beta$ in view of the definitions of $\kappa_\beta^{(t)}$ and $\bar{\kappa}_\beta$ in (71) and (60), respectively. Hence, it follows from this conclusion, the fact that $\mathcal{P}_\beta^{(t)}(\bar{\epsilon}) \leq \bar{\mathcal{P}}_\beta(\epsilon_{\rm p}, \epsilon_{\rm c})$, Proposition 3.4(b), and part (a) that the total number of ADAP-FISTA calls performed by HALLaR is at most

$$\mathcal{J}\left[(1 + \bar{\kappa}_\beta)\right]\bar{\mathcal{P}}_\beta(\epsilon_{\rm p}, \epsilon_{\rm c}) + \frac{C_\sigma \max\{8\bar{\mathcal{G}}_\beta + 144\beta\|\mathcal{A}\|^2, 1/\lambda_0\}}{\bar{\rho}^2}\left[\sum_{t=1}^{T} \mathcal{L}_\beta(X_{t-1}, p_{t-1}) - \mathcal{L}_\beta(X_t, p_{t-1})\right]$$

where $\mathcal{J}$ is as in (61) and T is the last iteration index of HALLaR. The result in (c) then follows immediately from the above relation together with the fact that relation (78) implies that $\sum_{t=1}^{T}\left[\mathcal{L}_\beta(X_{t-1}, p_{t-1}) - \mathcal{L}_\beta(X_t, p_{t-1})\right] \leq \bar{\mathcal{F}}_\beta$. □

4 Computational experiments

In this section the performance of HALLaR is tested against state-of-the art SDP solvers. The experiments are performed on a 2019 Macbook Pro with an 8-core CPU and 32 GB of memory. The methods are tested in SDPs arising from the following applications: maximum stable set, phase retrieval, and matrix completion.

This section is organized into five subsections. The first subsection provides details on the implementation of HALLaR. The second subsection explains the SDP solvers considered in the experiments. The remaining subsections describe the results of the computational experiments in each of the applications.

4.1 Implementation details

Our implementation of HALLaR uses the Julia programming language. The implementation applies to a class of SDPs slightly more general than (P). Let $\mathbb{F} \in \{\mathbb{R}, \mathbb{C}\}$ be either the field of real or complex numbers. Let $\mathbb{S}^n(\mathbb{R})$ (resp. $\mathbb{S}^n(\mathbb{C})$) be the space of $n \times n$ symmetric (resp. complex Hermitian) matrices, with Frobenius inner product $\bullet$ and with positive semidefinite partial order $\succeq$. The implementation applies to SDPs of the form

$$\min_X \quad \{C \bullet X \quad : \quad \mathcal{A}X = b, \quad \operatorname{tr} X \leq \tau, \quad X \succeq 0, \quad X \in \mathbb{S}^n(\mathbb{F})\}$$

where $b \in \mathbb{R}^m$, $C \in \mathbb{S}(\mathbb{F})^n$, $\mathcal{A} : \mathbb{S}(\mathbb{F})^n \to \mathbb{R}^m$ is a linear map, and $\mathcal{A}^* : \mathbb{R}^m \to \mathbb{S}^n(\mathbb{F})$ is its adjoint. In contrast to (P), the trace of X is bounded by τ instead of one. The inputs for our implementation are the initial points U_0, p_0, the tolerances ϵ_c, ϵ_p, and the data describing the SDP instance, which is explained below. In the experiments, the primal initial point U_0 is a random $n \times 1$ matrix with entries generated independently from the Gaussian distribution over $\mathbb{F}$, and the dual initial point p_0 is the zero vector. Our computational results below are based on a variant of HALLaR, also referred to as HALLaR in this section, which differs slightly from one described in Section 3 in that the penalty parameter β and the tolerance $\bar{\epsilon}$ for the AL subproblem are chosen in an adaptive manner based on some of the ideas of the LANCELOT method [16].

The data defining the SDP instance involves matrices of size $n \times n$ which should not be stored as dense arrays in large scale settings. Instead of storing a matrix $M \in \mathbb{S}^n(\mathbb{F})$, it is assumed that a routine that evaluates the linear operator $\mathfrak{L}(M) : \mathbb{F}^n \to \mathbb{F}^n$, $v \mapsto Mv$ is given by the user.

Similar to the Sketchy-CGAL method of [67], our implementation of HALLaR requires the following user inputs to describe the SDP instance:

(i) The vector $b \in \mathbb{R}^m$ and the scalar $\tau > 0$.

(ii) A routine for evaluating the linear operator $\mathfrak{L}(C)$.

(iii) A routine for evaluating linear operators of the form $\mathfrak{L}(\mathcal{A}^* p)$ for any $p \in \mathbb{R}^m$.

(iv) A routine for evaluating the quadratic function

$$q_{\mathcal{A}} : \mathbb{F}^n \to \mathbb{R}^m, \quad y \mapsto \mathcal{A}(yy^T). \tag{85}$$

Note that the routine in (iv) allows $\mathcal{A}$ to be evaluated on any matrix in factorized form since $\mathcal{A}(YY^T) = \sum_i q_{\mathcal{A}}(y_i)$ where the sum is over the columns y_i of Y. In addition, the routines in (ii) and (iii) allow to multiply matrices of the form $C + \mathcal{A}^* p$ with a matrix Y by multiplying by each of the columns y_i separately. It follows that all steps of HALLaR (including the steps of HLR

and ADAP-AIPP) can be performed by using only the above inputs. For instance, the eigenvalue computation in Step 2 of HLR is performed using iterative Krylov methods, which only require matrix-vector multiplications.

4.2 Competing methods

We compare HALLaR against the following SDP solvers:

- CSDP : Open source Julia solver based on interior point methods;
- COSMO: Open source Julia solver based on ADMM/operator splitting;
- ManiSDP: Open source MATLAB solver based on AL and Riemannian trust-region method [61];
- SDPLR : Open source MATLAB solver based on Burer and Monteiro's LR method;
- SDPNAL+ : Open MATLAB source solver based on AL with a semismooth Newton method;
- T-CGAL : Thin variant of the CGAL method [65] that stores the iterates X_t in factored form;
- r-Sketchy : Low-rank variant of CGAL that only stores a sketch of $\pi(X_t) \in \mathbb{F}^{n\times r}$ of the iterates $X_t \in \mathbb{S}^n(\mathbb{F})$.

We use the default parameters in all methods. The r-Sketchy method is tested with two possible values of r, namely, $r = 10$ and $r = 100$.

Given a tolerance $\epsilon > 0$, all methods, except SDPLR, stop when a primal solution $X \in \Delta^n$ and a dual solution $(p, \theta, S) \in \mathbb{R}^m \times \mathbb{R}_+ \times \mathbb{S}^+_n$ satisfying

$$\frac{\|\mathcal{A}X - b\|}{1 + \|b\|} \le \epsilon, \qquad \frac{|\text{pval} - \text{dval}|}{1 + |\text{pval}| + |\text{dval}|} \le \epsilon, \qquad \frac{\|C + \mathcal{A}^*p + \theta I - S\|}{1 + \|C\|} \le \epsilon, \tag{86}$$

is generated, where pval and dval denote the primal and dual values of the solution, respectively. SDPLR terminates solely based on primal feasibility, i.e., it terminates when the first condition in (86) is satisfied. The above conditions are standard in the SDP literature, although some solvers use the l_∞ norm instead of the Euclidean norm. Given a vector $p \in \mathbb{R}^m$, HALLaR, SDPLR, r-Sketchy, and T-CGAL, set $\theta := \max\{-\lambda_{\min}(C + \mathcal{A}^*p), 0\}$ and $S := C + \mathcal{A}^*p + \theta I$. The definition of θ implies that $S \succeq 0$ and that the left-hand side of the last inequality in (86) is zero.

Recall that a description of r-Sketchy is already given in the paragraph preceding the part titled "Structure of the paper" in the Introduction. We now comment on how this method keeps track of its iterates and how it terminates. First, it never stores the iterate U_t but only a rank r approximation $\tilde{U}_t$ of it as already mentioned in the aforementioned paragraph of the Introduction. (We refer to U_t as the implicit iterate as is never computed and $\tilde{U}_t$ as the computed one.) Second, it keeps track of the quantities $C \bullet (U_t U_t^T)$ and $\mathcal{A}(U_t U_t^T)$ which, as can be easily verified, allow the first two relative errors in (86) with $X = U_t U_t^T$ to be easily evaluated. Third, we kept the termination criterion for r-Sketchy code intact in that it still stops when all three errors in (86) computed with the implicit solution U_t, i.e., with $X = U_t U_t^T$, are less than or equal to ε. The fact that r-Sketchy terminates based on the implicit solution does not guarantee, and is even unlikely, that it would terminate based on the computed solution. Fourth, if r-Sketchy does not terminate within the time limit specified for each problem class, the maximum of the three errors in (86) at its final computed solution $\tilde{U}_t$, i.e., with $X = \tilde{U}_t \tilde{U}_t^T$, is reported.

The solver COSMO includes an optional chordal decomposition pre-processing step (thoroughly discussed in [24, 26, 58]), which has not been invoked in the computational experiments reported below. This ensures that all solvers are compared over the same set of SDP instances.

The tables in the following three subsections present the results of the computational experiments performed on large collections of maximum stable set, phase retrieval, and matrix completion, SDP instances. A relative tolerance $\epsilon = 10^{-5}$ is set and a time limit of either 10000 seconds (≈ 3 hours) or 14400 seconds ($= 4$ hours) is given. An entry of a table marked with $*/N$ (resp., $**$) means that the corresponding method finds an approximate solution (resp., crashed) with relative accuracy strictly larger than the desired accuracy 10^{-5} in which case N expresses the maximum of the three final relative accuracies in (86). For r-Sketchy, entries marked as $*/N$ mean that it did not terminate within the time limit and the maximum of the three final relative accuracies in (86) of its final computed solution $\tilde{U}_t$ was N. The bold numbers in the tables indicate the algorithm that had the best runtime for that instance.

4.3 Maximum stable set

Given a graph $G = ([n], E)$, the maximum stable set problem consists of finding a subset of vertices of largest cardinality such that no two vertices are connected by an edge. Lovász [41] introduced a constant, the ϑ-function, which upper bounds the value of the maximum stable set. The ϑ-function is the value of the SDP

$$\max \quad \{ee^T \bullet X \ : \ X_{ij} = 0, \ ij \in E, \ \operatorname{tr} X = 1, \ X \succeq 0, \quad X \in \mathbb{S}^n(\mathbb{R})\} \tag{87}$$

where $e = (1, 1, \ldots, 1) \in \mathbb{R}^n$ is the all ones vector. It was shown in [27] that the ϑ-function agrees exactly with the stable set number for perfect graphs.

Tables 1, 2, 3, 4, and 5 present the results of the computational experiments performed on the maximum stable set SDP. Table 1 compares HALLaR against all the methods listed in Subsection 4.2 on smaller graph instances, i.e., with number of vertices not exceeding 70,000. All graph instances considered, except the last instance, are taken from the GSET data set, a curated collection of randomly generated graphs that can be found in [64]. The larger GSET graphs (GSET 66-81) are all toroidal graphs where every vertex has degree 4. The last graph instance presented in Table 1 is a large toroidal graph with approximately 70,000 vertices that we generated ourselves. A time limit of 10000 seconds (approximately 3 hours) is given. Table 2 compares HALLaR against T-CGAL, 10-Sketchy, and 100-Sketchy, on large graph instances with up to 1 million vertices and 10 million edges. CSDP was not included in Table 2 since it crashed on all but one of the instances included in it. All graph instances considered in Table 2 are Hamming $H(d, 2)$ graphs, a special class of graphs that has 2^d number of vertices and $d\,2^{d-1}$ number of edges. The vertex set of such graphs can be seen as corresponding to binary words of length d, and the edges correspond to binary words that differ in one bit. A time limit of 14400 seconds (4 hours) is now given.

Tables 3 and 4 solely present the performance of HALLaR on extremely large-sized Hamming instances (i.e., with number of vertices exceeding 1 millon) and hard graph instances from real-world datasets, respectively. The graph instances considered in Table 4 are taken from the DIMACS10, Stanford SNAP, AG-Monien, GHS_indef, and Network Repositories [1, 17, 40, 54].

