SIAM J. Optim.
Vol. 31, No. 4, pp. 2695–2725

AN OPTIMAL-STORAGE APPROACH TO SEMIDEFINITE PROGRAMMING USING APPROXIMATE COMPLEMENTARITY[*]

LIJUN DING[†], ALP YURTSEVER[‡], VOLKAN CEVHER[§], JOEL A. TROPP[¶], AND MADELEINE UDELL[†]

Abstract. This paper develops a new storage-optimal algorithm that provably solves almost all semidefinite programs (SDPs). This method is particularly effective for weakly constrained SDPs under appropriate regularity conditions. The key idea is to formulate an approximate complementarity principle: Given an approximate solution to the dual SDP, the primal SDP has an approximate solution whose range is contained in the eigenspace with small eigenvalues of the dual slack matrix. For weakly constrained SDPs, this eigenspace has very low dimension, so this observation significantly reduces the search space for the primal solution. This result suggests an algorithmic strategy that can be implemented with minimal storage: (1) solve the dual SDP approximately; (2) compress the primal SDP to the eigenspace with small eigenvalues of the dual slack matrix; (3) solve the compressed primal SDP. The paper also provides numerical experiments showing that this approach is successful for a range of interesting large-scale SDPs.

Key words. semidefinite programs, storage optimality, low rank, complementary slackness, primal recovery

AMS subject classifications. 90C06, 90C22, 49M05

DOI. 10.1137/19M1244603

1. Introduction.

Consider a semidefinite program (SDP) in the standard form

$$\text{(P)} \qquad \begin{array}{ll} \text{minimize} & \mathbf{tr}(CX) \\ \text{subject to} & \mathcal{A}X = b \quad \text{and} \quad X \succeq 0. \end{array}$$

The primal variable is the symmetric, positive-semidefinite (psd) matrix $X \in \mathbf{S}^n_+$. The problem data comprises a symmetric (but possibly indefinite) objective matrix $C \in \mathbf{S}^n$, a right-hand-side $b \in \mathbf{R}^m$, and a linear map $\mathcal{A} : \mathbf{R}^{n\times n} \to \mathbf{R}^m$ with rank m, which can be expressed explicitly as $[\mathcal{A}H]_i = \mathbf{tr}(A_i H), i = 1, \dots, m$, for some symmetric $A_i \in \mathbf{S}^n$ and any $H \in \mathbf{R}^{n\times n}$. The notation $\mathbf{tr}(\cdot)$ stands for the trace operation: $\mathbf{tr}(A) = \sum_{i=1}^n a_{ii}$ for any $A \in \mathbf{R}^{n\times n}$ with (i,j)th entry $a_{ij} \in \mathbf{R}$.

[*]Received by the editors February 14, 2019; accepted for publication (in revised form) June 10, 2021; published electronically October 28, 2021.
https://doi.org/10.1137/19M1244603

Funding: The first and fifth authors were supported in part by DARPA award FA8750-17-2-0101. The second author received support through Early Postdoc.Mobility Fellowship P2ELP2 187955 from the Swiss National Science Foundation (SNSF) and partial postdoctoral support from NSF-CAREER grant IIS-1846088. The third author received funding for this project from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement 725594-time-data) and from the SNSF under grant 200021_178865/1. The fourth author was supported in part by ONR awards N-00014-11-1002, N-00014-17-12146, and N-00014-18-12363. Parts of this research were conducted while the fifth author was in residence at the Simons Institute and the second author was at EPFL.

[†]Operations Research and Information Engineering (ORIE), Cornell University, Ithaca, NY 14850 USA (ld446@cornell.edu, udell@cornell.edu).
[‡]Laboratory for Information and Decision Systems (LIDS), Massachusetts Institute of Technology, Cambridge, MA 02139 USA (alpy@mit.edu).
[§]Laboratory for Information and Inference Systems (LIONS), Ecole Polytechnique Fédérale de Lausanne (EPFL), 1015, Lausanne, Switzerland (volkan.cevher@epfl.ch).
[¶]Computing and Mathematical Sciences, California Institute of Technology, Pasadena, CA 91125-5000 USA (jtropp@cms.caltech.edu).

SDPs form a class of convex optimization problems with remarkable modeling power. But SDPs are challenging to solve because they involve a matrix variable $X \in \mathbf{S}^n_+ \subset \mathbf{R}^{n\times n}$ whose dimension n can rise into the millions or billions. For example, when using a matrix completion SDP in a recommender system, n is the number of users and products; when using a phase retrieval SDP to visualize a biological sample, n is the number of pixels in the recovered image. In these applications, most algorithms are prohibitively expensive because their storage costs are quadratic in n.

How much memory should be required to solve this problem? Any algorithm must be able to query the problem data and to report a representation of the solution. Informally, we say that an algorithm uses *optimal storage* if the working storage is no more than a constant multiple of the storage required for these operations [76]. (See subsection 1.2 for a formal definition.)

It is not obvious how to develop storage-optimal SDP algorithms. To see why, recall that all *weakly constrained* SDPs ($m = \mathcal{O}(n)$) admit low-rank solutions [8, 55], which can be expressed compactly in factored form. For these problems, a storage-optimal algorithm cannot even instantiate the matrix variable! One natural idea is to introduce an explicit low-rank factorization of the primal variable X and to minimize the problem over the factors [16].

Methods built from this idea provably work when the size of the factors is sufficiently large [12]. However, recent work [69] shows that they cannot provably solve all SDPs with optimal storage; see section 2.

In contrast, this paper develops a new algorithm that provably solves all *regular* SDPs, i.e., SDPs with strong duality, unique primal and dual solutions, and strict complementarity. These standard conditions hold not only generically [5, Definition 19] but also in many applications [25]. We defer the detailed description of these conditions to subsection 1.1.

Our method begins with the Lagrange dual of the primal SDP (P),

$$\text{(D)} \qquad \begin{array}{ll} \text{maximize} & b^\top y \\ \text{subject to} & C - \mathcal{A}^\top y \succeq 0 \end{array}$$

with dual variable $y \in \mathbf{R}^m$. The vector $b^\top$ is the transpose of b, and the linear map $\mathcal{A}^\top : \mathbf{R}^m \to \mathbf{R}^{n\times n}$ is the adjoint of the linear map $\mathcal{A}$. Note the range of $\mathcal{A}^\top$ is in $\mathbf{S}^n$ because C and A_is are symmetric. It is straightforward to compute an approximate solution to the dual SDP (D) with optimal storage using methods described in section 6. The challenge is to recover a primal solution from the approximate dual solution.

To meet this challenge, we develop a new *approximate complementarity principle* that holds for the regular SDP: Given an approximate dual solution y, we prove that there is a primal approximate solution X whose range is contained in the eigenspace with small eigenvalues of the dual slack matrix $C - \mathcal{A}^\top y$. This principle suggests an algorithm: we solve the primal SDP by searching over matrices with the appropriate range. This recovery problem is a (much smaller) SDP that can be solved with optimal storage.

1.1. Regularity assumptions.

First, assume that the primal (P) has a solution, say, $X_\star$, and the dual (D) has a *unique* solution $y_\star$. We require that strong duality holds:

$$p_\star := \mathbf{tr}(CX_\star) = b^\top y_\star =: d_\star. \qquad (1.1)$$

The condition (1.1) follows, for example, from Slater's constraint qualification.

Strong duality and feasibility imply that the solution $X_\star$ and the dual slack matrix $C - \mathcal{A}^\top y_\star$ satisfy the complementary slackness condition:

$$X_\star(C - \mathcal{A}^\top y_\star) = 0, \tag{1.2}$$

which implies that

$$\mathbf{rank}(X_\star) + \mathbf{rank}(C - \mathcal{A}^\top y_\star) \le n.$$

To ensure that the SDP is not degenerate, we make the stronger assumption that every solution pair $(X_\star, y_\star)$ satisfies the stronger *strict complementarity* condition:

$$\mathbf{rank}(X_\star) + \mathbf{rank}(C - \mathcal{A}^\top y_\star) = n. \tag{1.3}$$

Note that these assumptions ensure that *all* solutions have the same rank and therefore that the primal solution is actually *unique* [43, Corollary 2.5]. In particular, the rank $r_\star$ of the solution $X_\star$ satisfies the Barvinok–Pataki bound $\binom{r_\star+1}{2} \le m$.

To summarize, all results in this paper hold under the regularity assumptions: primal attainability, dual uniqueness, strong duality, and strict complementarity. These conditions hold for any SDP with attainable primal and dual solutions outside of a set of Lebesgue measure 0 [5]. Many structured SDP, such as those arising from relaxation of combinatorial problems like MaxCut, are also regular [25].

1.2. Optimal storage. Following [76], let us quantify the storage necessary to solve every SDP (P) that satisfies our assumptions in subsection 1.1 and that admits a solution with rank $r_\star$.

First, it is easy to see that $\Theta(nr_\star)$ numbers are sufficient to represent the rank-$r_\star$ solution in factored form. This cost is also necessary because *every* rank-$r_\star$ matrix is the solution to some SDP from our problem class.

To hide the internal complexity of the optimization problem (P), we will interact with the problem data using *data access oracles*. Suppose we can perform any of the following operations on arbitrary vectors $u, v \in \mathbf{R}^n$ and $y \in \mathbf{R}^m$:

$$u \mapsto Cu \quad \text{and} \quad (u,v) \mapsto \mathcal{A}(uv^\top) \quad \text{and} \quad (u,y) \mapsto (\mathcal{A}^\top y)u. \tag{1.4}$$

These oracles enjoy simple implementations in many concrete applications. The input and output of these operations clearly involve storing $\Theta(m+n)$ numbers.

In summary, any method that uses these data access oracles to solve every SDP from our class must store $\Omega(m+nr_\star)$ numbers. We say a method uses *optimal storage* if it provably solves the SDP using working storage $\mathcal{O}(m+nr_\star)$.

For many interesting problems, the number m of constraints is proportional to the dimension n. Moreover, the rank $r_\star$ of the solution is constant or logarithmic in n. In this case, a storage-optimal algorithm has working storage $\tilde{\mathcal{O}}(n)$, where the tilde suppresses log-like factors.