Table 5 displays a special comparison between HALLaR, SDPLR, and ManiSDP on 6 different Hamming graphs. One of the reasons for not including these codes in Table 1 is that they require inputs in SDPA format which, for the GSET graphs, are not available in any public available repository and would be quite time-consuming to generate. Another reason is that SDPLR terminates only based on the first condition in (86) and hence often finds a solution that does not satisfy the second condition in (86) with the desired accuracy of $\epsilon = 10^{-5}$. An entry marked with time/N in

Problem Instance	**Runtime (seconds)**						
Graph(n; $\|E\|$)	HALLaR	T-CGAL	10-Sketchy	100-Sketchy	CSDP	COSMO	SDPNAL+
G1(800; 19,176)	218.23	*/.13e-02	*/.31e-01	*/.31e-01	226.01	322.91	**60.30**
G10(800; 19,176)	241.51	*/.20e-02	*/.78e-01	*/.31e-01	220.44	229.22	**55.30**
G11(800; 1,600)	**3.01**	1631.45	220.59	234.76	3.74	118.66	73.50
G12(800; 1,600)	**3.06**	414.27	72.56	57.03	3.12	9531.99	70.20
G14(800; 4,694)	66.46	*/.27e-03	6642.00	7231.30	**9.31**	3755.36	115.30
G20(800; 4,672)	439.37	*/.37e-03	*/.12e-01	*/.12e-01	**11.42**	*/.69e-04	341.20
G43(1000; 9,990)	96.79	*/.25e-04	*/.51e-01	*/.26e-01	**42.39**	*/.10e-02	62.10
G51(1,000; 5,909)	190.98	*/.23e-03	*/.98e-02	*/.99e-02	**16.97**	*/.33e-04	284.40
G23(2,000; 19,990)	390.34	*/.64e-03	*/.86e-01	*/.19e-01	**288.77**	5739.78	503.80
G31(2,000; 19,990)	357.75	*/.68e-03	*/.20e-01	*/.19e-01	**290.72**	5946.84	498.30
G32(2,000; 4,000)	**3.62**	3732.70	329.17	349.73	29.04	*/.58e-03	853.90
G34(2,000; 4,000)	**3.52**	1705.60	162.85	177.84	29.04	458.11	1101.80
G35(2,000; 11,778)	730.54	*/.78e-03	*/.62e-02	*/.62e-02	730.54	**120.60**	2396.60
G41(2,000; 11,785)	555.02	*/.17e-02	*/.59e-02	*/.59e-02	**114.73**	*/.37e-03	2027.20
G48(3,000; 6,000)	**3.49**	4069.30	288.64	306.91	81.97	1840.91	6347.50
G55(5,000; 12,498)	**253.22**	*/.33e-02	*/.80e-02	*/.79e-02	535.57	*/.11e-02	*/.17e-01
G56(5,000; 12,498)	**264.46**	*/.38e-02	*/.80e-02	*/.79e-02	523.06	*/.27e-02	*/.20e00
G57(5,000; 10,000)	**4.14**	7348.50	791.75	831.06	336.93	8951.40	*/.10e00
G58(5,000; 29,570)	2539.83	*/.96e-02	*/.24e-01	*/.43e-02	**2177.03**	*/.20e-03	*/.43e-01
G59(5,000; 29,570)	2625.47	*/.54e-02	*/.24e-01	*/.43e-02	**2178.33**	*/.37e+02	*/.65e-01
G60(7,000; 17,148)	**476.65**	*/.21e-02	*/.13e-01	*/.67e-02	2216.50	*/.39e+02	*/.10e+01
G62(7,000; 14,000)	**5.00**	*/.28e-03	1795.50	1474.40	1463.18	*/.12e-03	*/.10e+01
G64(7,000; 41,459)	**3901.68**	*/.16e-01	*/.36e-02	*/.36e-02	7127.72	*/.99e+01	*/.10e+01
G66(9,000; 18,000)	**5.77**	*/.82e-01	2788.70	3022.70	2076.17	*/.77e-03	*/.10e+01
G67(10,000; 20,000)	**5.87**	*/.13e-01	3725.80	3941.70	7599.80	**	*/.47e+01
G72(10,000; 20,000)	**5.92**	*/.11e00	3936.30	3868.60	7450.01	**	*/.47e+01
G77(14,000; 28,000)	**8.08**	*/.24e00	*/.60e-02	*/.60e-02	**	**	*/.99e00
G81(20,000; 40,000)	**10.89**	*/.10e00	*/.91e-01	*/.71e-01	**	**	*/.10e+01
tor(69,192; 138,384)	**40.64**	*/.38e00	*/.63e00	*/.29e00	**	**	**

Table 1: Runtimes (in seconds) for the Maximum Stable Set problem. A relative tolerance of $\epsilon = 10^{-5}$ is set and a time limit of 10000 seconds is given. An entry marked with $*/N$ (resp., $**$) means that the corresponding method finds an approximate solution (resp., crashed) with relative accuracy strictly larger than the desired accuracy in which case N expresses the maximum of the three relative accuracies in (86).

Table 5 means that SDPLR finds an approximate solution (within the time limit) that satisfies the first condition but not the second one in (86), in which case N expresses the final value of the left hand side of the latter condition. An entry marked with */N1/N2 means that SDPLR did not finish in the allowable time in which case N_1 and N_2 express the left-hand sides of the first and second conditions in (86), respectively.

Remarks about the results presented in Tables 1, 2, 3, 4, and 5 are now given. As seen from Table 1, CSDP and HALLaR are the two best-performing methods on these smaller graph instances. HALLaR, however, is the only method that can solve each of the instances to the desired accuracy of 10^{-5} within the time limit of approximately 3 hours. On graph instances where the number of vertices exceeds 14,000 (resp., 10,000), CSDP (resp., COSMO) cannot perform a single iteration within 3 hours or crashes. SDPNAL+ crashed with a lack of memory error on the last graph instance

Problem Instance	**Runtime (seconds)**			
Graph(n; $\|E\|$)	HALLaR	T-CGAL	10-Sketchy	100-Sketchy
$H_{13,2}$(8,192; 53,248)	**5.04**	*/.23e00	1603.80	882.03
$H_{14,2}$(16,384; 114,688)	**9.09**	*/.45e00	6058.60	6712.20
$H_{15,2}$(32,768; 245,760)	**65.22**	*/.19e00	*/.19e-01	*/.14e-01
$H_{16,2}$(65,536; 524,288)	**104.71**	*/.11e-01	*/.24e00	*/.11e-01
$H_{17,2}$(131,072; 1,114,112)	**69.63**	*/.34e-01	*/.72e00	*/.32e-01
$H_{18,2}$(262,144; 2,359,296)	**244.90**	*/.99e-02	*/.88e-02	*/.31e00
$H_{19,2}$(524,288; 4,980,736)	**786.73**	*/.42e00	*/.35e00	*/.24e00
$H_{20,2}$(1,048,576; 10,485,760)	**1138.17**	*/.47e00	*/.31e-02	*/.31e-02

Table 2: Runtimes (in seconds) for the Maximum Stable Set problem. A relative tolerance of $\epsilon = 10^{-5}$ is set and a time limit of 14400 seconds (4 hours) is given. An entry marked with $*/N$ (resp., $**$) means that the corresponding method finds an approximate solution (resp., crashed) with relative accuracy strictly larger than the desired accuracy in which case N expresses the maximum of the three relative accuracies in (86).

Problem Instance	**Runtime of HALLaR (seconds)**		
Graph(n; $\|E\|$)	$\epsilon = 10^{-5}$	$\epsilon = 10^{-8}$	$\epsilon = 10^{-10}$
$H_{20,2}$(1,048,576; 10,485,760)	1138.17	1157.19	1265.06
$H_{21,2}$(2,097,152; 22,020,096)	2815.50	2931.55	3134.84
$H_{22,2}$(4,194,304; 46,137,344)	6264.50	6920.11	7668.22
$H_{23,2}$(8,388,608; 96,468,992)	14188.23	16673.39	17029.87
$H_{24,2}$(16,777,216; 201,326,592)	46677.82	51941.35	53241.758

Table 3: Runtimes of HALLaR (in seconds) for the Maximum Stable Set problem. Relative tolerances of $\epsilon = 10^{-5}$, 10^{-8}, and 10^{-10} are set.

with 69,192 vertices. Although T-CGAL, 10-Sketchy, and 100-Sketchy do not perform especially well on the smaller graph instances considered in Table 1, they are included for comparison on the larger graph instances considered in Table 2 since they require considerably less memory than CSDP, COSMO, and SDPNAL+. The results presented in Table 2 demonstrate that HALLaR performs especially well for larger instances as it is the only method that can solve all instances within the time limit of 4 hours. T-CGAL, 10-Sketchy, and 100-Sketchy cannot find a solution with the desired accuracy of 10^{-5} on most of the instances considered, often finding solutions with accuracies on the range of 10^{0} to 10^{-2}. CSDP was tested on the problems considered in Table 2 but not included for comparison since it crashed on every instance except one. COSMO and SDPNAL+ are not included for comparison due to their high memory requirements.

Tables 3 and 4 show that HALLaR can solve extremely large Hamming instances and hard real-world instances, respectively, within a couple of hours. Table 3 shows that HALLaR can solve with 10^{-10} accuracy a Hamming instance with 4 million vertices and 40 million edges (resp., 16 million vertices and 200 million edges) in approximately 2 hours (resp., 15 hours). Table 4 shows that HALLaR can solve with 10^{-5} accuracy a huge Debruijin graph instance (which arises in the context of genome assembly) in just under 4 hours.

The results presented in Table 5 display the superior performance of HALLaR compared to SDPLR and ManiSDP on six different Hamming graphs. Compared to these two methods, HALLaR not only finds more accurate solutions within the time limit of 4 hours but is also at least 80 times faster than them on the three largest instances.

Problem Instance			Runtime (seconds)
Problem Size $(n; m)$	Graph	Dataset	HALLaR
10,937; 75,488	wing_nodal	DIMACS10	1918.48
16,384; 49,122	delaunay_n14	DIMACS10	1355.01
16,386; 49,152	fe-sphere	DIMACS10	147.93
22,499; 43,858	cs4	DIMACS10	747.66
25,016; 62,063	hi2010	DIMACS10	3438.06
25,181; 62,875	ri2010	DIMACS10	2077.97
32,580; 77,799	vt2010	DIMACS10	2802.37
48,837; 117,275	nh2010	DIMACS10	8530.38
24,300; 34,992	aug3d	GHS_indef	8.56
32,430; 54,397	ia-email-EU	Network Repo	530.21
11,806; 32,730	Oregon-2	SNAP	2787.19
11,380; 39,206	wiki-RFA_negative	SNAP	1151.31
21,363; 91,286	ca-CondMat	SNAP	7354.75
31,379; 65,910	as-caida_G_001	SNAP	3237.93
26,518; 65,369	p2p-Gnutella24	SNAP	344.83
22,687; 54,705	p2p-Gnutella25	SNAP	235.03
36,682; 88,328	p2p-Gnutella30	SNAP	542.07
62,586; 147,892	p2p-Gnutella31	SNAP	1918.30
49,152; 69,632	cca	AG-Monien	47.24
49,152; 73,728	ccc	AG-Monien	12.14
49,152; 98,304	bfly	AG-Monien	13.15
16,384; 32,765	debr_G_12	AG-Monien	818.61
32,768; 65,533	debr_G_13	AG-Monien	504.29
65,536; 131,069	debr_G_14	AG-Monien	466.67
131,072; 262,141	debr_G_15	AG-Monien	488.07
262,144; 524,285	debr_G_16	AG-Monien	1266.71
524,288; 1,048,573	debr_G_17	AG-Monien	5793.57
1,048,576; 2,097,149	debr_G_18	AG-Monien	13679.12

Table 4: Runtimes (in seconds) for the Maximum stable set problem. A relative tolerance of $\epsilon = 10^{-5}$ is set.

Problem Instance	Runtime (seconds)		
Graph$(n; \|E\|)$	HALLaR	SDPLR	ManiSDP
$H_{10,2}(1024; 5120)$	2.90	**1.28**	4.65
$H_{11,2}(2048; 11264)$	**3.03**	10.14	20.38
$H_{12,2}(4096; 24576)$	**3.49**	56.60/.12e-03	140.26
$H_{13,2}(8192; 53248)$	**5.04**	399.89/.38e-03	1862.00
$H_{14,2}(16384; 114688)$	**9.09**	2469.11/.16e-02	12264.47
$H_{15,2}(32768; 245760)$	**65.22**	*/.11e-01/.46e00	*/.15e-01

Table 5: Runtimes (in seconds) for the maximum stable set problem. A relative tolerance of $\epsilon = 10^{-5}$ is set and a time limit of 14400 seconds (4 hours) is given. An entry marked with time/N means that SDPLR finds an approximate solution (within the time limit) that satisfies the first condition but not the second one in (86), in which case N expresses the final value of the left hand side of the latter condition. An entry marked with */N1/N2 means that SDPLR did not finish in 4 hours in which case N_1 and N_2 express the left-hand sides of the first and second conditions in (86), respectively.

4.4 Phase retrieval

Given m pairs $\{(a_i, b_i)\}_{i=1}^m \subseteq \mathbb{C}^n \times \mathbb{R}_+$, consider the problem of finding a vector $x \in \mathbb{C}^n$ such that

$$|\langle a_i, x\rangle|^2 = b_i, \quad i = 1, \ldots, m.$$

In other words, the goal is to retrieve x from the magnitude of m linear measurements. By creating the complex Hermitian matrix $X = xx^H$, this problem can be approached by solving the complex-valued SDP relaxation

$$\min_X \quad \left\{\mathrm{tr}(X) \quad : \quad \langle a_i a_i^H, X\rangle = b_i, \quad X \succeq 0, \quad X \in \mathbb{S}^n(\mathbb{C})\right\}.$$

The motivation of the trace objective function is that it promotes obtaining a low rank solution. It was shown in [12] that the relaxation is tight (i.e., the vector x can be retrieved from the SDP solution X) when the vectors a_i are sampled independently and uniformly on the unit sphere. Notice that this class of SDPs does not have a trace bound. However, since the objective function is precisely the trace, any bound on the optimal value can be used as the trace bound. In particular, the squared norm of the vector x is a valid trace bound. Even though x is unknown, bounds on its norm are known (see for example [66]).