Remark 1.1 (applications). The algorithmic framework we propose is most useful when the problem data has an efficient representation and the three operations (1.4) can be implemented with low arithmetic cost, as when the matrix C and the linear map $\mathcal{A}$ are sparse or structured. This favorable situation is common in practice, for example, in the MaxCut relaxation [33], matrix completion [61], phase retrieval [20, 68], and community detection [48]. See [57] for some other examples. We expect that (most of) the regularity assumptions used in this paper are satisfied for (most of) these problems; indeed, [25] verifies regularity for many of these problems.

1.3. From strict complementarity to storage optimality. Suppose that we have computed the *exact* unique dual solution $y_\star$. Complementary slackness (1.2) and strict complementarity (1.3) ensure that

$$\mathbf{range}(X_\star) \subset \mathbf{null}(C - \mathcal{A}^\top y_\star) \quad \text{and} \quad \dim(\mathbf{null}(C - \mathcal{A}^\top y_\star)) = \mathbf{rank}(X_\star).$$

Hence the slack matrix identifies the range of the primal solution.

Let $r_\star$ be the rank of the primal solution. Construct an orthonormal matrix $V_\star \in \mathbf{R}^{n\times r_\star}$ whose columns span $\mathbf{null}(C - \mathcal{A}^\top y_\star)$. The compression of the primal problem (P) to this subspace is

$$\begin{array}{ll} \text{minimize} & \mathbf{tr}(CV_\star S V_\star{}^\top) \\ \text{subject to} & \mathcal{A}(V_\star S V_\star{}^\top) = b \quad \text{and} \quad S \succeq 0. \end{array} \tag{1.5}$$

The variable $S \in \mathbf{S}_+^{r_\star}$ is a low-dimensional matrix when $r_\star$ is small. If $S_\star$ is a solution to (1.5), then $X_\star = V_\star S_\star V_\star{}^\top$ is a solution to the original SDP (P).

This strategy for solving the primal SDP can be implemented with a storage-optimal algorithm. Indeed, the variable y in the dual SDP (D) has length m, so there is no obstacle to solving the dual with storage $\Theta(m+n)$ using the subgradient type method described in section 6. We can compute the subspace $V_\star$ using the randomized range finder [34, Algorithm 4.1] with storage cost $\Theta(nr_\star)$. Last, we can solve the compressed primal SDP (1.5) using working storage $\Theta(m+n+r_\star^2)$ via the matrix-free method from [24, 53]. The total storage is the optimal $\Theta(m+nr_\star)$. Furthermore, all of these algorithms can be implemented with the data access oracles (1.4).

Hence—assuming exact solutions to the optimization problems—we have developed a storage-optimal approach to the SDP (P), summarized in Table 1, left.

1.4. The approximate complementarity principle. A major challenge remains: one very rarely has access to an exact dual solution! Rather, we usually have an approximate dual solution, obtained via some iterative dual solver.

This observation motivates us to formulate a new *approximate complementarity principle*. For now, assume that $r_\star$ is known. Given an approximate dual solution y, we can construct an orthonormal matrix $V \in \mathbf{R}^{n\times r_\star}$ whose columns are eigenvectors of $C - \mathcal{A}^\top y$ with the $r_\star$ smallest eigenvalues. Roughly speaking, the primal problem (P) admits an *approximate* solution X whose range is contained in $\mathbf{range}(V)$. We show the approximate solution is close to the true solution as measured in terms of suboptimality, infeasibility, and distance to the solution set.

We propose to recover the approximate primal solution by solving the semidefinite least-squares problem

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\left\|\mathcal{A}(VSV^\top) - b\right\|^2 \\ \text{subject to} & S \succeq 0 \end{array} \tag{MinFeasSDP}$$

Table 1
Exact and practical primal recovery.

Step	Exact primal recovery	Practical primal recovery
1	Compute dual solution $y_\star$	Compute approximate dual solution y
2	Compute basis $V_\star$ for $\mathbf{null}(C - \mathcal{A}^\top y_\star)$	Compute $r_\star$ eigenvectors of $C - \mathcal{A}^\top y$ with smallest eigenvalues; collect as columns of matrix V
3	Solve the compressed SDP (1.5)	Solve (MinFeasSDP)

with variable $S \in \mathbf{S}_{+}^{r_\star}$. Given a solution $\hat{S}$ to (MinFeasSDP), we obtain an (infeasible) approximate solution $X_{\mathrm{infeas}} = V\hat{S}V^\top$ to the primal problem.

In fact, it is essential to relax our attention to infeasible solutions because the feasible set of (P) should almost never contain a matrix with range V! This observation was very surprising to us, but it seems evident in retrospect (for example, using a dimension-counting argument together with Lemma A.1).

The resulting framework appears in Table 1, right. This approach for solving (P) leads to storage-optimal algorithms for the same reasons described in subsection 1.3. Our first main result ensures that this technique results in a provably good solution to the primal SDP (P).

THEOREM 1.2 (main theorem, informal). *Instate the regularity assumptions of subsection* 1.1. *Suppose the dual vector y has suboptimality $\epsilon := d_\star - b^\top y \le$ const. Consider the primal reconstruction X_{infeas} obtained by solving* (MinFeasSDP). *Then we may bound the distance between X_{infeas} to the primal solution $X_\star$ by*

$$\|X_{\mathrm{infeas}} - X_\star\|_F = \mathcal{O}(\sqrt{\epsilon}).$$

The constant in the $\mathcal{O}$ depends on the problem data $\mathcal{A}$, b, and C.

We state and prove the formal result as Theorem 4.1. As stated, this guarantee requires knowledge of the rank $r^\star$ of the solution; in section 5, we obtain a similar guarantee using an estimate for $r^\star$.

1.5. Paper organization. We discuss related work in section 2 with a focus on storage. Section 3 contains an overview of our notation and more detailed problem assumptions. Section 4 uses the approximate complementarity principle to develop practical, robust, and theoretically justified compressed SDPs such as (MinFeasSDP) for solving (P). These compressed SDPs are accompanied by detailed bounds on the quality of the computed solutions as compared with the true solution. Section 5 contains practical suggestions in solving these compressed SDPs such as choosing parameters and checking the solution quality numerically. Next, we turn to algorithms for solving the dual SDP: we explain how to compute an approximate dual solution efficiently in section 6, which provides the last ingredient for a complete method to solve (P). Section 7 shows numerically that the method is effective in practice. We conclude the paper with a discussion on contributions and future research directions in section 8.

2. Related work.

Semidefinite programming can be traced to a 1963 paper of Bellman and Fan [10]. Related questions emerged earlier in control theory, starting from Lyapunov's 1890 work on stability of dynamical systems. There are many classic applications in matrix analysis, dating to the 1930s. Graph theory provides another rich source of examples, beginning in the 1970s. See [13, 67, 65, 15] for more history and problem formulations.

2.1. Interior-point methods. The first reliable algorithms for semidefinite programming were interior-point methods (IPMs). These techniques were introduced independently by Nesterov and Nemirovski [51, 52] and Alizadeh [3, 4].

The success of these SDP algorithms motivated new applications. In particular, Goemans and Williamson [33] used semidefinite programming to design an approximation algorithm to compute the maximum-weight cut in a graph. Early SDP solvers could only handle graphs with a few hundred vertices [33, section 5], although computational advances quickly led to IPMs that could solve problems with thousands of vertices [11].

IPMs form a series of unconstrained problems whose solutions are feasible for the original SDP, and move toward the solutions of these unconstrained problems using Newton's method. As a result, IPMs converge to high accuracy in very few iterations but require substantial work per iteration. To solve a standard-form SDP with an $n \times n$ matrix variable and with m equality constraints (without further structural assumptions), a typical IPM requires $\mathcal{O}(\sqrt{n}\log(\frac{1}{\epsilon}))$ iterations to reach a solution with accuracy ϵ (in terms of objective value) [50] and $\mathcal{O}(mn^3 + m^2n^2 + m^3)$ arithmetic operations per iteration [6], so $\mathcal{O}(\sqrt{n}\log(\frac{1}{\epsilon})(mn^3+m^2n^2+m^3))$ arithmetic operations in total. Further, a typical IPM requires at least $\Theta(n^2 + m + m^2)$ working memory, and storing the data takes $\Theta(n^2m)$ memory [6].

As a consequence, these algorithms are not effective for solving large problem instances, unless they enjoy a lot of structure. Hence researchers began to search for methods that could scale to larger problems.

2.2. First-order methods. One counterreaction to the expense of IPMs was to develop first-order optimization algorithms for SDPs. This line of work began in the late 1990s, and it accelerated as SDPs emerged in the machine learning and signal processing literature in the 2000s.

Early on, Helmberg and Rendl [36] proposed a spectral bundle method for solving an SDP in dual form, and they showed that it converges to a dual solution when the trace of $X_\star$ is constant. In contrast to IPMs, the spectral bundle method has low per-iteration complexity. On the other hand, the convergence rate is not known, and there is no convergence guarantee on the primal side, so there is no explicit control on the storage and arithmetic costs.

Popular first-order algorithms include the proximal gradient method [59], accelerated variants [9], and the alternating direction method of multipliers [30, 31, 14, 54]. These methods provably solve the original convex formulation of (P). But they all store the full primal matrix variable, so they are not storage-efficient.

Recently, Friedlander and Macedo [29] have proposed a novel first-order method that is based on gauge duality, rather than Lagrangian duality. This approach converts an SDP into an eigenvalue optimization problem. The authors propose a mechanism for using a dual solution to construct a primal solution. This paper is similar in spirit to our approach, but it lacks an analysis of the accuracy of the primal solution. Moreover, it only applies to problems with a positive-definite objective, i.e., $C \succ 0$.

2.3. Storage-efficient first-order methods. Motivated by problems in signal processing and machine learning, a number of authors have revived the conditional gradient method (CGM) [28, 44]. In particular, Hazan [35] suggested using CGM for semidefinite programming. Clarkson [23] developed a new analysis, and Jaggi [38], showed how this algorithm applies to a wide range of interesting problems.

The appeal of the CGM is that it computes an approximate solution to an SDP as a sum of rank-one updates; each rank-one update is obtained from an approximate eigenvector computation. In particular, after t iterations, the iterate has rank at most t. This property has led to the exaggeration that CGM is a "storage-efficient" optimization method when terminated early enough. Unfortunately, CGM converges very slowly, so the iterates do not have controlled rank. The literature describes many heuristics for attempting to control the rank of the iterates [56, 74], but these methods all lack guarantees.

Some of the authors of this paper [76] have shown how to use CGM to design a storage-optimal algorithm for a class of SDPs by sketching the decision variable. Concurrently with the present work, sketching methods have been extended to yield storage-optimal solvers for standard-form SDP by using algorithms that generalize CGM [73, 72, 75].

We also mention a subgradient method developed by Renegar [58] that can be used to solve either the primal or the dual SDP. Renegar's method has a computational profile similar to CGM, and it does not have controlled storage costs.

2.4. Factorization methods. There is also a large class of heuristic SDP algorithms based on matrix factorization. The key idea is to factorize the matrix variable $X = FF^\top, F \in \mathbf{R}^{n\times r}$ and to reformulate the SDP (P) as

$$\begin{array}{ll} \text{minimize} & \mathbf{tr}(CFF^\top) \\ \text{subject to} & \mathcal{A}(FF^\top) = b. \end{array} \tag{2.1}$$

We can apply a wide range of nonlinear programming methods to optimize (2.1) with respect to the variable F. In contrast to the convex methods described above, these techniques only offer incomplete guarantees on storage, arithmetic, and convergence.

The factorization idea originates in the paper [37] of Homer and Peinado. They focused on the MaxCut SDP, and the factor F was a *square* matrix, i.e., $r = n$. These choices result in an unconstrained nonconvex optimization problem that can be tackled with a first-order optimization algorithm.

Theoretical work of Barvinok [8] and Pataki [55] demonstrates that the primal SDP (P) always admits a solution with rank r, with $\binom{r+1}{2} \le m$. (Note, however, that the SDP can have solutions with much lower or higher rank.)

Inspired by the existence of low-rank solutions to SDP, Burer and Monteiro [16] proposed to solve the optimization problem (2.1) where the variable $F \in \mathbf{R}^{n\times p}$ is constrained to be a *tall* matrix ($p \ll n$). The number p is called the factorization rank. It is clear that every rank-r solution to the SDP (P) induces a solution to the factorized problem (2.1) when $p \ge r$. Burer and Monteiro applied a limited-memory BFGS algorithm to solve (2.1) in an explicit effort to reduce storage costs.

In subsequent work, Burer and Monteiro [17] proved that, under technical conditions, the local minima of the nonconvex formulation (2.1) are global minima of the SDP (P), provided that the factorization rank p satisfies $\binom{p+1}{2} \ge m + 1$. As a consequence, algorithms based on (2.1) often set the factorization rank $p \approx \sqrt{2m}$, so the storage costs are $\Omega(n\sqrt{m})$.

Unfortunately, a recent result of Waldspurger and Walters [69, Theorem 2 and Remark 2] demonstrates that the formulation (2.1) cannot lead to storage-optimal algorithms for interesting SDPs which are verified to be regular in [25]. In particular, suppose that the feasible set of (P) satisfies a mild technical condition and contains a matrix with rank *one*. Whenever the factorization rank satisfies $\binom{p+1}{2} + p \le m$, there is a set of cost matrices C with positive Lebesgue measure for which the factorized problem (2.1) has (1) a unique global optimizer with rank one and (2) at least one suboptimal local minimizer, while the original SDP has a unique primal and dual solution that satisfy strict complementarity. In this situation, the variable in the factorized SDP actually requires $\Omega(n\sqrt{m})$ storage, which is not optimal if $m = \omega(1)$. In view of this negative result, we omit a detailed review of the literature on the analysis of factorization methods. See [69] for a full discussion.

3. Basics and notation.

Here we introduce some additional notation and some metrics for evaluating the quality of a solution and the conditioning of an SDP.

3.1. Notation. We will work with the Frobenius norm $\|\cdot\|_{\mathrm{F}}$, the ℓ_2 operator norm $\|\cdot\|_{\mathrm{op}}$, and its dual, the ℓ_2 nuclear norm $\|\cdot\|_*$. We reserve the symbols $\|\cdot\|$ and

$\|\cdot\|_2$ for the norm induced by the canonical inner product of the underlying real vector space.[1]

For a matrix $B \in \mathbf{R}^{d_1 \times d_2}$, we arrange its singular values in decreasing order:

$$\sigma_1(B) \geq \cdots \geq \sigma_{\min(d_1,d_2)}(B).$$

Define $\sigma_{\min}(B) = \sigma_{\min(d_1,d_2)}(B)$ and $\sigma_{\max}(B) = \sigma_1(B)$, and let $\sigma_{\min>0}(B)$ denote the smallest *nonzero* singular value of B. For a linear operator $\mathcal{B} : \mathbf{S}^{d_1} \to \mathbf{R}^{d_2}$, we define

$$\sigma_{\min}(\mathcal{B}) = \min_{\|A\|=1} \|\mathcal{B}(A)\| \quad \text{and} \quad \|\mathcal{B}\|_{\text{op}} = \max_{\|A\|=1} \|\mathcal{B}(A)\|.$$

We use analogous notation for the eigenvalues of a symmetric matrix. In particular, the map $\lambda_i(\cdot) : \mathbf{S}^n \to \mathbf{R}$ reports the ith largest eigenvalue of its argument.

3.2. Optimal solutions. Instate the notation and regularity assumptions from subsection 1.1. Define the slack operator $Z : \mathbf{R}^n \to \mathbf{S}^n$ that maps a putative dual solution $y \in \mathbf{R}^m$ to its associated slack matrix $Z(y) := C - \mathcal{A}^\top y$. We omit the dependence on y if it is clear from the context.

Let the rank of primal solution being $r_\star$ and denote its range as $\mathcal{V}_\star$. We also fix an orthonormal matrix $V_\star \in \mathbf{R}^{n \times r_\star}$ whose columns span $\mathcal{V}_\star$. Introduce the subspace $\mathcal{U}_\star = \mathbf{range}(Z(y_\star))$, and let $U_\star \in \mathbf{R}^{n \times (n-r_\star)}$ be a fixed orthonormal basis for $\mathcal{U}_\star$. We have the decomposition $\mathcal{V}_\star + \mathcal{U}_\star = \mathbf{R}^{n \times n}$.

For a matrix $V \in \mathbf{R}^{n \times r}$, define the compressed cost matrix and constraint map

$$C_V := V^\top C V \quad \text{and} \quad \mathcal{A}_V(S) := \mathcal{A}(VSV^\top) \quad \text{for } S \in \mathbf{S}^r. \tag{3.1}$$

In particular, $\mathcal{A}_{V_\star}$ is the compression of the constraint map onto the range of $X_\star$.

3.3. Conditioning of the SDP. Our analysis depends on conditioning properties of the pair of primal (P) and dual (D) SDPs.

First, we measure the strength of the complementarity condition (1.2) using the spectral gaps of the primal solution $X_\star$ and dual slack matrix $Z(y_\star)$:

$$\lambda_{\min>0}(X_\star) \quad \text{and} \quad \lambda_{\min>0}(Z(y_\star)).$$

These two numbers capture how far we can perturb the solutions before the complementarity condition fails.

Second, we measure the robustness of the primal solution to perturbations of the problem data b using the quantity

$$\kappa := \frac{\sigma_{\max}(\mathcal{A})}{\sigma_{\min}(\mathcal{A}_{V^\star})}. \tag{3.2}$$

This term arises because we have to understand the conditioning of the system $\mathcal{A}_{V_\star}(S) = b$ of linear equations in the variable $S \in \mathbf{S}^{r_\star}$.

3.4. Quality of solutions. We measure the quality of a primal matrix variable $X \in \mathbf{S}^n_+$ and a dual vector $y \in \mathbf{R}_m$ in terms of their suboptimality, their infeasibility, and their distance to the true solutions. Table 2 gives formulas for these quantities.

[1]For symmetric matrices, we regard the trace inner product as the canonical one. For the Cartesian product $\mathbf{S}^n \times \mathbf{R}^m$, we regard the sum of trace inner product on $\mathbf{S}^n$ and the dot product on $\mathbf{R}^m$ as the canonical one.

TABLE 2
Quality of a primal matrix $X \in \mathbf{S}^n_+$ and a dual vector $y \in \mathbf{R}^m$.

	Primal matrix X	Dual vector y
Suboptimality (ϵ)	$\mathbf{tr}(CX) - p_\star$	$d_\star - b^\top y$
Infeasibility (δ)	$\max\{\|\mathcal{A}X - b\|, (-\lambda_{\min}(X))_+\}$	$(-\lambda_{\min}(Z(y)))_+$
Distance to solution (d)	$\|X - X^\star\|_{\mathrm{F}}$	$\|y - y_\star\|_2$

We say that a matrix X is an (ϵ, δ)-solution of (P) if its suboptimality $\epsilon_p(X)$ is at most ϵ and its infeasibility $\delta_p(X)$ is at most δ.

The primal suboptimality $\epsilon_p(X)$ and infeasibility $\delta_p(X)$ are both controlled by the distance $d_p(X)$ to the primal solution:

$$\epsilon_p(X) \le \|C\|_{\mathrm{F}}\, d_p(X) \quad \text{and} \quad \delta_p(X) \le \max\{1, \|\mathcal{A}\|_{\mathrm{op}}\} d_p(X). \tag{3.3}$$

We can also control the distance of a dual vector y and its slack matrix $Z(y)$ from their optima using the following quadratic growth lemma.