Computational experiments are performed on the synthetic data set from [67] that is based on the coded diffraction pattern model from [11]. Given n, the hidden solution vector $x \in \mathbb{C}^n$ is generated from the complex standard normal distribution. The are $m = 12n$ measurements that are indexed by pairs $(j, l) \in [12] \times [n]$. Consider vectors $y_j \in \mathbb{C}^n$ for $j \in [12]$, where the entries of y_j are products of of two independent random variables: the first is the uniform distribution on $\{1, i, -1, -i\}$, and the second chooses from $\{\sqrt{2}/2, \sqrt{3}\}$ with probabilities 4/5 and 1/5. The linear measurements correspond to modulating the vector x with each of the y_j's and then taking a discrete Fourier transform:

$$\langle a_{j,k}, x\rangle := \mathrm{DFT}(y_j \circ x)_l \text{ for } j \in [12],\ l \in [n]$$

where $\circ$ denotes the Hadamard product, and $\mathrm{DFT}(\cdot)_l$ denotes the l-th entry of the discrete Fourier transform. The vector b is obtained by applying the measurements to x. The trace bound is set as $\tau = 3n$, similarly as in [67].

Tables 6 and 7 present the results of the computational experiments performed on the phase retrieval SDP. As mentioned in the above paragraph, all instances considered are taken from a synthetic dataset that can be found in [67]. Table 6 compares HALLaR against T-CGAL, 10-Sketchy, and 100-Sketchy on medium sized phase retrieval instances, i.e., the dimension n is either 10000 or 31623. The ranks of the outputted solutions of HALLaR and T-CGAL are now also reported. For entries corresponding to HALLaR and T-CGAL, the number reported after the last forward slash indicates the rank of that corresponding method's outputted solution. A time limit of 14400 seconds (4 hours) is given. Table 7 solely presents the performance of HALLaR on larger sized phase retrieval instances, i.e., with dimension n greater than or equal to 100,000. The rank of the outputted solution of HALLaR is again reported.

Table 6 only compares HALLaR against T-CGAL and Sketchy-CGAL since these are the only methods that take advantage of the fact that the linear maps $\mathcal{A}$ and $\mathcal{A}^*$ in the phase retrieval SDP can be evaluated efficiently using the fast Fourier transform (FFT). As seen from Table 6, HALLaR is the best performing method and the only method that can solve each instance to a relative accuracy of 10^{-5} within the time limit of 4 hours. T-CGAL and Sketchy-CGAL were unable to solve most instances to the desired accuracy, often finding solutions with accuracies on the range of

Problem Instance	**Runtime (seconds)**			
Problem Size $(n; m)$	HALLaR	T-CGAL	10-Sketchy	100-Sketchy
10,000; 120,000	**69.11/2**	*/.18e-01/561	*/.13e-01	*/.25e-01
10,000; 120,000	**66.14/2**	*/.54e-01/521	11112.00	*/.80e-01
10,000; 120,000	**64.42/2**	*/.12e00/224	*/.31e-01	*/.13e00
10,000; 120,000	**99.98/2**	*/.28e-01/201	*/.13e00	*/.26e-01
31,623; 379,476	**620.82/3**	*/.29e00/1432	*/.77e-01	*/.23e00
31,623; 379,476	**982.34/2**	*/.23e00/729	*/.63e-01	*/.93e00
31,623; 379,476	**870.25/2**	*/.66e00/794	*/.65e-02	*/.78e-01
31,623; 379,476	**712.09/2**	*/.10e+01/1280	*/.10e+01	*/.82e00

Table 6: Runtimes (in seconds) for the Phase Retrieval problem. A relative tolerance of $\epsilon = 10^{-5}$ is set and a time limit of 14400 seconds (4 hours) is given. An entry marked with $*/N$ means that the corresponding method finds an approximate solution with relative accuracy strictly larger than the desired accuracy in which case N expresses the maximum of the three relative accuracies in (86). For entries corresponding to HALLaR and T-CGAL, the number reported after the last forward slash indicates that the rank of that corresponding method's outputted solution.

Problem Instance	**Runtime (seconds)**
Problem Size $(n; m)$	HALLaR
100,000; 1,200,000	1042.92/4
100,000; 1,200,000	1147.46/3
100,000; 1,200,000	929.67/5
100,000; 1,200,000	939.23/5
316,228; 3,794,736	8426.94/5
316,228; 3,794,736	2684.83/1
316,228; 3,794,736	7117.31/6
316,228; 3,794,736	7489.42/7
3,162,278; 37,947,336	40569.10/1

Table 7: Runtimes (in seconds) for the Phase Retrieval problem. The number after the forward slash indicates the rank of HALLaR's outputted solution. A relative tolerance of $\epsilon = 10^{-5}$ is set.

10^0 to 10^{-2} in 4 hours. Sketchy-CGAL was also over 150 times slower than HALLaR on the single instance that it was able to find a 10^{-5} accurate solution.

Since T-CGAL and Sketchy-CGAL did not perform well on the medium sized phase retrieval instances considered in Table 6, computational results for large sized phase retrieval instances are only presented for HALLaR in Table 7. The results presented in Table 7 show that HALLaR solves a phase retrieval SDP instance with dimension pair $(n, m) \approx (10^5, 10^6)$ in approximately 15 minutes and also one with dimension pair $(n, m) \approx (10^6, 10^7)$ in just 11 hours.

4.5 Matrix completion

Consider the problem of retrieving a low rank matrix $M \in \mathbb{R}^{n_1 \times n_2}$, where $n_1 \leq n_2$, by observing a subset of its entries: M_{ij}, $ij \in \Omega$. A standard approach to tackle this problem is by considering the nuclear norm relaxation:

$$\min_{Y} \quad \left\{ \|Y\|_* \quad : \quad Y_{ij} = M_{ij}, \ \forall\, ij \in \Omega, \quad Y \in \mathbb{R}^{n_1 \times n_2} \right\}$$

The above problem can be rephrased as the following SDP:

$$\min_X \quad \left\{ \frac{1}{2}\operatorname{tr}(X) \quad : \quad X = \begin{pmatrix} W_1 & Y \\ Y^T & W_2 \end{pmatrix} \succeq 0, \quad Y_{i,j} = M_{i,j} \; \forall \, ij \in \Omega, \quad X \in \mathbb{S}^{n_1+n_2}(\mathbb{R}) \right\} \tag{88}$$

Problem Instance		**Runtime (seconds)**	
Problem Size $(n; m)$	r	HALLaR	10-Sketchy
10,000; 828,931	3	**321.81**	*/.81e00/.89e-02
10,000; 828,931	3	**332.54**	*/.80e00/.82e-02
10,000; 2,302,586	5	**1117.60**	*/.92e00/.28e00
10,000; 2,302,586	5	**1067.15**	*/.11e+01/ .41e00
31,623; 2,948,996	3	**1681.03**	*/.81e00/.69e-02
31,623; 2,948,996	3	**1362.22**	*/.81e00/.82e-02
31,623; 8,191,654	5	**4740.48**	*/.90e00/.43e-01
31,623; 8,191,654	5	**5238.57**	*/.90e00/.84e-01

Table 8: Runtimes (in seconds) for the Matrix Completion problem. A relative tolerance of $\epsilon = 10^{-5}$ is set and a time limit of 14400 seconds (4 hours) is given. An entry marked with $*/N_1/N_2$ means that the implicit solution corresponding to 10-Sketchy had relative accuracy strictly larger than the desired accuracy in which case N_1 (resp., N_2) expresses the maximum of the three relative accuracies in (86) of its computed (resp., implicit) solution.

Problem Instance		**Runtime (seconds)**
Problem Size $(n; m)$	r	HALLaR
75,000; 3,367,574	2	3279.85
75,000; 7,577,040	3	5083.68
100,000; 4,605,171	2	2872.44
100,000; 10,361,633	3	6048.63
150,000; 7,151,035	2	10967.74
150,000; 16,089,828	3	14908.08
200,000; 9,764,859	2	13454.12
200,000; 21,970,931	3	28021.56

Table 9: Runtimes (in seconds) for the Matrix Completion problem. A relative tolerance of $\epsilon = 10^{-5}$ is set.

The nuclear norm relaxation was introduced in [22]. It was shown in [10] it provably completes the matrix when $m = |\Omega|$ is sufficiently large and the indices of the observations are independent and uniform.

Similar to the SDP formulation of phase retrieval in subsection 4.4, the SDP formulation of matrix completion does not include a trace bound, but the objective function is a multiple of the trace. Hence, any bound on the optimal value leads to a trace bound. In particular, a valid trace bound is $2\|Y_0\|_*$, where $Y_0 \in \mathbb{R}^{n_1 \times n_2}$ is the trivial completion, which agrees with M_{ij} in the observed entries and has zeros everywhere else. However, computing the nuclear norm of Y_0 is expensive, as it requires an SVD decomposition. In the experiments the inexpensive, though weaker, bound $\tau = 2\sqrt{n_1}\|Y_0\|_F$ is used instead.

The matrix completion instances are generated randomly, using the following procedure. Given $r \leq n_1 \leq n_2$, the hidden solution matrix M is the product UV^T, where the matrices $U \in \mathbb{R}^{n_1 \times r}$ and $V \in \mathbb{R}^{n_2 \times r}$ have independent standard Gaussian random variables as entries. Afterwards, m

independent and uniformly random observations from M are taken. The number of observations is $m = \lceil \gamma\, r(n_1 + n_2 - r) \rceil$ where $\gamma = r \log(n_1 + n_2)$ is the oversampling ratio.

Tables 8 and 9 present the results of the computational experiments performed on the matrix completion SDP. All instances are generated randomly using the procedure described in the previous paragraph. Table 8 compares HALLaR against 10-Sketchy on medium sized matrix completion instances, i.e., the dimension $n = n_1 + n_2$ is either 10000 or 31623. A time limit of 14400 seconds (4 hours) is given. On instances where 10-Sketchy did not terminate within the time limit, the relative accuracy of both of its computed and implicit solutions are now reported. An entry marked with $*/N_1/N_2$ means that, within 4 hours, the implicit solution corresponding to 10-Sketchy had relative accuracy strictly larger than the desired accuracy in which case N_1 (resp., N_2) expresses the maximum of the three relative accuracies in (86) of its computed (resp., implicit) solution. Table 9 solely presents the performance of HALLaR on larger sized matrix completion instances, i.e., with dimension n greater than or equal to 75000.

Table 8 only compares HALLaR against 10-Sketchy due to 10-Sketchy's low memory requirements and its superior/comparable performance to 100-Sketchy and T-CGAL on previous problem classes. As seen from Table 8, HALLaR is the best performing method and the only method that can solve each instance to a relative accuracy of 10^{-5} within the time limit of 4 hours. 10-Sketchy is unable to solve a single instance to the desired accuracy, often finding solutions with accuracies on the range of 10^0 to 10^{-2} in 4 hours.

Since 10-Sketchy did not perform well on the medium sized matrix completion instances considered in Table 8, computational results for large sized matrix completion instances are only presented for HALLaR in Table 9. The results presented in Table 9 show that HALLaR solves a matrix completion instance with dimension pair $(n, m) \approx (10^5, 10^6)$ in approximately 48 minutes and also one with dimension pair $(n, m) \approx (10^5, 10^7)$ in just 1.7 hours.

A Technical Results

The following section states some useful facts about the spectraplex Δ^n defined in (1). The first result characterizes the optimal solution a given linear form over the set Δ^n. The second result characterizes its ϵ-normal cone.

A.1 Characterization of Optimal Solution of Linear Form over Spectraplex

Consider the problem

$$\min_U \{G \bullet U : U \in \Delta^n\}, \tag{89}$$

where Δ^n is as in (1). The optimality condition for (89) implies that Z is an optimal solution of (89) if and only if there exists $\theta \in \mathbb{R}$ such that the pair (Z, θ) satisfies

$$G + \theta I \succeq 0, \quad (G + \theta I) \bullet Z = 0, \quad \theta \geq 0, \quad \theta(\mathrm{tr}(Z) - 1) = 0. \tag{90}$$

The following proposition explicitly characterizes solutions of the above problem using the special structure of Δ^n.

Proposition A.1. *Let $(\lambda_{\min}, v_{\min})$ be a minimum eigenpair of G and define*

$$\theta^F = \max\{-\lambda_{\min}(G), 0\}, \quad Z^F = \begin{cases} v_{\min} v_{\min}^T & \text{if } \theta^F > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{91}$$

Then the pair $(Z, \theta) = (Z^F, \theta^F)$ satisfies (90)*. As a consequence, Z^F is an optimal solution of* (89)*.*

Proof. Consider the pair (Z^F, θ^F) as defined in (91). It is immediate from the definition of θ^F that $\theta^F \geq 0$. Consider now two cases. For the first case, suppose that $\theta^F = 0$ and hence $Z^F = 0$. It then follows immediately that the pair (Z^F, θ^F) satisfies the optimality conditions in (90).

For the second case, suppose that $\theta^F > 0$. Thus $\theta^F = -\lambda_{\min}(G)$ and $Z^F = v_{\min} v_{\min}^T$. Clearly, then $G + \theta^F I \succeq 0$ and $\operatorname{tr}(Z^F) = 1$ and hence the pair (Z^F, θ^F) satisfies the first and fourth relations in (90). The fact that $(-\theta^F, v_{\min})$ is an eigenpair implies that $G \bullet Z^F = -\theta^F$ and hence the pair (Z^F, θ^F) also satisfies the second relation in (90).