LEMMA 3.1 (quadratic growth). *Instate the regularity assumptions from subsection* 1.1. *For any dual feasible y with dual slack matrix $Z(y) := C - \mathcal{A}^\top y$ and dual suboptimality $\epsilon = \epsilon_d(y) = d_\star - b^\top y$, we have*

(3.4)
$$\|(Z(y), y) - (Z(y_\star), y_\star)\| \le \frac{1}{\sigma_{\min}(\mathcal{D})}\left[\frac{\epsilon}{\lambda_{\min>0}(X_\star)} + \sqrt{\frac{2\epsilon}{\lambda_{\min>0}(X_\star)}\|Z(y)\|_{\mathrm{op}}}\right],$$

where the linear operator $\mathcal{D} : \mathbf{S}^n \times \mathbf{R}^m \to \mathbf{S}^n \times \mathbf{S}^n$ is defined by

$$\mathcal{D}(Z, y) := (Z - (U_\star {U_\star}^\top) Z (U_\star {U_\star}^\top), Z + \mathcal{A}^\top y).$$

The orthonormal matrix $U_\star$ is defined in subsection 3.2. *The quantity $\sigma_{\min}(\mathcal{D})$ is defined as $\sigma_{\min}(\mathcal{D}) := \min_{\|(Z,y)\|=1} \|(Z - (U_\star {U_\star}^\top) Z (U_\star {U_\star}^\top), Z + \mathcal{A}^\top y)\|$.*

The proof of Lemma 3.1 can be found in Appendix A. The name *quadratic growth* arises from a limit of inequality (3.4): when ϵ is small, the second term in the bracket dominates the first term, so $\|y - y_\star\|_2^2 = \mathcal{O}(\epsilon)$ [27].

4. Reduced SDPs and approximate complementarity.

In this section, we describe two reduced SDP formulations, and we explain when their solutions are nearly optimal for the original SDP (P). We can interpret these results as constructive proofs of the approximate complementarity principle.

4.1. Reduced SDPs.

Suppose that we have obtained a dual approximate solution y and its associated dual slack matrix $Z(y) := C - \mathcal{A}^\top y$. Let r be a rank parameter, which we will discuss later. Construct an orthonormal matrix $V \in \mathbf{R}^{n\times r}$ whose range is an r-dimensional invariant subspace associated with the r smallest eigenvalues of the dual slack matrix $Z(y)$. Our goal is to compute a matrix X with range V that approximately solves the primal SDP (P).

Our first approach minimizes infeasibility over all psd matrices with range V:

$$\text{(MinFeasSDP)} \qquad \begin{array}{ll} \text{minimize} & \frac{1}{2}\|\mathcal{A}_V(S) - b\|^2 \\ \text{subject to} & S \succeq 0 \end{array}$$

with variable $S \in \mathbf{S}^r$. Given a solution $\hat{S}$, we can form an approximate solution $X_{\text{infeas}} = V\hat{S}V^\top$ for the primal SDP (P). This is the same method from subsection 1.4.

TABLE 3
Comparison of (MinFeasSDP) *and* (MinObjSDP) *given a feasible ϵ-suboptimal dual vector y.*

Assumption and quality	(MinFeasSDP)	(MinObjSDP)
Require $r = r_\star$?	Yes	No
Suboptimality	$\mathcal{O}(\kappa\sqrt{\epsilon})$	$\mathcal{O}(\sqrt{\epsilon})$
Infeasibility	$\mathcal{O}(\kappa\sqrt{\epsilon})$	$\mathcal{O}(\sqrt{\epsilon})$
Distance to the solution	$\mathcal{O}(\kappa\sqrt{\epsilon})$	Remark 4.8

Our second approach minimizes the objective value over all psd matrices with range V, subject to a specified limit δ on infeasibility:

$$\text{(MinObjSDP)} \qquad \begin{array}{ll} \text{minimize} & \mathbf{tr}(C_V S) \\ \text{subject to} & \|\mathcal{A}_V(S) - b\| \le \delta \quad \text{and} \quad S \succeq 0 \end{array}$$

with variable $S \in \mathbf{S}^r$. Given a solution $\tilde{S}$, we can form an approximate solution $X_{\text{obj}} = V\tilde{S}V^\top$ for the primal SDP (P).

As we will see, both approaches lead to satisfactory solutions to the original SDP (P) under appropriate assumptions. Theorem 4.1 addresses the performance of (MinFeasSDP), while Theorem 4.6 addresses the performance of (MinObjSDP). Table 3 summarizes the hypotheses we impose to study each of the two problems, as well as the outcomes of the analysis.

The bounds in this section depend on the problem data and rely on assumptions that are not easy to check. We discuss how to check the quality of X_{infeas} and X_{obj} in section 5.

4.2. Analysis of (MinFeasSDP). First, we establish a result that connects the solution of (MinFeasSDP) with the solution of the original problem (P).

THEOREM 4.1 (analysis of (MinFeasSDP)). *Instate the regularity assumptions in subsection* 1.1. *Moreover, assume the solution rank $r_\star$ is known. Set $r = r_\star$. Let $y \in \mathbf{R}^m$ be feasible for the dual SDP* (D) *with suboptimality $\epsilon = \epsilon_d(y) = d_\star - b^\top y < c_1$, where the constant $c_1 > 0$ depends only on $\mathcal{A}, b$, and C. Then the threshold $T := \lambda_{n-r}(Z(y))$ obeys*

$$T := \lambda_{n-r}(Z(y)) \ge \frac{1}{2}\lambda_{n-r}(Z(y_\star)) > 0,$$

and we have the bound

$$\|X_{\text{infeas}} - X_\star\|_F \le (1 + 2\kappa)\left(\frac{\epsilon}{T} + \sqrt{2\frac{\epsilon}{T}\|X_\star\|_{\text{op}}}\right). \tag{4.1}$$

This bound shows that $\|X_{\text{infeas}} - X_\star\|_F^2 = \mathcal{O}(\epsilon)$ when the dual vector y is ϵ suboptimal. Notice this result requires knowledge of the solution rank $r_\star$. The proof of Theorem 4.1 occupies the rest of this section.

4.2.1. Primal optimizers and the reduced search space. The first step in the argument is to prove that $X_\star$ is near the search space $\{VSV^\top : S \in \mathbf{S}_+^r\}$ of the reduced problems.

LEMMA 4.2. *Instate the regularity assumptions in subsection* 1.1. *Further suppose $y \in \mathbf{R}^m$ is feasible and ϵ-suboptimal for the dual SDP* (D), *and construct the orthonormal matrix V as in subsection* 4.1. *Assume that the threshold $T :=$*

$\lambda_{n-r}(C-\mathcal{A}^\top y) > 0$. *Define* $P_V(X) = VV^\top XVV^\top$, *and* $P_{V^\perp}(X) = X - P_V(X)$ *for any* $X \in \mathbf{S}^n$. *Then for any solution* $X_\star$ *of the primal SDP* (P),

$$\|P_{V^\perp}(X_\star)\|_F \le \frac{\epsilon}{T} + \sqrt{2\frac{\epsilon}{T}\|X_\star\|_{\mathrm{op}}} \text{ and } \|P_{V^\perp}(X_\star)\|_* \le \frac{\epsilon}{T} + 2\sqrt{r\frac{\epsilon}{T}\|X_\star\|_{\mathrm{op}}}.$$

To prove the lemma, we will utilize the following result (proved in Appendix B) which bounds the distance to subspaces via the inner product. This result might be of independent interest.

LEMMA 4.3. *Suppose* $X, Z \in \mathbf{S}^n$ *are both positive semidefinite. Let* $V \in \mathbf{R}^{n\times r}$ *be the matrices formed by the eigenvectors with the smallest* r *eigenvalues of* Z. *Let* $\epsilon = \mathbf{tr}(XZ)$ *and* $P_V(X) = VV^\top XVV^\top$. *If* $T = \lambda_{n-r}(Z) > 0$, *then*

$$\|X - P_V(X)\|_F \le \frac{\epsilon}{T} + \sqrt{2\frac{\epsilon}{T}\|X\|_{\mathrm{op}}}, \text{ and } \|X - P_V(X)\|_* \le \frac{\epsilon}{T} + 2\sqrt{r\frac{\epsilon}{T}\|X\|_{\mathrm{op}}}.$$

Now we are ready to prove Lemma 4.2.

Proof of Lemma 4.2. We shall utilize Lemma 4.3. Simply set Z in Lemma 4.3 to be $C-\mathcal{A}^\top y$ from the approximate dual solution y and X to be the primal solution $X_\star$. Using strong duality in the following step (a) and feasibility of $X_\star$ in the following step (b), we have

$$\epsilon = b^\top y_\star - b^\top y \overset{(a)}{=} \mathbf{tr}(CX_\star) - b^\top y \overset{(b)}{=} \mathbf{tr}(CX_\star) - (\mathcal{A}X_\star)^\top y = \mathbf{tr}(X_\star Z).$$

Hence, we can apply Lemma 4.3 to obtain the bounds in Lemma 4.2. ◻

4.2.2. Relationship between the solutions of (MinFeasSDP) and (P). Lemma 4.2 shows that any solution $X_\star$ of (P) is close to its compression $VV^\top X_\star VV^\top$ onto the range of V. Next, we show that X_{infeas} is also close to $VV^\top X_\star VV^\top$. We can invoke strong convexity of the objective of (MinFeasSDP) to achieve this goal.

LEMMA 4.4. *Instate the assumptions and notation from Lemma* 4.2. *Assume* $\sigma_{\min}(\mathcal{A}_V) > 0$ *and that the threshold* $T = \lambda_{n-r}(Z(y)) > 0$. *Then*

$$\|X_{\text{infeas}} - X_\star\|_F \le \left(1 + \frac{\sigma_{\max}(\mathcal{A})}{\sigma_{\min}(\mathcal{A}_V)}\right)\left(\frac{\epsilon}{T} + \sqrt{2\frac{\epsilon}{T}\|X_\star\|_{\mathrm{op}}}\right), \tag{4.2}$$

where $X_\star$ *is any solution of the primal SDP* (P).

Proof. Since we assume that $\sigma_{\min}(\mathcal{A}_V) > 0$, we know the objective of (Min FeasSDP), $f(S) = \frac{1}{2}\|\mathcal{A}_V(S) - b\|_2^2$, is $\sigma_{\min}^2(\mathcal{A}_V)$-strongly convex, and so the solution $S_\star$ is unique. We then have for any $S \in \mathbf{S}^r$

$$\begin{aligned} f(S) - f(S^\star) &\overset{(a)}{\ge} \mathbf{tr}(\nabla f(S^\star)^\top (S - S^\star)) + \frac{\sigma_{\min}^2(\mathcal{A}_V)}{2}\|S - S^\star\|_F^2 \\ &\overset{(b)}{\ge} \frac{\sigma_{\min}^2(\mathcal{A}_V)}{2}\|S - S^\star\|_F^2, \end{aligned} \tag{4.3}$$

where step (a) uses strong convexity and step (b) is due to the optimality of $S_\star$.

Since $\mathcal{A}X_\star = b$, we can bound the objective of (MinFeasSDP) by $\|P_V(X) - X_\star\|_F$:

$$\left\|\mathcal{A}_V(V^\top XV) - b\right\|_2 = \|\mathcal{A}(P_V(X) - X_\star)\|_2 \quad \le \sigma_{\max}(\mathcal{A})\|P_V(X) - X_\star\|_F. \tag{4.4}$$

Combining pieces, we know that $S_\star$ satisfies

$$\begin{aligned}\left\|S^\star - V^\top X_\star V\right\|_{\mathrm{F}}^2 &\overset{(a)}{\le} \frac{2}{\sigma_{\min}^2(\mathcal{A}_V)}(f(V^\top X_\star V) - f(S^\star)) \overset{(b)}{\le} \frac{\sigma_{\max}^2(\mathcal{A})}{\sigma_{\min}^2(\mathcal{A}_V)}\left\|X_\star - P_V(X_\star)\right\|_{\mathrm{F}}^2 \\ &\overset{(c)}{\le} \frac{\sigma_{\max}^2(\mathcal{A})}{\sigma_{\min}^2(\mathcal{A}_V)}\left(\frac{\epsilon}{T} + \sqrt{2\frac{\epsilon}{T}\left\|X_\star\right\|_{\mathrm{op}}}\right)^2,\end{aligned}$$

where step (a) uses (4.3), step (b) uses (4.4) for $X = X_\star$ and $f(S^\star) \ge 0$, and step (c) uses Lemma 4.2. Lifting to the larger space $\mathbf{R}^{n\times n}$, we see

$$\begin{aligned}\left\|VS^\star V^\top - X_\star\right\|_{\mathrm{F}} &\le \left\|VS^\star V^\top - P_V(X_\star)\right\|_{\mathrm{F}} + \left\|X_\star - P_V(X)\right\|_{\mathrm{F}} \\ &\overset{(a)}{=} \left\|S^\star - V^\top X_\star V\right\|_{\mathrm{F}} + \left\|X_\star - P_V(X_\star)\right\|_{\mathrm{F}} \\ &\overset{(b)}{\le} \left(1 + \frac{\sigma_{\max}(\mathcal{A})}{\sigma_{\min}(\mathcal{A}_V)}\right)\left(\frac{\epsilon}{T} + \sqrt{2\frac{\epsilon}{T}\left\|X_\star\right\|_{\mathrm{op}}}\right).\end{aligned}$$

Here we use the unitary invariance of $\|\cdot\|_{\mathrm{F}}$ in (a). The inequality (b) is due to our bound above for $S_\star$ and Lemma 4.2. ∎

4.2.3. Lower bounds for the threshold and minimum singular value. Finally, we must confirm that the extra hypotheses of Lemma 4.4 hold, i.e., $T > 0$ and $\sigma_{\min}(\mathcal{A}_V) > 0$.

We explain the intuition here. Strict complementarity forces $\lambda_{n-r}(Z(y_\star)) > 0$. If Z is close to $Z(y_\star)$, then we expect that $T > 0$ by continuity. When $X_\star$ is unique, Lemma A.1 implies that $\mathbf{null}(\mathcal{A}_{V_\star}) = \{0\}$. As a consequence, $\sigma_{\min}(\mathcal{A}_{V_\star}) > 0$. If V is close to $V_\star$, then we expect that $\sigma_{\min}(\mathcal{A}_V) > 0$ as well. We have the following rigorous statement.

LEMMA 4.5. *Instate the hypotheses of Theorem* 4.1. *Then*

$$\begin{aligned}T = \lambda_{n-r}(Z(y)) &\ge \frac{1}{2}\lambda_{n-r}(Z(y_\star)); \\ \sigma_{\min}(\mathcal{A}_V) &\ge \frac{1}{2}\sigma_{\min}(\mathcal{A}_{V^\star}) > 0.\end{aligned}$$

Proof. We first prove the lower bound on the threshold T. Using $\|(Z, y) - (Z(y_\star), y_\star)\| \ge \|Z - Z(y_\star)\|_{\mathrm{op}} \ge \|Z\|_{\mathrm{op}} - \|Z(y_\star)\|_{\mathrm{op}}$ and quadratic growth (Lemma 3.1), we have

$$\|Z\|_{\mathrm{op}} - \|Z(y_\star)\|_{\mathrm{op}} \le \frac{1}{\sigma_{\min}(\mathcal{D})}\left(\frac{\epsilon}{\lambda_{\min>0}(X_\star)} + \sqrt{\frac{2\epsilon}{\lambda_{\min>0}(X_\star)}\|Z\|_{\mathrm{op}}}\right).$$

Thus for sufficiently small ϵ, we have $\|Z\|_{\mathrm{op}} \le 2\|Z(y_\star)\|_{\mathrm{op}}$. Substituting this bound into the previous inequality gives

$$\|(Z, y) - (Z(y_\star), y_\star)\| \le \frac{1}{\sigma_{\min}(\mathcal{D})}\left(\frac{\epsilon}{\lambda_{\min>0}(X_\star)} + \sqrt{\frac{4\epsilon}{\lambda_{\min>0}(X_\star)}\|Z(y_\star)\|_{\mathrm{op}}}\right). \tag{4.5}$$

Weyl's inequality tells us that $\lambda_{n-r}(Z(y_\star)) - T \le \|Z - Z(y_\star)\|_{\mathrm{op}}$. Using (4.5), we see that for all sufficiently small ϵ, $T := \lambda_{n-r}(C - \mathcal{A}^\top y) \ge \frac{1}{2}\lambda_{n-r}(Z(y_\star))$.

Next we prove the lower bound on $\mathcal{A}_V$. We have $\sigma_{\min}(\mathcal{A}_{V^\star}) > 0$ by Lemma A.1. It will be convenient to align the columns of V with those of $V^\star$ for our analysis. Consider the solution $O_\star$ to the orthogonal Procrustes problem $O_\star = \operatorname{argmin}_{OO^\top = I, O\in\mathbf{R}^{r\times r}} \|VO - V^\star\|$. Since $\sigma_{\min}(\mathcal{A}_V) = \sigma_{\min}(\mathcal{A}_{VO_\star})$ for orthonormal $O_\star$, without loss of generality, we suppose we have already performed the alignment and V is $VO_\star$ in the following.

Let $S_1 = \operatorname{argmin}_{\|S\|_{\mathrm{F}}=1} \|\mathcal{A}_V(S)\|_2$. Then we have

$$\begin{aligned}\sigma_{\min}(\mathcal{A}_{V_\star}) - \sigma_{\min}(\mathcal{A}_V) &\le \|\mathcal{A}_{V_\star}(S_1)\|_2 - \|\mathcal{A}_V(S_1)\|_2\\ &\le \left\|\mathcal{A}(V^\star S_1 (V^\star)^\top) - \mathcal{A}(VS_1V^\top)\right\|_2\\ &\le \|\mathcal{A}\|_{\mathrm{op}} \left\|V^\star S_1 (V^\star)^\top - (VS_1V^\top)\right\|_{\mathrm{F}}.\end{aligned} \tag{4.6}$$

Defining $E = V - V^\star$, we bound the term $\left\|V^\star S_1 (V^\star)^\top - (VS_1V^\top)\right\|_{\mathrm{F}}$ as

$$\begin{aligned}\left\|V^\star S_1 (V^\star)^\top - (VS_1V^\top)\right\|_{\mathrm{F}} &= \left\|ES_1(V_\star)^\top + V_\star S_1 E^\top + ES_1E^\top\right\|_{\mathrm{F}}\\ &\overset{(a)}{\le} 2\|E\|_{\mathrm{F}}\|V_\star S_1\|_{\mathrm{F}} + \|E\|_{\mathrm{F}}^2 \|S_1\|_{\mathrm{F}}\\ &\overset{(b)}{=} 2\|E\|_{\mathrm{F}} + \|E\|_{\mathrm{F}}^2,\end{aligned} \tag{4.7}$$

where (a) uses the triangle inequality and the submultiplicativity of the Frobenius norm. We use the orthogonality of the columns of V and of $V^\star$ and the fact that $\|S_1\|_{\mathrm{F}} = 1$ in step (b).

A variant of the Davis–Kahan inequality [71, Theorem 2] asserts that $\|E\|_{\mathrm{F}} \le 4\|Z - Z(y_\star)\|_{\mathrm{F}} / \lambda_{\min>0}(Z(y_\star))$. Combining this fact with inequality (4.5), we see $\|E\|_{\mathrm{op}} \to 0$ as $\epsilon \to 0$. Now using (4.7) and (4.6), we see that for all sufficiently small ϵ, $\sigma_{\min}(\mathcal{A}_V) \ge \frac{1}{2}\sigma_{\min}(\mathcal{A}_{V^\star}) > 0$. ∎

4.2.4. Proof of Theorem 4.1. Instate the hypotheses of Theorem 4.1. Now, Lemma 4.5 implies that $\sigma_{\min}(\mathcal{A}_V) > 0$ and that $T > 0$. Therefore, we can invoke Lemma 4.4 to obtain the stated bound on $\|X_{\text{infeas}} - X_\star\|_{\mathrm{F}}$.

4.3. Analysis of (MinObjSDP).

Next, we establish a result that connects the solution to (MinObjSDP) with the solution to the original problem (P).

THEOREM 4.6 (analysis of (MinObjSDP)). *Instate the regularity assumptions in subsection* 1.1. *Moreover, assume $r \ge r_\star$. Let $y \in \mathbf{R}^m$ be feasible for the dual SDP* (D) *with suboptimality $\epsilon = \epsilon_d(y) = d_\star - b^\top y < c_2$, where the constant $c_2 > 0$ depends only on $\mathcal{A}, b$, and C. Then the threshold $T := \lambda_{n-r}(Z(y))$ obeys*

$$T := \lambda_{n-r}(Z(y)) \ge \frac{1}{2}\lambda_{n-r}(Z(y_\star)) > 0.$$

Introduce the quantities

$$\begin{aligned}\delta_0 &:= \sigma_{\max}(\mathcal{A})\left(\frac{\epsilon}{T} + \sqrt{2\frac{\epsilon}{T}\|X_\star\|_{\mathrm{op}}}\right);\\ \epsilon_0 &:= \min\left\{\|C\|_F\left(\frac{\epsilon}{T} + \sqrt{2\frac{\epsilon}{T}\|X_\star\|_{\mathrm{op}}}\right), \|C\|_{\mathrm{op}}\left(\frac{\epsilon}{T} + \sqrt{2\frac{r\epsilon}{T}\|X_\star\|_{\mathrm{op}}}\right)\right\}.\end{aligned}$$

If we solve (MinObjSDP) *with the infeasibility parameter $\delta = \delta_0$, then the resulting matrix X_{obj} is an (ϵ_0, δ_0)-solution to* (P).