□

A.2 Characterization of ϵ-Normal Cone of Spectraplex

Proposition A.2. *Let $Z \in \Delta^n$ and $\theta = \theta^F$ where θ^F is as in (91). Then:*

a) there hold

$$\theta \geq 0, \quad G + \theta I \succeq 0; \tag{92}$$

b) for any $\epsilon > 0$, the inclusion holds

$$0 \in G + N^{\epsilon}_{\Delta^n}(Z) \tag{93}$$

if and only if

$$G \bullet Z + \theta \leq \epsilon. \tag{94}$$

Proof. (a) It follows immediately from the definition of θ^F in (91) and the fact that $\theta = \theta^F$ that the two relations in (92) hold.

(b) Proposition A.1 implies that the pair (Z^F, θ^F) satisfies the relation $G \bullet Z^F = -\theta^F$ where (Z^F, θ^F) is as in (91). It then follows from this relation, the fact that $\theta = \theta^F$, and the inclusions $-G \in N^{\epsilon}_{\Delta^n}(Z)$ and $Z^F \in \Delta^n$ that

$$\epsilon \geq -G \bullet (Z^F - Z) = G \bullet Z + \theta.$$

For the other direction, suppose the pair (Z, θ) satisfies (94) and let $U \in \Delta^n$. It then follows that

$$\begin{aligned} -G \bullet (U - Z) &= G \bullet Z - (G + \theta I) \bullet U + \theta \operatorname{tr}(U) \\ &\leq G \bullet Z + \theta - (G + \theta I) \bullet U \\ &\leq \epsilon - 0 = \epsilon. \end{aligned}$$

Hence, $-G \in N^{\epsilon}_{\Delta^n}(Z)$, proving the result. □

B ADAP-FISTA Method

Let $\mathbb{E}$ denote a finite-dimensional inner product real vector space with inner product and induced norm denoted by $\langle \cdot, \cdot \rangle$ and $\| \cdot \|$, respectively. Also, let $\psi_n : \mathbb{E} \to (-\infty, \infty]$ be a proper closed convex function whose domain $\operatorname{dom} \psi_n := \mathcal{N} \subseteq \mathbb{E}$, has finite diameter D_{ψ_n}.

ADAP-FISTA considers the following problem

$$\min\{\psi(u) := \psi_s(u) + \psi_n(u) : u \in \mathbb{E}\} \tag{95}$$

where ψ_s is assumed to satisfy the following assumption:

(B1) ψ_s is a real-valued function that is differentiable on $\mathbb{E}$ and there exists $\bar{L} \geq 0$ such that

$$\|\nabla\psi_s(u') - \nabla\psi_s(u)\| \leq \bar{L}\|u' - u\| \quad \forall u, u' \in \tilde{B}, \tag{96}$$

where

$$\tilde{B} := \mathcal{N} + \bar{B}_{D_{\psi_n}} \tag{97}$$

and $\bar{B}_l := \{u : \|u\| \leq l\}$ is the closed unit ball centered at 0 with radius l.

We now describe the type of approximate solution that ADAP-FISTA aims to find.

Problem: Given ψ satisfying the above assumptions, a point $x_0 \in \mathcal{N}$, a parameter $\sigma \in (0, \infty)$, the problem is to find a pair $(y, r) \in \mathcal{N} \times \mathbb{E}$ such that

$$\|r\| \leq \sigma\|y - x_0\|, \quad r \in \nabla\psi_s(y) + \partial\psi_n(y). \tag{98}$$

We are now ready to present the ADAP-FISTA algorithm below.

ADAP-FISTA Method

Universal Parameters: $\sigma > 0$ and $\chi \in (0, 1)$.
Input: initial point $x_0 \in \mathcal{N}$, scalars $\mu > 0$, $L_0 > \mu$, and function pair (ψ_s, ψ_n).

0. set $y_0 = x_0$, $A_0 = 0$, $\tau_0 = 1$, and $i = 0$;

1. Set $L_{i+1} = L_i$;

2. Compute

$$a_i = \frac{\tau_i + \sqrt{\tau_i^2 + 4\tau_i A_i(L_{i+1} - \mu)}}{2(L_{i+1} - \mu)}, \quad \tilde{x}_i = \frac{A_i y_i + a_i x_i}{A_i + a_i}, \tag{99}$$

$$y_{i+1} := \arg\min_{u \in \mathcal{N}} \left\{ q_i(u; \tilde{x}_i, L_{i+1}) := \ell_{\psi_s}(u; \tilde{x}_i) + \psi_n(u) + \frac{L_{i+1}}{2}\|u - \tilde{x}_i\|^2 \right\}, \tag{100}$$

If the inequality

$$\ell_{\psi_s}(y_{i+1}; \tilde{x}_i) + \frac{(1 - \chi)L_{i+1}}{4}\|y_{i+1} - \tilde{x}_i\|^2 \geq \psi_s(y_{i+1}) \tag{101}$$

holds go to step 3; else set $L_{i+1} \leftarrow 2L_{i+1}$ and repeat step 2;

3. Compute

$$A_{i+1} = A_i + a_i, \quad \tau_{i+1} = \tau_i + a_i\mu, \tag{102}$$

$$s_{i+1} = (L_{i+1} - \mu)(\tilde{x}_i - y_{i+1}), \tag{103}$$

$$x_{i+1} = \frac{1}{\tau_{i+1}}\left[\mu a_i y_{i+1} + \tau_i x_i - a_i s_{i+1}\right]; \tag{104}$$

4. If the inequality

$$\|y_{i+1} - x_0\|^2 \geq \chi A_{i+1} L_{i+1}\|y_{i+1} - \tilde{x}_i\|^2, \tag{105}$$

holds, then go to step 5; otherwise, stop with **failure**;

5. Compute

$$v_{i+1} = \nabla\psi_s(y_{i+1}) - \nabla\psi_s(\tilde{x}_i) + L_{i+1}(\tilde{x}_i - y_{i+1}). \tag{106}$$

If the inequality

$$\|v_{i+1}\| \leq \sigma\|y_{i+1} - x_0\| \tag{107}$$

holds then stop with **success** and output $(y, v, L) := (y_{i+1}, v_{i+1}, L_{i+1})$; otherwise, $i \leftarrow i + 1$ and go to step 1.

The ADAP-FISTA method was first developed in [57]. The method assumes that the gradient of ψ_s is Lipschitz continuous on all of $\mathbb{E}$ since it requires Lipchitz continuity of the gradient at the sequences of points $\{\tilde{x}_i\}$ and $\{y_i\}$. This assumption can be relaxed to as in (B3) by showing that the sequence $\{\tilde{x}_i\}$ lies in the set $\tilde{B}$ defined in (97). The following lemma establishes this result inductively by using several key lemmas which can be found in [57].

Lemma B.1. *Let $m \geq 1$ and suppose ADAP-FISTA generates sequence $\{\tilde{x}_i\}_{i=0}^m \subseteq \tilde{B}$. Then, the following statements hold:*

(a) $L_0 \leq L_{i+1} \leq \max\{L_0, 4\bar{L}/(1-\chi)\}$ for any $i \in \{0, \dots m\}$;

(b) for any $x \in \mathcal{N}$, the relation

$$A_i[\psi(y_{m+1}) - \psi(x)] + \frac{\tau_{m+1}}{2}\|x - x_{m+1}\|^2 \leq \frac{1}{2}\|x - x_0\|^2 - \frac{\chi}{2}\sum_{i=0}^{m} A_{i+1}L_{i+1}\|y_{i+1} - \tilde{x}_i\|^2 \tag{108}$$

holds;

(c) $\tilde{x}_{m+1} \in \tilde{B}$.

As a consequence, the entire sequence $\{\tilde{x}_i\} \subseteq \tilde{B}$.

Proof. (a) Let $i \in \{0, \dots m\}$. Clearly $y_{i+1} \in \tilde{B}$ since the definition of $\tilde{B}$ in (97) implies that $\tilde{B} \supseteq \mathcal{N}$. Now, using the facts that $\tilde{x}_i \in \tilde{B}$ and $y_{i+1} \in \tilde{B}$, assumption (B1) implies that $\nabla\psi_s$ is Lipschitz continuous at these points. The proof of (a) is then identical to the one of Lemma A.3(b) in [57].

(b) Clearly, since ADAP-FISTA generates sequence $\{\tilde{x}_i\}_{i=0}^m$, its loop in step 2 always terminates during its first m iterations. Hence, ADAP-FISTA also generates sequences $\{y_i\}_{i=0}^{m+1}$ and $\{x_i\}_{i=0}^{m+1}$. The proof of relation (108) then follows from this observation, (a), and by using similar arguments to the ones developed in Lemmas A.6-A.10 of [57].

(c) It follows from the fact that $\tau_{m+1} \geq 1$, relation (108) with $x = y_{m+1}$, and the definition of D_{ψ_n} that $\|y_{m+1} - x_{m+1}\| \leq \|y_{m+1} - x_0\| \leq D_{\psi_n}$. This relation, the fact that $y_{m+1} \in \mathcal{N}$, and the definition of $\tilde{B}$ in (97) then imply that

$$x_{m+1} \subseteq \mathcal{N} + \bar{B}_{D_{\psi_n}} = \tilde{B}.$$

The result then immediately follows from the fact that $\tilde{x}_{m+1}$ is a convex combination of x_{m+1} and y_{m+1} and that $\tilde{B}$ is a convex set.

The last statement in Proposition B.1 follows immediately from (c) and a simple induction argument. □

We now present the main convergence results of ADAP-FISTA, whose proofs can be found in [57]. Proposition B.2 below gives an iteration complexity bound regardless if ADAP-FISTA terminates with success or failure and shows that if ADAP-FISTA successfully stops, then it obtains a stationary solution of (95) with respect to a relative error criterion. It also shows that ADAP-FISTA always stops successfully whenever ψ_s is μ-strongly convex.

Proposition B.2. *The following statements about ADAP-FISTA hold:*

(a) if $L_0 = \mathcal{O}(\bar{L})$, it always stops (with either success or failure) in at most

$$\mathcal{O}_1\left(\sqrt{\frac{\bar{L}}{\mu}}\log_1^+(\bar{L})\right)$$

iterations/resolvent evaluations;

(b) if it stops successfully, it terminates with a triple $(y, v, L) \in \mathcal{N} \times \mathbb{E} \times \mathbb{R}_{++}$ satisfying

$$v \in \nabla\psi_s(y) + \partial\psi_n(y), \quad \|v\| \le \sigma\|y - x_0\|, \quad L \le \max\{L_0, 4\bar{L}/(1-\chi)\}; \tag{109}$$

(c) if ψ_s is μ-convex on $\mathcal{N}$, then ADAP-FISTA always terminates with success and its output (y, v, L), in addition to satisfying (109) also satisfies the inclusion $v \in \partial(\psi_s + \psi_n)(y)$.

C Proof of Proposition 2.4

This section provides the proof of Proposition 2.4 stated in Subsection 2.2.

Let $\bar{B}_r := \{U : \|U\|_F \le r\}$. The following lemma establishes that the function $\tilde{g}(\cdot)$ in (15) has Lipschitz continuous gradient over $\bar{B}_3$.

Lemma C.1. *The function $\tilde{g}(U)$ defined in (15) is $L_{\tilde{g}}$-smooth on $\bar{B}_3$ where $L_{\tilde{g}}$ is as in (25).*

Proof. Let $U_1, U_2 \in \bar{B}_3$ be given. Adding and subtracting $U_1U_2^T$ and using the triangle inequality, we have

$$\|U_1U_1^T - U_2U_2^T\|_F \le \|U_1\|\,\|U_1 - U_2\|_F + \|U_2\|\,\|U_1 - U_2\|_F \le 6\|U_1 - U_2\|_F. \tag{110}$$

This relation, the chain rule, the triangle inequality, the facts that $U_1, U_2 \in \bar{B}_3$ and g is L_g-smooth on $\bar{B}_3$, and the definition of $\bar{G}$ in (25), then imply that

$$\begin{aligned}
&\|\nabla\tilde{g}(U_1) - \nabla\tilde{g}(U_2)\|_F = \|2\nabla g(U_1U_1^T)U_1 - 2\nabla g(U_2U_2^T)U_2\|_F \\
&\le \|2\nabla g(U_1U_1^T)U_1 - 2\nabla g(U_1U_1^T)U_2\|_F + \|2\nabla g(U_1U_1^T)U_2 - 2\nabla g(U_2U_2^T)U_2\|_F \\
&\le 2\|\nabla g(U_1U_1^T)\|_F\,\|U_1 - U_2\|_F + 2\|U_2\|\,\|\nabla g(U_1U_1^T) - \nabla g(U_2U_2^T)\|_F \\
&\overset{(25)}{\le} 2\bar{G}\|U_1 - U_2\|_F + 6\|\nabla g(U_1U_1^T) - \nabla g(U_2U_2^T)\|_F \\
&\overset{(9)}{\le} 2\bar{G}\|U_1 - U_2\|_F + 6L_g\|U_1U_1^T - U_2U_2^T\|_F \\
&\overset{(110)}{\le} 2\bar{G}\|U_1 - U_2\|_F + 36L_g\|U_1 - U_2\|_F = (2\bar{G} + 36L_g)\|U_1 - U_2\|_F.
\end{aligned}$$

The conclusion of the lemma now follows from the above inequality and the definition of $L_{\tilde{g}}$ in (25). □

The following lemma establishes key properties about each ADAP-FISTA call made in step 1 of ADAP-AIPP. It is a translation of the results in Proposition B.2.