If in addition $C = I$, then X_{obj} is superoptimal with $0 \ge \epsilon_0 \ge -\frac{\epsilon}{T}$.

The analysis in Theorem 4.1 of (MinFeasSDP) requires knowledge of the solution rank $r_\star$, and the bounds depend on the conditioning κ. In contrast, Theorem 4.6 does not require knowledge of $r_\star$, and the bounds do not depend on κ. Table 3 compares our findings for the two optimization problems Theorems 4.1 and 4.6.

Remark 4.7. The quality of the primal reconstruction depends on the ratio between the threshold T and the suboptimality ϵ. The quality improves as the suboptimality ϵ decreases, so the primal reconstruction approaches optimality as the dual estimate y approaches optimality. The threshold T is increasing in the rank estimate r, and so the primal reconstruction improves as r increases. Since r controls the storage required for the primal reconstruction, we see that the quality of the primal reconstruction improves as our storage budget increases.

Remark 4.8. Using the concluding remarks of [63], the above bound on suboptimality and infeasibility shows that the distance between X_{obj} and $X_\star$ is at most $\mathcal{O}(\epsilon^{1/4})$. Here, the $\mathcal{O}(\cdot)$ notation omits constants depending on $\mathcal{A}$, b, and C.

The proof of Theorem 4.6 occupies the rest of this subsection.

4.3.1. Bound on the threshold via quadratic growth. We first bound T when the suboptimality of y is bounded. This bound is a simple consequence of quadratic growth (Lemma 3.1).

Lemma 4.9. *Instate the hypotheses of Theorem* 4.6. *Then*

$$T := \lambda_{n-r}(Z(y)) \geq \frac{1}{2}\lambda_{n-r}(Z(y_\star)) > 0.$$

Proof. The proof follows exactly the same line (without even changing the notation) as the proof of Lemma 4.5 in assuring $\lambda_{n-r_\star}(Z(y)) > 0$, by noting $\lambda_{n-r}(Z(y_\star)) > 0$ whenever $r \geq n - \mathbf{rank}(Z(y_\star))$. ∎

4.3.2. Proof of Theorem 4.6. Lemma 4.2 shows that any primal solution $X_\star$ is close to $VV^\top X_\star VV^\top =: P_V(X_\star)$. We must ensure that $P_V(X_\star)$ is feasible for (MinObjSDP). This is achieved by setting the infeasibility parameter in (MinObjSDP) as

$$\delta := \sigma_{\max}(\mathcal{A})\left(\frac{\epsilon}{T} + \sqrt{2\frac{\epsilon\,\|X_\star\|_{\mathrm{op}}}{T}}\right).$$

This choice also guarantees all solutions to (MinObjSDP) are δ-feasible.

The solution to (MinObjSDP) is δ_0-feasible by construction. It remains to show the solution is ϵ_0-suboptimal. We can bound the suboptimality of the feasible point $P_V(X_\star)$ to produce a bound on the suboptimality of the solution to (MinObjSDP). We use Hölder's inequality to translate the bound on the distance between $P_V(X_\star)$ and $X_\star$, from Lemma 4.2, into a bound on the suboptimality:

$$\begin{aligned}&\mathbf{tr}(C(P_V(X_\star) - X_\star))\\ &\quad\leq \epsilon_0 := \min\left\{\|C\|_{\mathrm{F}}\left(\frac{\epsilon}{T} + \sqrt{2\frac{\epsilon}{T}\|X_\star\|_{\mathrm{op}}}\right), \|C\|_{\mathrm{op}}\left(\frac{\epsilon}{T} + \sqrt{2\frac{r\epsilon}{T}\|X_\star\|_{\mathrm{op}}}\right)\right\}.\end{aligned}$$

Thus $P_V(X_\star)$ is ϵ_0 suboptimal. Since $P_V(X_\star)$ is also feasible for (MinObjSDP), the solution to (MinObjSDP) is also ϵ_0 suboptimal.

To prove the improvement for the case $C = I$, we first complete V to form a basis $W = [U\,V]$ for $\mathbf{R}^n$, where $U = [v_{r+1}, \ldots, v_n] \in \mathbf{R}^{n\times(n-r)}$ and where v_i is the eigenvector of Z associated with the ith smallest eigenvalue. Define $X_1 = U^\top X_\star U$ and $X_2 = V^\top X_\star V$. We first note that

$$\mathbf{tr}(X_\star) = \mathbf{tr}(W^\top X_\star W) = \mathbf{tr}(X_1) + \mathbf{tr}(X_2) \quad \text{and} \quad \mathbf{tr}(X_2) = \mathbf{tr}(VV^\top X_\star VV^\top).$$

We can bound $\mathbf{tr}(X_1)$ using the following inequality:

$$\epsilon \overset{(a)}{=} \mathbf{tr}(ZX_\star) = \sum_{i=1}^{n} \lambda_{n-i+1}(Z) v_i^\top X_\star v_i \overset{(b)}{\geq} T \sum_{i=r+1}^{n} v_i^\top X_\star v_i = \mathbf{tr}(UX_\star U^\top) = \mathbf{tr}(X_1).$$

Here step (a) is due to strong duality and we use $v_i^\top X_\star v_i \geq 0$ in step (b) as $X_\star \succeq 0$. Combining pieces and $\mathbf{tr}(X_1) \geq 0$ as $X_1 \succeq 0$, we find that

$$\mathbf{tr}(X_\star) \geq \mathbf{tr}(VV^\top X_\star VV^\top) \geq \mathbf{tr}(X_\star) - \frac{\epsilon}{T}.$$

This completes the argument.

5. Computational aspects of primal recovery.

The previous section introduced two methods, (MinFeasSDP) and (MinObjSDP), to recover an approximate primal from an approximate dual solution y. It contains theoretical bounds on suboptimality, infeasibility, and distance to the solution set of the primal SDP (P). We summarize this approach as Algorithm 5.1.

In this section, we turn this approach into a practical optimal storage algorithm by answering the following questions:

1. How should we solve (MinFeasSDP) and (MinObjSDP)?
2. How should we choose δ in (MinObjSDP)?
3. How should we choose the rank parameter r?
4. How can we estimate the suboptimality, the infeasibility, and (possibly) the distance to the solution to use as stopping conditions?

In particular, our choices for algorithmic parameters should not depend on any quantities that are unknown or difficult to compute. We address each question in turn.

For this discussion, let us quantify the cost of the three data access oracles (1.4). We use the mnemonic notation L_C, $L_{\mathcal{A}}$, and $L_{\mathcal{A}^\top}$ for the respective running time (denominated in flops) of the three operations.

5.1. Solving MinFeasSDP and MinObjSDP.

Suppose that we have a dual estimate $y \in \mathbf{R}^m$ and that we have chosen $r = \mathcal{O}(r_\star)$ and δ. Each recovery problem, (MinFeasSDP) and (MinObjSDP), is an SDP with an $r \times r$ decision variable and m linear constraints. We now discuss how to solve them with optimal storage $\mathcal{O}(m + nr)$. First, we present four operators that form the computational core of all the storage-optimal algorithms we consider here. We list their input and output dimension, storage requirement (sum of input output dimensions), and time complexity in evaluating these operators in Table 4.

Algorithm 5.1 Primal recovery via (MinFeasSDP) or (MinObjSDP).

```
Require: Problem data $\mathcal{A}$, $C$, and $b$; dual vector $y$ and positive integer $r$
1: Compute an orthonormal matrix $V \in \mathbf{R}^{n\times r}$ whose range is an invariant subspace
   of $C - \mathcal{A}^\top y$ associated with the $r$ smallest eigenvalues.
2: Option 1: Solve (MinFeasSDP) to obtain a matrix $\hat{S}_1 \in \mathbf{S}^r_+$.
3: Option 2: Solve (MinObjSDP) by setting $\delta = \gamma\|\mathcal{A}_V(\hat{S}_1) - b\|$ with some $\gamma \geq 1$,
   where $\hat{S}_1$ is obtained from solving (MinFeasSDP). Obtain $\hat{S}_2$.
4: return $(V, S_1)$ for option 1, and $(V, S_2)$ for option 2.
```

TABLE 4

Required operators for solving MinFeasSDP and MinObjSDP. The operators $P_{\mathbf{B}_\delta}$ and $P_{\mathbf{S}_+^r}$ are projections of the ℓ_2 norm ball $\mathbf{B}_\delta := \{y \in \mathbf{R}^m \mid \|y\|_2 \le \delta\}$ and of the PSD matrices of side dimension r, $\mathbf{S}_+^r$, respectively.

Operator	Input	Output	Storage req.	Time compl.
$\mathcal{A}_V$	$S \in \mathbf{S}^r$	$\mathcal{A}_V(S) \in \mathbf{R}^m$	$r^2 + m$	$\mathcal{O}(r^2 L_{\mathcal{A}})$
$\mathcal{A}_V^\top$	$y \in \mathbf{R}^m$	$V^\top(\mathcal{A}^\top(y))V \in \mathbf{S}^r$	$m + r^2$	$\mathcal{O}(rL_{\mathcal{A}^\top} + nr^2)$
$P_{\mathbf{B}_\delta}$	$y \in \mathbf{R}^m$	$P_{\mathbf{B}_\delta}(y) \in \mathbf{R}^m$	$2m$	$\mathcal{O}(m)$
$P_{\mathbf{S}_+^r}$	$S \in \mathbf{S}^r$	$P_{\mathbf{S}_+^r}(S) \in \mathbf{S}^r$	$2r^2$	$\mathcal{O}(r^3)$

Any algorithm that uses a constant number of calls to these operators at each iteration (and at most $\mathcal{O}(m+nr)$ additional storage) achieves optimal storage $\mathcal{O}(m+nr)$. To be concrete, we describe algorithms to solve (MinFeasSDP) and (MinObjSDP) that achieve optimal storage. Many other algorithmic choices are possible.