Lemma C.2. *Let (ψ_s, ψ_n) be as in (20). The following statements about each ADAP-FISTA call made in the j-th iteration of ADAP-AIPP hold:*

(a) if $\underline{M}_j = \mathcal{O}(1 + \lambda_0 L_{\tilde{g}})$, it always stops (with either success or failure) in at most

$$\mathcal{O}_1\left(\sqrt{2\,(1+\lambda_0 L_{\tilde{g}})}\log_1^+(1+\lambda_0 L_{\tilde{g}})\right)$$

iterations/resolvent evaluations where λ_0 is the initial prox stepsize and $L_{\tilde{g}}$ is as in (25);

(b) if ADAP-FISTA stops successfully, it terminates with a triple $(W, V, L) \in \bar{B}_1 \times \mathbb{S}^n \times \mathbb{R}_{++}$ satisfying

$$V \in \lambda\left[\nabla\tilde{g}(W) + \partial\delta_{\bar{B}_1}(W)\right] + (W - W_{j-1}) \tag{111}$$

$$\|V\|_F \le \sigma\|W - W_{j-1}\|_F, \quad L \le \max\{\underline{M}_j, \omega(1+\lambda_0 L_{\tilde{g}})\} \tag{112}$$

where $\omega = 4/(1-\chi)$;

(c) if ψ_s is $1/2$-convex on $\bar{B}_1$, then ADAP-FISTA always terminates with success and its output (W, V, L) always satisfies relation (21).

Proof. (a) The result follows directly from Proposition B.2 in Appendix B and hence the proof relies on verifying its assumptions. First it is easy to see that $\operatorname{dom}\psi_n + \bar{B}_2 = \bar{B}_3$, where ψ_n is as in (20). It also follows immediately from the fact that $\lambda \le \lambda_0$ and from Lemma C.1 that ψ_s is $(1+\lambda_0 L_{\tilde{g}})$-smooth on $\bar{B}_3$ in view of its definition in (20). These two observations imply that $\bar{L} = 1 + \lambda_0 L_{\tilde{g}}$ satisfies (96). Hence, it follows from this conclusion, the fact that each ADAP-FISTA call is made with $(\mu, L_0) = (1/2, \underline{M}_j)$, and from Proposition B.2(a) that statement (a) holds.

(b) In view of the definition of ψ_s in (20), it is easy that see that $\nabla\psi_s(W) = \lambda\tilde{g}(W) + (W - W_{j-1})$. Statement (b) then immediately follows from this observation, the fact that each ADAP-FISTA call is made with inputs $x_0 = W_{j-1}$, $L_0 = \underline{M}_j$, and (ψ_s, ψ_n) as in (20), and from Proposition B.2(b) with $\bar{L} = 1 + \lambda_0 L_{\tilde{g}}$.

(c) It follows immediately from Proposition B.2(c) and the fact that each ADAP-FISTA call is made with inputs (ψ_s, ψ_n) as in (20) and $\mu = 1/2$ that the first conclusion of statement (c) holds and that output (W, V, L) satisfies inclusion

$$V \in \partial\left(\lambda\left(\tilde{g} + \delta_{\bar{B}_1}\right)(\cdot) + \frac{1}{2}\|\cdot - W_{j-1}\|_F^2\right)(W). \tag{113}$$

Inclusion (113) and the definition of subdifferential in (5) then immediately imply that output (W, V, L) satisfies relation (21). □

The lemma below shows that, in every iteration of ADAP-AIPP, the loop within steps 1 and 2 always stops and shows key properties of its output. Its proof (included here for completeness) closely follows the one of Proposition 3.1 of [57].

Lemma C.3. *The following statements about ADAP-AIPP hold for every $j \ge 1$:*

(a) the function ψ_s in (20) has $(1+\lambda_0 L_{\tilde{g}})$-Lipschitz continuous gradient on $\bar{B}_3$;

(b) the loop within steps 1 and 2 of its j-th iteration always ends and the output $(W_j, V_j, R_j, \lambda_j, \bar{M}_j)$ obtained at the end of step 2 satisfies

$$R_j \in \nabla\tilde{g}(W_j) + \partial\delta_{\bar{B}_1}(W_j); \tag{114}$$

$$\left(\frac{1-\sigma}{\sigma}\right)\|V_j\|_F \le \|\lambda_j R_j\|_F \le (1+\sigma)\|W_j - W_{j-1}\|_F; \tag{115}$$

$$\lambda_j\tilde{g}(W_{j-1}) - \left[\lambda\tilde{g}(W_j) + \frac{1}{2}\|W_j - W_{j-1}\|_F^2\right] \ge V_j \bullet (W_{j-1} - W_j); \tag{116}$$

$$\bar{M}_j \le \max\{\underline{M}_j, \omega(1+\lambda_0 L_{\tilde{g}})\}; \tag{117}$$

$$\lambda_0 \ge \lambda_j \ge \underline{\lambda}, \tag{118}$$

where $\omega = 4/(1-\chi)$, λ_0 is the initial prox stepsize, and $L_{\tilde{g}}$ and $\underline{\lambda}$ are as in (25) and (26), respectively; moreover, every prox stepsize λ generated within the aforementioned loop is in $[\underline{\lambda}, \lambda_0]$.

Proof. (a) It follows that ψ_s has $(1+\lambda L_{\tilde{g}})$-Lipschitz continuous gradient on $\bar{B}_3$ in view of its definition in (20) and Lemma C.1. The result then follows immediately from the fact that $\lambda \le \lambda_0$.

(b) We first claim that if the loop consisting of steps 1 and 2 of the j-th iteration of ADAP-AIPP stops, then relations (114), (115), (116), and (117) hold. Indeed, assume that the loop consisting of steps 1 and 2 of the j-th iteration of ADAP-AIPP stops. It then follows from the logic within step 1 and 2 of ADAP-AIPP that the last ADAP-FISTA call within the loop stops successfully and outputs triple $(W_j, V_j, \bar{M}_j)$ satisfying (21), which immediately implies that (116) holds. Since (a) implies that $\bar{L} = 1 + \lambda_0 L_{\tilde{g}}$ satisfies relation (96), it follows Proposition B.2(b) with (ψ_s, ψ_n) as in (20), $x_0 = W_{j-1}$, and $L_0 = \underline{M}_j$ that the triple $(W_j, V_j, \bar{M}_j) = (y, v, L)$ satisfies inequality (117) and the following two relations

$$V_j \in \lambda_j[\nabla\tilde{g}(W_j) + \partial\delta_{\bar{B}_1}(W_j)] + W_j - W_{j-1} \tag{119}$$

$$\|V_j\|_F \le \sigma\|W_j - W_{j-1}\|_F. \tag{120}$$

Now, using the definition of R_j in (22), it is easy to see that the inclusion (119) is equivalent to (114) and that the inequality in (120) together with the triangle inequality for norms imply the two inequalities in (115).

We now claim that if step 1 is performed with a prox stepsize $\lambda \le 1/(2L_{\tilde{g}})$ in the j-th iteration, then for every $l > j$, we have that $\lambda_{l-1} = \lambda$ and the l-th iteration performs step 1 only once. To show the claim, assume that $\lambda \le 1/(2L_{\tilde{g}})$. Using this assumption and the fact that Lemma C.1 implies that $\tilde{g}$ is $L_{\tilde{g}}$ weakly convex on $\bar{B}_3$, it is easy to see that the function ψ_s in (20) is strongly convex on $\bar{B}_1 \subseteq \bar{B}_3$ with modulus $1 - \lambda L_{\tilde{g}} \ge 1/2$. Since each ADAP-FISTA call is performed in step 1 of ADAP-AIPP with $\mu = 1/2$, it follows immediately from Proposition B.2(c) with (ψ_s, ψ_n) as in (20) that ADAP-FISTA terminates successfully and outputs a pair (W, V) satisfying $V \in \partial(\psi_s + \psi_n)(W)$. This inclusion, the definitions of (ψ_s, ψ_n), and the definition of subdifferential in (5), then imply that (21) holds. Hence, in view of the termination criteria of step 2 of ADAP-AIPP, it follows that $\lambda_j = \lambda$. It is then easy to see, by the way λ is updated in step 2 of ADAP-AIPP, that λ is not halved in the $(j+1)$-th iteration or any subsequent iteration, hence proving the claim.

It is now straightforward to see that the above two claims, the fact that the initial value of the prox stepsize is equal to λ_0, and the way λ_j is updated in ADAP-AIPP, imply that the lemma holds. □

Lemma C.4. *For any $j \ge 1$, the quantity $\underline{M}_j$ satisfies*

$$\underline{M}_j \le \omega(1+\lambda_0 L_{\tilde{g}}) \tag{121}$$

where $\omega = 4/(1-\chi)$, λ_0 is the initial prox stepsize, and $L_{\tilde{g}}$ is as in (25).

Proof. The result follows from a simple induction argument. The inequality with $j = 1$ is immediate due to the facts that $\underline{\mathrm{M}}_1 = 1$ and $\omega > 1$. Now suppose inequality (121) holds for $j-1$. It then follows from relation (117) and the fact that $\underline{\mathrm{M}}_j \leq \bar{M}_{j-1}$ that

$$\underline{\mathrm{M}}_j \leq \bar{M}_{j-1} \overset{(117)}{\leq} \max\{\underline{\mathrm{M}}_{j-1}, \omega(1+\lambda_0 L_{\tilde{g}})\} = \omega(1+\lambda_0 L_{\tilde{g}}),$$

where the equality is due to the assumption that (121) holds for $j-1$. Hence, Lemma C.4 holds. □

Remark C.5. *It follows from Lemma C.4 that $\underline{\mathrm{M}}_j = \mathcal{O}(1+\lambda_0 L_{\tilde{g}})$ and hence Lemma C.2(a) implies that each ADAP-FISTA call made in step 1 of ADAP-AIPP performs at most*

$$\mathcal{O}_1\left(\sqrt{2\,(1+\lambda_0 L_{\tilde{g}})}\log_1^+(1+\lambda_0 L_{\tilde{g}})\right)$$

iterations/resolvent evaluations where λ_0 is the initial prox stepsize and $L_{\tilde{g}}$ is as in (25).

The following lemma shows that ADAP-AIPP is a descent method.

Lemma C.6. *If j is an iteration index for ADAP-AIPP, then*

$$\frac{\underline{\lambda}}{C_\sigma}\|R_j\|_F^2 \leq \tilde{g}(W_{j-1}) - \tilde{g}(W_j) \tag{122}$$

where C_σ and $\underline{\lambda}$ are as in (26).

Proof. It follows immediately from the first inequality in (115), relation (116), and the definitions of R_j and C_σ in (22) and (26), respectively that:

$$\begin{aligned}
\lambda_j\tilde{g}(W_{j-1}) - \lambda_j\tilde{g}(W_j) &\overset{(116)}{\geq} \frac{1}{2}\|W_j - W_{j-1}\|_F^2 + V_j \bullet (W_{j-1} - W_j) \\
&= \frac{1}{2}\|W_{j-1} - W_j + V_j\|_F^2 - \frac{1}{2}\|V_j\|_F^2 \\
&\overset{(22)}{=} \frac{1}{2}\|\lambda_j R_j\|_F^2 - \frac{1}{2}\|V_j\|_F^2 \\
&\overset{(115)}{\geq} \frac{1}{2}\|\lambda_j R_j\|_F^2 - \frac{\sigma^2}{2(1-\sigma)^2}\|\lambda_j R_j\|_F^2 \\
&= \frac{1-2\sigma}{2(1-\sigma)^2}\|\lambda_j R_j\|_F^2 \overset{(26)}{=} \frac{\|\lambda_j R_j\|_F^2}{C_\sigma}.
\end{aligned} \tag{123}$$

Dividing inequality (123) by λ_j and using relation (118) then imply

$$\tilde{g}(W_{j-1}) - \tilde{g}(W_j) \overset{(123)}{\geq} \frac{\lambda_j}{C_\sigma}\|R_j\|_F^2 \overset{(118)}{\geq} \frac{\underline{\lambda}}{C_\sigma}\|R_j\|_F^2$$

from which the result of the lemma immediately follows. □

We are now ready to give the proof of Proposition 2.4.

Proof of Proposition 2.4. (a) The first statement of (a) follows immediately from the fact that relation (114) and the termination criterion of ADAP-AIPP in its step 3 imply that the pair $(\overline{W}, \overline{R})$ satisfies (24). Assume now by contradiction that (27) does not hold. This implies that there exists an iteration index l such that

$$l > 1 + \frac{C_\sigma}{\underline{\lambda}\bar{\rho}^2}\left[\tilde{g}(\underline{W}) - \tilde{g}(W_l)\right]. \tag{124}$$

As ADAP-AIPP generates l as an iteration index, it does not terminate at the $(l-1)$-th iteration. In view of step 3 of ADAP-AIPP, this implies that $\|R_j\|_F > \bar{\rho}$ for every $j = 1, \ldots, l-1$. Using this conclusion and the fact that $W_0 = \underline{W}$, and summing inequality (122) from 1 to l, we conclude that

$$(l-1)\bar{\rho}^2 < \sum_{j=1}^{l-1} \|R_j\|_F^2 \leq \sum_{j=1}^{l} \|R_j\|_F^2 \overset{(122)}{\leq} \frac{C_\sigma}{\underline{\lambda}} \sum_{j=1}^{l} \tilde{g}(W_{j-1}) - \tilde{g}(W_j) = \frac{C_\sigma}{\underline{\lambda}}\left[\tilde{g}(\underline{W}) - \tilde{g}(W_l)\right]$$

which can be easily seen to contradict (124).

(b) The result follows immediately from (a) and the fact that the number of times λ is divided by 2 in step 2 of ADAP-AIPP is at most $\lceil \log_0^+(\lambda_0/\underline{\lambda})/\log 2 \rceil$. □

D Relaxed Frank-Wolfe Method

Let $\mathbb{E}$ denote a finite-dimensional inner product real vector space with inner product and induced norm denoted by $\langle \cdot, \cdot \rangle$ and $\|\cdot\|$, respectively. Let $\Omega \subseteq \mathbb{E}$ be a nonempty compact convex set with diameter D_Ω. Consider the problem

$$(P) \quad g_* := \min_U \{g(U) : U \in \Omega\} \tag{125}$$

where $g : \mathbb{E} \to \mathbb{R}$ is a convex function that satisfies the following assumption:

(A1) there exists $L_g > 0$ such that

$$g(U') - \ell_g(U'; U) \leq \frac{L_g}{2}\|U' - U\|^2 \quad \forall U, U' \in \Omega. \tag{126}$$

The formal description of the Relaxed FW (RFW) method and its main complexity result for finding a near-optimal solution of (125) are presented below. The proof of the main result is given in the next subsection.

RFW Method

Input: tolerance $\bar{\epsilon} > 0$ and initial point $\tilde{Z}_0 \in \Omega$.
Output: a point $\bar{Z}$.

0. set $k = 1$;

1. find a point $Z_k \in \Omega$ such that

$$g(Z_k) \leq g(\tilde{Z}_{k-1}); \tag{127}$$

2. compute

$$Z_k^F \in \arg\min\{\ell_g(U; Z_k) : U \in \Omega\}, \quad D_k := Z_k - Z_k^F, \quad \epsilon_k := \langle \nabla g(Z_k), D_k \rangle; \tag{128}$$

3. if $\epsilon_k \leq \bar{\epsilon}$, then **stop** and output the point $\bar{Z} = Z_k$; else compute

$$\alpha_k = \arg\min_{\alpha} \{g(Z_k - \alpha D_k) : \alpha \in [0,1]\}\} \tag{129}$$

and set

$$\tilde{Z}_k = Z_k - \alpha_k D_k; \tag{130}$$

4. set $k \leftarrow k+1$ and go to step 1.

Theorem D.1. *For a given tolerance $\bar{\epsilon} > 0$, the RFW method finds a point $\bar{Z} \in \Omega$ such that*

$$0 \in \nabla g(\bar{Z}) + \partial_{\bar{\epsilon}} \delta_\Omega(\bar{Z}) \tag{131}$$

in at most

$$\left\lceil 1 + \frac{4 \max\left\{ g(\tilde{Z}_0) - g_*, \sqrt{\left(g(\tilde{Z}_0) - g_*\right) L_g D_\Omega^2}, L_g D_\Omega^2 \right\}}{\bar{\epsilon}} \right\rceil \tag{132}$$

iterations where L_g is as in (126) *and D_Ω is the diameter of Ω.*

D.1 Proof of Theorem D.1

This subsection is dedicated to proving Theorem D.1.

The following lemma establishes important properties of the iterates Z_k and ϵ_k.

Lemma D.2. *For every $k \geq 1$, the following relations hold:*

$$\epsilon_k \geq g(Z_k) - g_*, \tag{133}$$

$$0 \in \nabla g(Z_k) + \partial_{\epsilon_k} \delta_\Omega(Z_k), \tag{134}$$

where ϵ_k is defined in (128) *and g_* is the optimal value of* (125).

Proof. Suppose $Z' \in \Omega$. It follows from the fact g is convex, the definitions of D_k and ϵ_k in (128), and the way Z_k^F is computed in (128) that

$$\begin{aligned} g(Z_k) - \epsilon_k &\overset{(128)}{=} g(Z_k) + \langle \nabla g(Z_k), Z_k^F - Z_k \rangle \\ &\overset{(128)}{\leq} g(Z_k) + \langle \nabla g(Z_k), Z' - Z_k \rangle \leq g(Z'). \end{aligned} \tag{135}$$

Since (135) holds for any Z' in Ω, it must hold for the minimizer Z^* of (125), and hence $g(Z_k) - \epsilon_k \leq g_*$, which immediately shows relation (133).

It follows from the fact that $Z_k^F \in \arg\min\{\ell_g(U; Z_k) : U \in \Omega\}$ that

$$0 \in \nabla g(Z_k) + \partial \delta_\Omega(Z_k^F).$$

It then follows from the above relation, the definition of ϵ-subdifferential in (5), and the definition of ϵ_k in (128) that inclusion (134) holds. □

The following lemma establishes that the RFW method is a descent method.

Lemma D.3. *Define*

$$\hat{\alpha}_k := \min\left\{1, \frac{\epsilon_k}{L_g D_\Omega^2}\right\} \quad \forall k \geq 1. \tag{136}$$

Then the following statements hold for every $k \geq 1$:

$$\epsilon_k \leq \frac{2}{\hat{\alpha}_k}\left(g(Z_k) - g(\tilde{Z}_k)\right), \tag{137}$$

$$g(Z_{k+1}) \leq g(\tilde{Z}_k) \leq g(Z_k). \tag{138}$$

Proof. It follows from the definitions of ϵ_k, α_k, $\tilde{Z}_k$, and $\hat{\alpha}_k$ in (128), (129), (130), (136), respectively, the fact that $\|D_k\|^2 \leq D_\Omega^2$, and from applying inequality (126) with $U' = Z_k - \hat{\alpha}_k D_k$ and $U = Z_k$ that

$$\begin{aligned} g(\tilde{Z}_k) &\overset{(129),(130)}{\leq} g(Z_k - \hat{\alpha}_k D_k) \overset{(126)}{\leq} g(Z_k) - \hat{\alpha}_k \langle \nabla g(Z_k), D_k\rangle + \frac{\hat{\alpha}_k^2 L_g}{2} D_\Omega^2 \\ &\overset{(128)}{=} g(Z_k) - \hat{\alpha}_k \epsilon_k + \frac{\hat{\alpha}_k^2 L_g}{2} D_\Omega^2 \overset{(136)}{\leq} g(Z_k) - \hat{\alpha}_k \epsilon_k + \frac{\hat{\alpha}_k \epsilon_k}{2} = g(Z_k) - \frac{\hat{\alpha}_k \epsilon_k}{2} \end{aligned}$$

which immediately implies relation (137).

The first inequality in (138) follows immediately from (127). The second inequality in (138) follows immediately from relations (137) and (133). □

The next proposition establishes the convergence rate of the RFW method.

Proposition D.4. *For every $k \geq 2$, the following relations hold:*

$$g(Z_k) - g_* \leq \frac{2}{k-1} \max\left\{g(\tilde{Z}_0) - g_*, L_g D_\Omega^2\right\}, \tag{139}$$

$$\min_{k \leq j < 2k} \epsilon_j \leq \frac{4}{k-1} \max\left\{g(\tilde{Z}_0) - g_*, \sqrt{\left(g(\tilde{Z}_0) - g_*\right) L_g D_\Omega^2}, L_g D_\Omega^2\right\}. \tag{140}$$

Proof. Define $\tilde{\gamma}_0 := g(\tilde{Z}_0) - g_*$ and let $\gamma_j := g(Z_j) - g_*$ for any iteration index j. It then follows from relations (133) and (137) and relation (138) with $k = j$ that the following two relations hold

$$\frac{\hat{\alpha}_j}{2}\gamma_j \overset{(133)}{\leq} \frac{\hat{\alpha}_j}{2}\epsilon_j \overset{(137)}{\leq} g(Z_j) - g(\tilde{Z}_j) \overset{(138)}{\leq} g(Z_j) - g(Z_{j+1}) = \gamma_j - \gamma_{j+1} \tag{141}$$

$$\gamma_{j+1} \overset{(138)}{\leq} \gamma_j \overset{(133)}{\leq} \epsilon_j. \tag{142}$$

Hence, using relations (141) and (142), relation (127) with $k = 1$, and the expression for $\hat{\alpha}_j$ in (136), it follows that

$$\frac{1}{\gamma_{j+1}} - \frac{1}{\gamma_j} = \frac{\gamma_j - \gamma_{j+1}}{\gamma_{j+1}\gamma_j} \overset{(141)}{\geq} \frac{\hat{\alpha}_j \gamma_j}{2\gamma_{j+1}\gamma_j} \overset{(136)}{=} \frac{1}{2\gamma_{j+1}} \min\left\{1, \frac{\epsilon_j}{L_g D_\Omega^2}\right\} \tag{143}$$

$$\overset{(142)}{\geq} \frac{1}{2} \min\left\{\frac{1}{\gamma_1}, \frac{1}{L_g D_\Omega^2}\right\} \overset{(127)}{\geq} \frac{1}{2} \min\left\{\frac{1}{\tilde{\gamma}_0}, \frac{1}{L_g D_\Omega^2}\right\}. \tag{144}$$

It follows from summing the inequality in (143) from $j = 1$ to $k - 1$ that

$$\frac{1}{\gamma_k} \geq \frac{1}{\gamma_k} - \frac{1}{\gamma_1} \geq \frac{k-1}{2} \min\left\{\frac{1}{\tilde{\gamma}_0}, \frac{1}{L_g D_\Omega^2}\right\}, \tag{145}$$

which, together, with the definition of γ_k implies relation (139).

It follows from summing the relation in (137) from $j = k$ to $j = 2k + 1$ and relations (138) and (145) that

$$\begin{aligned}\frac{2}{k-1}\max\{\tilde{\gamma}_0, L_g D_\Omega^2\} &\overset{(145)}{\geq} \gamma_k \geq g(Z_k) - g(Z_{2k+1}) \\ &= \sum_{j=k}^{2k} g(Z_j) - g(Z_{j+1}) \overset{(138)}{\geq} \sum_{j=k}^{2k} g(Z_j) - g(\tilde{Z}_j) \overset{(137)}{\geq} \sum_{j=k}^{2k} \frac{\hat{\alpha}_j}{2}\epsilon_j. \end{aligned} \tag{146}$$

It now follows from relation (146) and the definition of $\hat{\alpha}_j$ in (136) that

$$\frac{4}{(k-1)^2}\max\left\{\tilde{\gamma}_0, L_g D_\Omega^2\right\} \overset{(146)}{\geq} \min_{k\leq j\leq 2k} \hat{\alpha}_j\epsilon_j \overset{(136)}{\geq} \min_{k\leq j\leq 2k}\left\{1, \frac{\epsilon_j}{L_g D_\Omega^2}\right\}\min_{k\leq j\leq 2k}\epsilon_j, \tag{147}$$

which implies relation (140) in view of the definition of $\tilde{\gamma}_0$. □

We are now ready to prove Theorem D.1.

Proof of Theorem D.1. The stopping criterion in step 3 of the RFW method and relation (134) immediately imply that output $\bar{Z}$ satisfies relation (131).

In view of the stopping criterion in step 3 of the RFW method, the iteration complexity result in (132) follows immediately from relation (140). □

E AL method for linearly-constrained convex optimization

This section is dedicated to analyzing the convergence of the augmented Lagrangian framework for solving linearly-constrained convex optimization problems.

Let $\mathbb{E}$ denote an Euclidean space, $\mathcal{A} : \mathbb{E} \to \mathbb{R}^m$ be a linear operator, $b \in \mathbb{R}^m$, $f : \mathbb{E} \to \mathbb{R}$ be a differentiable convex function, and $h : \mathbb{E} \to (-\infty, \infty]$ be a closed proper convex function. Consider the linearly-constrained convex optimization problem

$$\min\{\phi(X) := f(X) + h(X) : \mathcal{A}X = b\}, \tag{148}$$

where the domain of h has finite diameter D_h. The following assumption is also made.

Assumption E.1. *There exists (X_*, p_*) such that*

$$0 \in \nabla f(X_*) + \partial h(X_*) + \mathcal{A}^* p_*, \quad \mathcal{A}X_* - b = 0 \tag{149}$$

Given a previous dual iterate p_{t-1}, the AL framework finds the next primal iterate X_t by

$$X_t \approx \arg\min_X \mathcal{L}_\beta(X; p_{t-1}) \tag{150}$$

where

$$\mathcal{L}_\beta(X; p) := f(X) + h(X) + \langle p, \mathcal{A}X - b\rangle + \frac{\beta}{2}\|\mathcal{A}X - b\|^2 \tag{151}$$

is the augmented Lagrangian function and $\beta > 0$ is a fixed penalty parameter. We assume the existence of a blackbox that inexactly solves such minimization problems as in (150).