- For (MinFeasSDP), we can use the accelerated projected gradient method [50]. This method uses the operators $\mathcal{A}_V, \mathcal{A}_V^\top$, and $P_{\mathbf{S}_+^r}$. Each iteration requires one call each to $\mathcal{A}_V$, $\mathcal{A}_V^\top$, and $P_{\mathbf{S}_+^r}$ and a constant number of additions in $\mathbf{R}^m$ and $\mathbf{S}_+^r$. Hence the per-iteration flop count is $\mathcal{O}(r^2 L_{\mathcal{A}} + rL_{\mathcal{A}^\top} + m + r^2 n)$. As for storage, the accelerated projected gradient method requires $\mathcal{O}(m + r^2)$ working memory to store the residual $\mathcal{A}_V(S) - b$, the computed gradient, and iterates of size r^2. Hence this method is storage optimal.
- For (MinObjSDP), we can use the Chambolle–Pock method [21] as described in [26, section SM1]. This method requires access to the operators $\mathcal{A}_V, \mathcal{A}_V^\top$, $P_{\mathbf{B}_\delta}$, and $P_{\mathbf{S}_+^r}$. It also stores the matrix $C_V = V^T CV \in \mathbf{R}^{r\times r}$ explicitly. We can compute C_V in $r^2 L_C$ time and store it using r^2 storage. Each iteration requires one call each to $\mathcal{A}_V$, $\mathcal{A}_V^\top$, $P_{\mathbf{B}_\delta}$, and $P_{\mathbf{S}_+^r}$ and a constant number of additions in $\mathbf{R}^m$ and $\mathbf{S}_+^r$. Hence the per-iteration flop count is $\mathcal{O}(r^2 L_{\mathcal{A}} + rL_{\mathcal{A}^\top} + m + r^2 n)$. As for storage, the Chambolle–Pock method requires $\mathcal{O}(m + r^2)$ working memory to store the residual $\mathcal{A}_V(S) - b$, one dual iterate of size m, two primal iterates of size r^2, and a few intermediate quantities of size r^2 or m. Hence the method is again storage optimal.

5.2. Choosing the rank parameter r. Theorem 4.1 shows that (Min FeasSDP) recovers the solution when the rank estimate r is accurate. Alas, as r increases, (MinFeasSDP) can have multiple solutions. Hence it is important to use information about the objective function as well (e.g., using (MinObjSDP)) to recover the solution to (P)—in theory. In practice, we find that (MinFeasSDP) recovers the primal solution well so long as r satisfies the Barvinok–Pataki bound $\frac{r(r+1)}{2} \le m$.

Theorem 4.6 shows that (MinObjSDP) is more robust and provides useful results so long as the rank estimate r exceeds the true rank $r_\star$. Indeed, the quality of the solution improves as r increases. A user seeking the best possible solution to (MinObjSDP) should choose the largest rank estimate r for which the SDP (MinObjSDP) can still be solved, given computational and storage limitations.

It is tempting to consider the spectrum of the dual slack matrix $C - \mathcal{A}^\top y$, and in particular its smallest eigenvalues, to guess the true rank of the solution. We do not know of any reliable rules that use this idea.

5.3. Choosing the infeasibility parameter δ. To solve (MinObjSDP), we must choose a bound δ on the acceptable infeasibility. (Recall that (MinObjSDP) is generally not feasible when $\delta = 0$.) This bound can be chosen using the result of (MinFeasSDP). Concretely, solve (MinFeasSDP) to obtain a solution X_{infeas}. Then set $\delta = \gamma \left\|\mathcal{A}_V(X_{\text{infeas}}) - b\right\|_2$ for some $\gamma \geq 1$. This choice guarantees that (MinObjSDP) is feasible. In our numerics, we find $\gamma = 1.1$ works well.[2]

5.4. Bounds on suboptimality, infeasibility, and distance to the solution set. Suppose we solve either (MinObjSDP) or (MinFeasSDP) to obtain a primal estimate $X = X_{\text{obj}}$ or $X = X_{\text{infeas}}$. How can we estimate the suboptimality, infeasibility, and distance of X to the solution set of (P)?

The first two metrics are straightforward to compute. We can bound the suboptimality by $\epsilon_p(X) \leq \mathbf{tr}(CX) - b^\top y$. We can compute the infeasibility as $\delta_p(X) = \left\|\mathcal{A}X - b\right\|_2$. In the optimization literature, scaled versions of the suboptimality and infeasibility called KKT residuals [70, 47, 77] are generally used as stopping criteria.

The distance to the solution requires additional assumptions, such as surjectivity of the restricted constraint map $\mathcal{A}_V$. With these assumptions, Lemma 4.4 yields a computable (but possibly loose) bound. We refer the interested reader to the supplement [26].

6. Computational aspects of the dual SDP (D).

The previous two subsections showed how to efficiently recover an approximate primal solution from an approximate dual solution. We now discuss how to (approximately) solve the dual SDP (D) with optimal storage and with a low per-iteration computational cost. Together, the (storage-optimal) dual solver and (storage-optimal) primal recovery compose a new algorithm for solving regular SDPs with optimal storage.

6.1. Exact penalty formulation. It will be useful to introduce an unconstrained version of the dual SDP (D), parametrized by a real positive number α, which we call the penalized dual SDP:

$$\text{maximize} \quad b^\top y + \alpha \min\{\lambda_{\min}(C - \mathcal{A}^\top y), 0\}. \tag{6.1}$$

That is, we penalize vectors y that violate the dual constraint $C - \mathcal{A}^\top y \succeq 0$.

Problem (6.1) is an exact penalty formulation for the dual SDP (D). Indeed, problem (6.1) is the dual of the primal SDP (P) enhanced with a trace constraint $\mathbf{tr}(X) \leq \alpha$: problem (6.1) can be rewritten as $\max_y \min_{X \succeq 0, \mathbf{tr}(X) \leq \alpha} \mathbf{tr}(CX) + (b - \mathcal{A}X)^\top y$. The following lemma shows that the solution of problem (6.1) and the solution set of the dual SDP (D) are the same when α is large enough. The proof is based on [60, Theorem 7.21].

LEMMA 6.1. *Instate the assumptions in subsection* 1.1. *If $b \neq 0$ and $\alpha > \mathbf{tr}(X_\star)$, then the penalized dual SDP* (6.1) *and the dual SDP* (D) *have the same solution $y_\star$.*

Proof of Lemma 6.1. We first note that the dual solution $y_\star$ is the only solution to $\min_{\lambda_{\max}(\mathcal{A}^\top y - C) \leq 0} -b^\top y$. Using [60, Theorem 7.21], we know that the penalty form (6.1) has $y_\star$ as its only solution as long as $\alpha > \alpha_0$ for any $\alpha_0 \geq 0$ satisfying the KKT condition:

$$b \in \alpha_0 \mathcal{A}\left(\partial(\lambda_{\max}(-Z(y_\star)))\right) \quad \text{and} \quad \alpha_0 \lambda_{\max}(-Z(y_\star)) = 0.$$

This is the case by noting $X_\star \in \mathbf{tr}(X_\star)\partial(\lambda_{\max}(-Z(y_\star)))$, $\mathbf{tr}(X_\star)\lambda_{\max}(-Z(y_\star)) = 0$, and $\mathcal{A}(X_\star) = b$. Hence we can choose $\alpha_0 = \mathbf{tr}(X_\star)$. ∎

[2] It is sometimes possible to choose δ directly without solving (MinFeasSDP) [26, Theorem SM2.1]; however, this approach is rarely practical.

Algorithm 6.1 Dual algorithm + primal recovery.

```
Require: Problem data $\mathcal{A}$, $C$ and $b$
Require: Positive integer $r$ and an iterative algorithm $\mathcal{G}$ for solving the dual SDP
1: for $k = 1, 2, \dots$ do
2:     Compute the $k$th dual $y_k$ via $k$th iteration of $\mathcal{G}$
3:     Compute a recovered primal $\hat{X}_k = V\hat{S}V^\top$ using primal recovery, Algorithm 5.1.
4: end for
```

Remark 6.2. The objective of problem (6.1) lower bounds the optimal value $p_\star$ for any $y \in \mathbf{R}^m$ when $\alpha \geq \mathbf{tr}(X_\star)$. In contrast, the dual SDP (P) provides a valid bound only if y is feasible.

Thus, as long as we know an upper bound on the nuclear norm of the primal solution, then we can solve problem (6.1) to find the dual optimal solution $y_\star$. It is often easy to find a bound on $\|X_\star\|_*$:

1. *Nuclear norm objective.* Suppose the objective in (P) is $\|X\|_* = \mathbf{tr}(X)$. Problems using this objective include matrix completion [19], phase retrieval [20], and covariance estimation [22]. In these settings, it is generally easy to find a feasible solution or to bound the objective via a spectral method. (See [41] for matrix completion and [18] for phase retrieval.)
2. *Constant trace constraints.* Suppose the constraint $\mathcal{A}X = b$ enforces $\mathbf{tr}(X) = \beta$ for some constant β. Problems with this constraint include MaxCut [33], community detection [48], and PhaseCut in [68]. Then any $\alpha > \beta$ serves as an upper bound. In the powerflow problem [7, 46], we have constraints $X_{ii} \leq \beta_i$ for all$i$. Then any $\alpha > \sum_{i=1}^n \beta_i$ serves as an upper bound. (The powerflow problem does not directly fit into our standard form (P), but a small modification of our framework can handle the problem.)

When no such bound is available, we may search over α numerically. For example, solve problem (6.1) for $\alpha = 2, 4, 8, \dots, 2^d$ for some integer d (perhaps, in parallel, simultaneously). Since any feasible y for the dual SDP (D) may be used to recover the primal, using (MinFeasSDP) and (MinObjSDP), we can use any approximate solution of the penalized dual SDP, problem (6.1), for any α, as long as it is feasible for the dual SDP.