Blackbox AL. *Given a pair* $(\hat{\epsilon}_{\rm c}, \hat{\epsilon}_{\rm d}) \in \mathbb{R}^2_+$ *and convex functions* $g : \mathbb{E} \to \mathbb{R}$ *and* $h : \mathbb{E} \to \mathbb{R}$*, the blackbox returns a pair* $(\hat{X}, \hat{R})$ *satisfying*

$$\hat{X} \in \operatorname{dom} h, \quad \hat{R} \in \nabla g(\hat{X}) + \partial_{\hat{\epsilon}_{\rm c}} h(\hat{X}), \qquad \|\hat{R}\| \leq \hat{\epsilon}_{\rm d}. \tag{152}$$

The AL framework is now presented formally below.

AL Framework

Input: $p_0 \in \mathbb{R}^m$, tolerances $\epsilon_{\rm p} > 0$, $\epsilon_{\rm d} \geq 0, \epsilon_{\rm c} \geq 0$, and penalty parameter $\beta > 0$.
Output: triple $(\bar{X}, \bar{p}, \bar{R})$.

0. Set $t = 1$ and

$$\hat{\epsilon}_{\rm c} = \min\{\epsilon_{\rm c},\ \beta\epsilon_{\rm p}^2/6\}, \quad \hat{\epsilon}_{\rm d} = \min\{\epsilon_{\rm d},\ \beta\epsilon_{\rm p}^2/(6D_h)\}; \tag{153}$$

1. Call the Blackbox AL with tolerance pair $(\hat{\epsilon}_{\rm c}, \hat{\epsilon}_{\rm d})$ and functions $h = h$ and $g(\cdot) = \mathcal{L}_\beta(\cdot; p_{t-1})$ and let (X_t, R_t) be its output;

2. Set

$$p_t = p_{t-1} + \beta(\mathcal{A}X_t - b); \tag{154}$$

3. If $\|\mathcal{A}X_t - b\| \leq \epsilon_{\rm p}$, then set $T = t$ and **return** (X_T, p_T, R_T);

4. Set $t \leftarrow t + 1$ and **go to** step **1.**

The following result states the main iteration complexity of the AL framework and establishes the boundedness of its sequence of Lagrange multipliers. The proof of the result is given in the next subsection.

Theorem E.1. *Under Assumption E.1, the following statements about the AL framework hold:*

(a) the AL framework terminates with an iterate $(X_T, p_T, R_T) \in \operatorname{dom} h \times \mathbb{R}^m \times \mathbb{E}$ *such that*

$$R_T \in \nabla f(X_T) + \partial_{\epsilon_{\rm c}} h(X_T) + \mathcal{A}^* p_T, \quad \|R_T\| \leq \epsilon_{\rm d}, \quad \|\mathcal{A}X_T - b\| \leq \epsilon_{\rm p} \tag{155}$$

and

$$T \leq \left\lceil \frac{3\|p_* - p_0\|^2}{\beta^2 \epsilon_{\rm p}^2} \right\rceil; \tag{156}$$

(b) there hold

$$\max_{t \in \{0,\ldots T\}} \|p_t\| \leq \|p_*\| + \sqrt{3\|p_* - p_0\|^2 + 2\beta(D_h\hat{\epsilon}_{\rm d} + \hat{\epsilon}_{\rm c})}, \tag{157}$$

$$\beta^2 \sum_{l=1}^{T} \|\mathcal{A}X_l - b\|^2 \leq 3\|p_* - p_0\|^2 + 2\beta(D_h\hat{\epsilon}_{\rm d} + \hat{\epsilon}_{\rm c}), \tag{158}$$

where p_* *is an optimal Lagrange multiplier and* $\hat{\epsilon}_{\rm c}$ *and* $\hat{\epsilon}_{\rm d}$ *are as in* (153).

E.1 Proof of Theorem E.1

This subsection is dedicated to proving Theorem E.1. The proof relies on the following two preliminary lemmas.

Lemma E.2. *For any $t \geq 1$, the following relation holds*

$$\langle \mathcal{A}X_t - b, p_* - p_t \rangle \geq \langle R_t, X_* - X_t \rangle - \hat{\epsilon}_{\mathrm{c}}, \tag{159}$$

where (X_*, p_*) is an optimal primal-dual pair of (148).

Proof. Since (X_*, p_*) is an optimal primal-dual pair, it follows that

$$0 \in \partial\phi(X_*) + \mathcal{A}^* p_*, \quad \mathcal{A}X_* = b, \tag{160}$$

where ϕ is as in (148). Relation (152) implies that

$$R_t \in \nabla f(X_t) + \partial_{\hat{\epsilon}_{\mathrm{c}}} h(X_t) + \mathcal{A}^* p_t \subseteq \partial_{\hat{\epsilon}_{\mathrm{c}}} \phi(X_t) + \mathcal{A}^* p_t. \tag{161}$$

It follows from relations (160) and (161) and the definition of $\hat{\epsilon}_{\mathrm{c}}$-subdifferential that

$$\begin{aligned} \phi(X_t) - \phi(X_*) &\overset{(160)}{\geq} \langle -\mathcal{A}^* p_*, X_t - X_* \rangle \\ \phi(X_*) - \phi(X_t) &\overset{(161)}{\geq} \langle R_t - \mathcal{A}^* p_t, X_* - X_t \rangle - \hat{\epsilon}_{\mathrm{c}}. \end{aligned}$$

Adding the two above relations implies that

$$\langle \mathcal{A}^*(p_* - p_t), X_t - X_* \rangle \geq \langle R_t, X_* - X_t \rangle - \hat{\epsilon}_{\mathrm{c}}. \tag{162}$$

Relations (160) and (162) then imply that

$$\langle \mathcal{A}X_t - b, p_* - p_t \rangle \overset{(160)}{=} \langle \mathcal{A}(X_t - X_*), p_* - p_t \rangle = \langle X_t - X_*, \mathcal{A}^*(p_* - p_t) \rangle \overset{(162)}{\geq} \langle R_t, X_* - X_t \rangle - \hat{\epsilon}_{\mathrm{c}},$$

from which the result immediately follows. □

Lemma E.3. *For any iteration index t of the AL framework, there holds:*

$$\beta^2 \sum_{l=1}^{t} \|\mathcal{A}X_l - b\|^2 \leq \|p_* - p_0\|^2 - \|p_* - p_t\|^2 + 2\beta t(D_h \hat{\epsilon}_{\mathrm{d}} + \hat{\epsilon}_{\mathrm{c}}). \tag{163}$$

Proof. Let t be an iteration index of the AL framework and suppose $l \leq t$. By completing the square and using relation (154), it follows that

$$\begin{aligned} \|p_* - p_{l-1}\|^2 - \|p_* - p_l\|^2 &= \|p_{l-1} - p_l\|^2 + 2\langle p_l - p_{l-1}, p_* - p_l \rangle \\ &\overset{(154)}{=} \beta^2 \|\mathcal{A}X_l - b\|^2 + 2\beta \langle \mathcal{A}X_l - b, p_* - p_l \rangle. \end{aligned} \tag{164}$$

Moreover, relation (159), the definition of D_h, the Cauchy-Schwarz inequality, and the fact that the Blackbox is called in step 1 with tolerance $\hat{\epsilon}_d$, imply that

$$2\beta \langle \mathcal{A}X_l - b, p_* - p_l \rangle \overset{(159)}{\geq} 2\beta \langle R_l, X_* - X_l \rangle - 2\beta \hat{\epsilon}_{\mathrm{c}} \geq -2\beta D_h \hat{\epsilon}_{\mathrm{d}} - 2\beta \hat{\epsilon}_{\mathrm{c}}. \tag{165}$$

Combining relations (164) and (165), we then conclude that

$$\|p_* - p_{l-1}\|^2 - \|p_* - p_l\|^2 \geq \beta^2 \|\mathcal{A}X_l - b\|^2 - 2\beta D_h \hat{\epsilon}_{\mathrm{d}} - 2\beta \hat{\epsilon}_{\mathrm{c}}. \tag{166}$$

The conclusion of the lemma now follows by summing relation (166) from $l = 1$ to t. □

We are now ready to prove Theorem E.1

Proof of Theorem E.1. (a) Let t be an iteration index of the AL framework. The fact that the Blackbox AL is called in step 1 with inputs g and $(\hat{\epsilon}_{\rm c}, \hat{\epsilon}_{\rm d})$ implies that its output (X_t, R_t) satisfies that $\|R_t\| \leq \hat{\epsilon}_{\rm d}$ and also

$$\begin{aligned} R_t \in \nabla g(X_t) + \partial_{\hat{\epsilon}_{\rm c}} h(X_t) &= \nabla f(X_t) + \mathcal{A}^*(p_{t-1} + \beta(\mathcal{A}X_t - b)) + \partial_{\hat{\epsilon}_{\rm c}} h(X_t) \\ &= \nabla f(X_t) + \partial_{\hat{\epsilon}_{\rm c}} h(X_t) + \mathcal{A}^* p_t. \end{aligned}$$

Since $\hat{\epsilon}_{\rm c} \leq \epsilon_{\rm c}$ and $\hat{\epsilon}_{\rm d} \leq \epsilon_{\rm d}$, it follows that

$$R_t \in \nabla f(X_t) + \partial_{\epsilon_{\rm c}} h(X_t) + \mathcal{A}^*(p_t), \qquad \|R_t\| \leq \epsilon_{\rm d}.$$

Since the above relations hold for any iteration index t, the output (X_T, p_T, R_T) of the AL framework satisfies the first two relations in (155). It remains to show that the AL framework terminates and that its last iteration index T satisfies (156). Suppose by contradiction that the AL framework generates an iteration index $\hat{t}$ satisfying

$$\hat{t} > \left\lceil \frac{3\|p_* - p_0\|^2}{\beta^2 \epsilon_{\rm p}^2} \right\rceil. \tag{167}$$

In view of the stopping criterion of step 3 of the AL framework, this implies that $\|\mathcal{A}X_t - b\| > \epsilon_{\rm p}$ for every $t = 1, \ldots \hat{t} - 1$. Using this conclusion, relation (163) with $t = \hat{t} - 1$, and the definitions of $\hat{\epsilon}_{\rm c}$ and $\hat{\epsilon}_{\rm d}$ in (153), it follows that

$$\begin{aligned} (\hat{t}-1)\epsilon_p^2 < \sum_{l=1}^{\hat{t}-1} \|\mathcal{A}X_l - b\|^2 &\overset{(163)}{\leq} \frac{\|p_* - p_0\|^2}{\beta^2} + (\hat{t}-1)\frac{2\beta(D_h\hat{\epsilon}_{\rm d} + \hat{\epsilon}_{\rm c})}{\beta^2} \\ &\overset{(153)}{\leq} \frac{\|p_* - p_0\|^2}{\beta^2} + (\hat{t}-1)\frac{2\epsilon_p^2}{3} \end{aligned} \tag{168}$$

which clearly contradicts the bound on $\hat{t}$ in (167). Hence, in view of this conclusion and the termination criterion in step 3, the AL framework must terminate with final iteration index T satisfying (156) and output (X_T, p_T, R_T) satisfying the third relation in (155).

(b) Let $t \leq T$ where T is the final iteration index of the AL framework. It then follows from taking square root of relation (163) and triangle inequality that

$$\|p_t\| \overset{(163)}{\leq} \|p_*\| + \sqrt{\|p_* - p_0\|^2 + 2\beta t(D_h\hat{\epsilon}_{\rm d} + \hat{\epsilon}_{\rm c})} \leq \|p_*\| + \sqrt{\|p_* - p_0\|^2 + 2\beta T(D_h\hat{\epsilon}_{\rm d} + \hat{\epsilon}_{\rm c})}. \tag{169}$$

The fact that T satisfies relation (156) and the definitions of $\hat{\epsilon}_{\rm c}$ and $\hat{\epsilon}_{\rm d}$ in (153) then imply that

$$2\beta T(D_h\hat{\epsilon}_{\rm d} + \hat{\epsilon}_{\rm c}) \overset{(156)}{\leq} \frac{6(D_h\hat{\epsilon}_{\rm d} + \hat{\epsilon}_{\rm c})}{\beta\epsilon_p^2}\|p_* - p_0\|^2 + 2\beta(D_h\hat{\epsilon}_{\rm d} + \hat{\epsilon}_{\rm c}) \overset{(153)}{\leq} 2\|p_* - p_0\|^2 + 2\beta(D_h\hat{\epsilon}_{\rm d} + \hat{\epsilon}_{\rm c}). \tag{170}$$

Relation (157) then immediately follows from combining relations (169) and (170).

It follows from relation (163) with $t = T$ that

$$\beta^2 \sum_{l=1}^{T} \|\mathcal{A}X_l - b\|^2 \overset{(163)}{\leq} \|p_* - p_0\|^2 + 2\beta T(D_h\hat{\epsilon}_{\rm d} + \hat{\epsilon}_{\rm c}). \tag{171}$$

Combining relations (170) and (171) then immediately implies inequality (158). □

References

[1] David A. Bader, Henning Meyerhenke, Peter Sanders, and Dorothea Wagner, editors. *Graph Partitioning and Graph Clustering, 10th DIMACS Implementation Challenge Workshop, Georgia Institute of Technology, Atlanta, GA, USA, February 13-14, 2012. Proceedings*, volume 588 of *Contemporary Mathematics*. American Mathematical Society, 2013.

[2] Alexander I. Barvinok. Problems of distance geometry and convex properties of quadratic maps. *Discrete & Computational Geometry*, 13:189–202, 1995.

[3] Srinadh Bhojanapalli, Nicolas Boumal, Prateek Jain, and Praneeth Netrapalli. Smoothed analysis for low-rank solutions to semidefinite programs in quadratic penalty form. In *Conference On Learning Theory*, pages 3243–3270. PMLR, 2018.

[4] Nicolas Boumal, Vlad Voroninski, and Afonso Bandeira. The non-convex burer-monteiro approach works on smooth semidefinite programs. *Advances in Neural Information Processing Systems*, 29, 2016.

[5] Nicolas Boumal, Vladislav Voroninski, and Afonso S Bandeira. Deterministic guarantees for burer-monteiro factorizations of smooth semidefinite programs. *Communications on Pure and Applied Mathematics*, 73(3):581–608, 2020.

[6] Samuel Burer and Renato DC Monteiro. A projected gradient algorithm for solving the maxcut SDP relaxation. *Optimization methods and Software*, 15(3-4):175–200, 2001.

[7] Samuel Burer and Renato DC Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical programming*, 95(2):329–357, 2003.

[8] Samuel Burer and Renato DC Monteiro. Local minima and convergence in low-rank semidefinite programming. *Mathematical programming*, 103(3):427–444, 2005.

[9] M.L. Overton C. Helmberg and F. Rendl. The spectral bundle method with second-order information. *Optimization Methods and Software*, 29(4):855–876, 2014.

[10] Emmanuel Candes and Benjamin Recht. Exact matrix completion via convex optimization. *Communications of the ACM*, 55(6):111–119, 2012.

[11] Emmanuel J Candes, Xiaodong Li, and Mahdi Soltanolkotabi. Phase retrieval from coded diffraction patterns. *Applied and Computational Harmonic Analysis*, 39(2):277–299, 2015.

[12] Emmanuel J Candes, Thomas Strohmer, and Vladislav Voroninski. Phaselift: Exact and stable signal recovery from magnitude measurements via convex programming. *Communications on Pure and Applied Mathematics*, 66(8):1241–1274, 2013.

[13] Y. Carmon, J. C. Duchi, O. Hinder, and A. Sidford. Accelerated methods for nonconvex optimization. *SIAM J. Optim.*, 28(2):1751–1772, 2018.

[14] Diego Cifuentes. On the Burer–Monteiro method for general semidefinite programs. *Optimization Letters*, 15(6):2299–2309, 2021.

[15] Diego Cifuentes and Ankur Moitra. Polynomial time guarantees for the Burer-Monteiro method. *Advances in Neural Information Processing Systems*, 35:23923–23935, 2022.

[16] Andrew R Conn, Nicholas IM Gould, and Philippe Toint. A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Numerical Analysis*, 28(2):545–572, 1991.

[17] Timothy A. Davis and Yifan Hu. The university of florida sparse matrix collection. *ACM Trans. Math. Softw.*, 38(1), dec 2011.

[18] Qi Deng, Qing Feng, Wenzhi Gao, Dongdong Ge, Bo Jiang, Yuntian Jiang, Jingsong Liu, Tianhao Liu, Chenyu Xue, Yinyu Ye, et al. New developments of ADMM-based interior point methods for linear programming and conic programming. *arXiv preprint arXiv:2209.01793*, 2022.

[19] Lijun Ding and Benjamin Grimmer. Revisiting spectral bundle methods: Primal-dual (sub)linear convergence rates. *SIAM Journal on Optimization*, 33(2):1305–1332, 2023.

[20] Lijun Ding, Alp Yurtsever, Volkan Cevher, Joel A Tropp, and Madeleine Udell. An optimal-storage approach to semidefinite programming using approximate complementarity. *SIAM Journal on Optimization*, 31(4):2695–2725, 2021.

[21] Murat A Erdogdu, Asuman Ozdaglar, Pablo A Parrilo, and Nuri Denizcan Vanli. Convergence rate of block-coordinate maximization Burer–Monteiro method for solving large SDPs. *Mathematical Programming*, 195(1-2):243–281, 2022.

[22] Maryam Fazel. *Matrix rank minimization with applications*. PhD thesis, PhD thesis, Stanford University, 2002.

[23] Robert M Freund, Paul Grigas, and Rahul Mazumder. An extended Frank–Wolfe method with "in-face" directions, and its application to low-rank matrix completion. *SIAM Journal on optimization*, 27(1):319–346, 2017.

[24] Mituhiro Fukuda, Masakazu Kojima, Kazuo Murota, and Kazuhide Nakata. Exploiting sparsity in semidefinite programming via matrix completion i: General framework. *SIAM Journal on Optimization*, 11(3):647–674, 2001.

[25] Michael Garstka, Mark Cannon, and Paul Goulart. Cosmo: A conic operator splitting method for convex conic problems. *Journal of Optimization Theory and Applications*, 190(3):779–810, 2021.

[26] Robert Grone, Charles R. Johnson, Eduardo M. Sá, and Henry Wolkowicz. Positive definite completions of partial hermitian matrices. *Linear Algebra and its Applications*, 58:109–124, 1984.

[27] Martin Grötschel, László Lovász, and Alexander Schrijver. Polynomial algorithms for perfect graphs. In *North-Holland mathematics studies*, volume 88, pages 325–356. Elsevier, 1984.

[28] Zaid Harchaoui, Anatoli Juditsky, and Arkadi Nemirovski. Conditional gradient algorithms for norm-regularized smooth convex optimization. *Math. Program.*, 152(1-2):75–112, 2015.

[29] Elad Hazan. Sparse approximate solutions to semidefinite programs. In *Latin American symposium on theoretical informatics*, pages 306–316. Springer, 2008.

[30] C. Helmberg and K.C. Kiwiel. A spectral bundle method with bounds. *Math. Program.*, 93(2):173–194, 2002.

[31] C. Helmberg and F. Rendl. A spectral bundle method for semidefinite programming. *SIAM Journal on Optimization*, 10(3):673–696, 2000.

[32] Steven Homer and Marcus Peinado. Design and performance of parallel and distributed approximation algorithms for maxcut. *Journal of Parallel and Distributed Computing*, 46(1):48–61, 1997.

[33] Wen Huang, Kyle A Gallivan, and Xiangxiong Zhang. Solving PhaseLift by low-rank Riemannian optimization methods for complex semidefinite constraints. *SIAM Journal on Scientific Computing*, 39(5):B840–B859, 2017.

[34] Martin Jaggi and Marek Sulovský. A simple algorithm for nuclear norm regularized problems. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 471–478, Madison, WI, USA, 2010. Omnipress.

[35] Michel Journée, Francis Bach, P-A Absil, and Rodolphe Sepulchre. Low-rank optimization on the cone of positive semidefinite matrices. *SIAM Journal on Optimization*, 20(5):2327–2351, 2010.

[36] W. Kong, J.G. Melo, and R.D.C. Monteiro. Complexity of a quadratic penalty accelerated inexact proximal point method for solving linearly constrained nonconvex composite programs. *SIAM J. Optim.*, 29(4):2566–2593, 2019.

[37] W. Kong, J.G. Melo, and R.D.C. Monteiro. An efficient adaptive accelerated inexact proximal point method for solving linearly constrained nonconvex composite problems. *Comput. Optim. Appl.*, 76(2):305–346, 2019.

[38] Brian Kulis, Arun C Surendran, and John C Platt. Fast low-rank semidefinite programming for embedding and clustering. In *Artificial Intelligence and Statistics*, pages 235–242. PMLR, 2007.

[39] Sören Laue. A hybrid algorithm for convex semidefinite optimization. In *Proceedings of the 32nd International Conference on Machine Learning, ICML*, 2012.

[40] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[41] László Lovász. On the Shannon capacity of a graph. *IEEE Transactions on Information theory*, 25(1):1–7, 1979.

[42] Ramtin Madani, Abdulrahman Kalbat, and Javad Lavaei. ADMM for sparse semidefinite programming with applications to optimal power flow problem. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 5932–5939. IEEE, 2015.

[43] Anirudha Majumdar, Georgina Hall, and Amir Ali Ahmadi. Recent scalability improvements for semidefinite programming with applications in machine learning, control, and robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:331–360, 2020.

[44] Song Mei, Theodor Misiakiewicz, Andrea Montanari, and Roberto Imbuzeiro Oliveira. Solving SDPs for synchronization and MaxCut problems via the G rothendieck inequality. In *Conference on learning theory*, pages 1476–1515. PMLR, 2017.

[45] F. Oustry. A second-order bundle method to minimize the maximum eigenvalue function. *Math. Program.*, 89(1):1–33, 2000.

[46] Brendan O'donoghue, Eric Chu, Neal Parikh, and Stephen Boyd. Conic optimization via operator splitting and homogeneous self-dual embedding. *Journal of Optimization Theory and Applications*, 169:1042–1068, 2016.

[47] C. Paquette, H. Lin, D. Drusvyatskiy, J. Mairal, and Z. Harchaoui. Catalyst for gradient-based nonconvex optimization. In *AISTATS 2018-21st International Conference on Artificial Intelligence and Statistics*, pages 1–10, 2018.

[48] Gábor Pataki. On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues. *Mathematics of operations research*, 23(2):339–358, 1998.

[49] Thomas Pumir, Samy Jelassi, and Nicolas Boumal. Smoothed analysis of the low-rank approach for smooth semidefinite programs. *Advances in Neural Information Processing Systems*, 31, 2018.

[50] Nikhil Rao, Parikshit Shah, and Stephen Wright. Conditional gradient with enhancement and truncation for atomic-norm regularization. In *NIPS workshop on Greedy Algorithms*. Citeseer, 2013.

[51] James Renegar. Accelerated first-order methods for hyperbolic programming. *Mathematical Programming*, 173(1-2):1–35, 2019.

[52] David M Rosen. Scalable low-rank semidefinite programming for certifiably correct machine perception. In *Algorithmic Foundations of Robotics XIV: Proceedings of the Fourteenth Workshop on the Algorithmic Foundations of Robotics 14*, pages 551–566. Springer, 2021.

[53] David M Rosen, Luca Carlone, Afonso S Bandeira, and John J Leonard. SE-Sync: A certifiably correct algorithm for synchronization over the special Euclidean group. *The International Journal of Robotics Research*, 38(2-3):95–125, 2019.

[54] Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015.

[55] Alexander Shapiro. Rank-reducibility of a symmetric matrix and sampling theory of minimum trace factor analysis. *Psychometrika*, 47:187–199, 1982.

[56] Nimita Shinde, Vishnu Narayanan, and James Saunderson. Memory-efficient structured convex optimization via extreme point sampling. *SIAM Journal on Mathematics of Data Science*, 3(3):787–814, 2021.

[57] A. Sujanani and R.D.C. Monteiro. An adaptive superfast inexact proximal augmented Lagrangian method for smooth nonconvex composite optimization problems. *J. Scientific Computing*, 97(2), 2023.

[58] Lieven Vandenberghe, Martin S Andersen, et al. Chordal graphs and semidefinite optimization. *Foundations and Trends® in Optimization*, 1(4):241–433, 2015.

[59] Irene Waldspurger and Alden Waters. Rank optimality for the Burer–Monteiro factorization. *SIAM journal on Optimization*, 30(3):2577–2602, 2020.

[60] Alex L Wang and Fatma Kilinc-Karzan. Accelerated first-order methods for a class of semidefinite programs. *arXiv preprint arXiv:2206.00224*, 2022.

[61] J. Wang and L. Hu. Solving Low-Rank Semidefinite Programs via Manifold Optimization. *Available on arXiv:2303.01722*, 2023.

[62] Yifei Wang, Kangkang Deng, Haoyang Liu, and Zaiwen Wen. A decomposition augmented lagrangian method for low-rank semidefinite programming. *SIAM Journal on Optimization*, 33(3):1361–1390, 2023.

[63] Liuqin Yang, Defeng Sun, and Kim-Chuan Toh. SDPNAL+: a majorized semismooth Newton-CG augmented L agrangian method for semidefinite programming with nonnegative constraints. *Mathematical Programming Computation*, 7(3):331–366, 2015.

[64] Yinyu Ye. Gset dataset of random graphs. `https://www.cise.ufl.edu/research/sparse/matrices/Gset`, 2003.

[65] Alp Yurtsever, Olivier Fercoq, and Volkan Cevher. A conditional-gradient-based augmented lagrangian framework. In *International Conference on Machine Learning*, pages 7272–7281. PMLR, 2019.

[66] Alp Yurtsever, Ya-Ping Hsieh, and Volkan Cevher. Scalable convex methods for phase retrieval. In *2015 IEEE 6th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pages 381–384. IEEE, 2015.

[67] Alp Yurtsever, Joel A Tropp, Olivier Fercoq, Madeleine Udell, and Volkan Cevher. Scalable semidefinite programming. *SIAM Journal on Mathematics of Data Science*, 3(1):171–200, 2021.

[68] Xin-Yuan Zhao, Defeng Sun, and Kim-Chuan Toh. A newton-cg augmented lagrangian method for semidefinite programming. *SIAM Journal on Optimization*, 20(4):1737–1765, 2010.

[69] Yang Zheng, Giovanni Fantuzzi, Antonis Papachristodoulou, Paul Goulart, and Andrew Wynn. Fast ADMM for semidefinite programs with chordal sparsity. In *2017 American Control Conference (ACC)*, pages 3335–3340. IEEE, 2017.