Alternatively, we can use the method of [58] to solve (D) directly if a strictly feasible dual point is known. For example, when $C \succ 0$, $y = 0 \in \mathbf{R}^m$ is strictly feasible.

6.2. Computational cost and convergence rate for primal approximation.

Suppose we have an iterative algorithm $\mathcal{G}$ to solve the dual problem. Denote by y_k the kth iterate of $\mathcal{G}$. Each dual iterate y_k generates a corresponding primal iterate using either (MinFeasSDP) or (MinObjSDP). We summarize this approach to solving the primal SDP in Algorithm 6.1.

The primal iterates X_k generated by Algorithm 6.1 converge to a solution of the primal SDP (P) by our theory.[3] However, it would be computational folly to recover the primal at every iteration: the primal recovery problem is much more

[3]Iterative algorithms for solving the dual SDP (D) may not give a feasible point y. If a strictly feasible point is available, we can produce a sequence of feasible points from a sequence of (possibly infeasible) iterates by averaging the sequence with the strictly feasible point; see [26, Lemmas SM3.1 and SM3.2]. This technique preserves the convergence rate. Alternatively, our theory can be extended to handle the infeasible case; we omit this analysis for simplicity.

computationally challenging than a single iteration of most methods for solving the dual. Hence, to determine when (or how often) to recover the primal iterate from the dual, we would like to understand how quickly the recovered primal iterates converge to the solution of the primal problem.

To simplify exposition as we discuss algorithms for solving the dual, we reformulate the penalized dual SDP as a convex minimization problem,

$$\text{minimize} \quad g_\alpha(y) := -b^\top y + \alpha \max\{\lambda_{\max}(-C + \mathcal{A}^\top y), 0\}, \tag{6.2}$$

which has the same solution set as the penalized dual SDP (6.1).

We focus on the convergence of suboptimality and infeasibility, as these two quantities are easier to measure than distance to the solution set. Recall from Table 3 that

$$\epsilon\text{-optimal dual feasible } y \xrightarrow[\text{or MinFeasSDP}]{\text{MinObjSDP}} (\mathcal{O}(\sqrt{\epsilon}), \mathcal{O}(\sqrt{\epsilon}))\text{-primal solution } X \tag{6.3}$$

if $\kappa = \mathcal{O}(1)$. Thus the convergence rate of the primal sequence depends strongly on the convergence rate of the algorithm we use to solve the penalized dual SDP.

6.2.1. Subgradient methods, storage cost, and per-iteration time cost.

We focus on subgradient-type methods for solving the penalized dual SDP (6.1), because the objective g_α is nonsmooth but has an efficiently computable subgradient. Any subgradient method follows a recurrence of the form

$$y_0 \in \mathbf{R}^m \quad \text{and} \quad y_{k+1} = y_k - \eta_k g_k, \tag{6.4}$$

where g_k is a subgradient of g_α at y_k and $\eta_k \geq 0$ is the step size. Subgradient-type methods differ in the methods for choosing the step size η_k and in their use of parallelism. However, they are all easy to run for our problem because it is easy to compute a subgradient of the dual objective with penalty g_α.

LEMMA 6.3. *Let $Z = C - \mathcal{A}^\top y$. The subdifferential of the function g_α is*

$$\partial g_\alpha(y) = \begin{cases} -b + \mathbf{conv}\{\alpha \mathcal{A}(vv^\top) \mid Zv = \lambda_{\min}(Z)v\}, & \lambda_{\min}(Z) < 0, \\ -b, & \lambda_{\min}(Z) > 0, \\ -b + \beta\, \mathbf{conv}\{\alpha \mathcal{A}(vv^\top) \mid Zv = \lambda_{\min}(Z)v, \beta \in [0,1]\}, & \lambda_{\min}(Z) = 0. \end{cases}$$

This result follows directly via standard subdifferential calculus from the subdifferential of the maximum eigenvalue $\lambda_{\max}(\cdot)$. Thus our storage cost is simply $\mathcal{O}(m+n)$, where m is due to storing the decision variable y and the gradient g_k, and n is due to the intermediate eigenvector $v \in \mathbf{R}^n$. The main computational cost in computing a subgradient of the objective in (6.2) is computing the smallest eigenvalue $\lambda_{\min}(C - \mathcal{A}^\top y)$ and the corresponding eigenvector v of the matrix $C - \mathcal{A}^\top y$. Since $C - \mathcal{A}^\top y$ can be efficiently applied to vectors (using the data access oracles (1.4)), we can compute this eigenpair efficiently using the randomized Lanczos method [42].

6.2.2. Convergence rate of the dual and primal.

The best available subgradient method [40] has convergence rate $\mathcal{O}(1/\epsilon)$ when the quadratic growth condition is satisfied. (This result does not seem to appear in the literature for SDP; however, it is a simple consequence of [40, Table 1] together with the quadratic growth condition proved in Lemma 3.1.) Thus, our primal recovery algorithm has convergence rate $\mathcal{O}(1/\sqrt{\epsilon})$, using the relation between dual convergence and primal convergence in (6.3). Unfortunately, the algorithm in [40] involves many unknown constants. In practice, we recommend using dual solvers that require less tuning such as AcceleGrad [45], which is the one we used in section 7.

Table 5
Problems for numerics.

MaxCut	Matrix completion
minimize $\frac{1}{\xi}\,\mathbf{tr}(-LX)$ subject to $\mathbf{diag}(X) = \frac{1}{\beta}\mathbf{1}$ $X \succeq 0$	minimize $\frac{1}{\xi}\left(\mathbf{tr}(W_1) + \mathbf{tr}(W_2)\right)$ subject to $X_{ij} = \frac{1}{\beta}\bar{X}_{ij},\ (i,j) \in \Omega$ $\begin{bmatrix} W_1 & X \\ X^\top & W_2 \end{bmatrix} \succeq 0$

7. Numerical experiments. This section demonstrates the effectiveness of our methods numerically. We first show that Algorithm 5.1 (primal recovery) recovers an approximate primal given an approximate dual solution. Next, we show that Algorithm 6.1 with primal recovery achieves reasonable accuracy ($10^{-1} \sim 10^{-2}$) for large-scale SDP, e.g., $10^5 \times 10^5$, with substantially lower storage requirements compared to other SDP solvers.

We test our methods on the MaxCut and matrix completion SDPs, defined in Table 5.

For MaxCut, L is the Laplacian of a given graph. For matrix completion, Ω is the set of indices of the observed entries of the underlying simulated matrix $\bar{X} \in \mathbf{R}^{n_1 \times n_2}$. We set $\xi = \|L\|_{\mathrm{F}}$ and $\beta = 1.1n$ for MaxCut and $\xi = \frac{1}{\sqrt{n_1+n_2}}$ and $\beta = 2.2\left\|\bar{X}\right\|_*$ for matrix completion in all experiments.

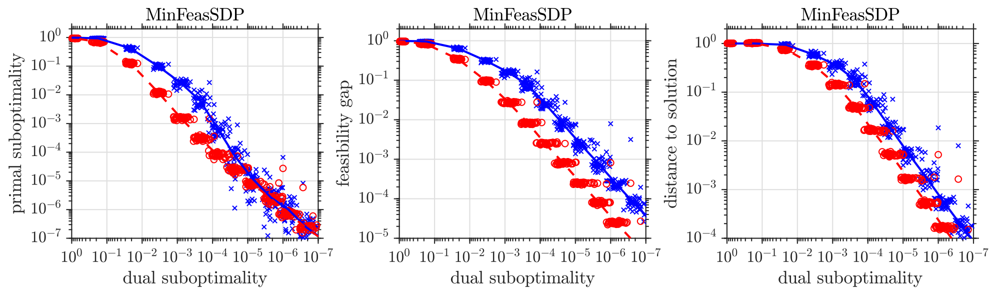
Problem scaling. We can improve the conditioning of the penalized dual problem by reparametrizing the problem with a rescaled objective and constraints. These changes preserve the solution set (up to a simple rescaling). A reasonable scaling is crucial in practice, since we use first-order methods to solve the dual. In our experiments, we scale the problem (the choice of β and ξ) so that the cost matrix $\|C\|_{\mathrm{F}} = 1$ and the penalty parameter in (6.1) $\alpha = 1$. This scaling also ensures the trace of the solution is independent of the problem size n, which makes it easier to compare performance across different problems.

7.1. Primal recovery. Our first experiment confirms numerically that Algorithm 5.1 (primal recovery) recovers an approximate primal from an approximate dual solution, validating our theoretical results. As an example, we present results for the MaxCut SDP from the `G1` dataset from the GSet group[2] (with $n = 800$ nodes). Results for matrix completion and for other MaxCut problems are similar; corresponding experiments for matrix completion can be found in [26, subsection SM5.3]. To evaluate our method, we compare the recovered primal with the primal dual solution $X_\star$, $y_\star$ obtained with SeDuMi, an interior point solver [62]. Empirically, the rank of the primal solution $r_\star = 13$.

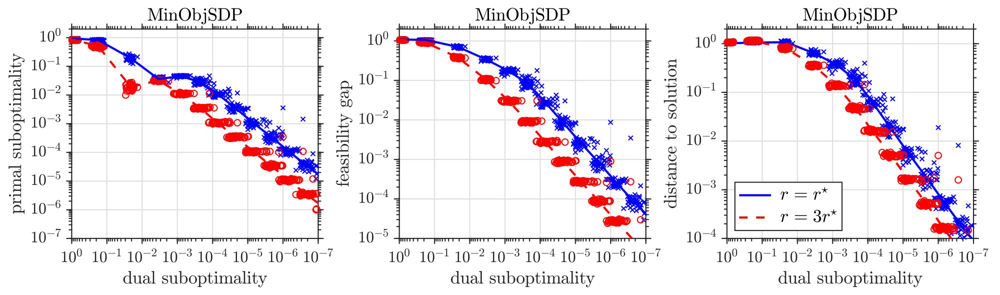
We perturb the true dual solution $y_\star$ to generate approximate dual solutions

$$y = y_\star + \varepsilon s \left\|y_\star\right\|_2,$$

where ε is the noise level, which we vary from 1 to 10^{-5}, and s is a uniformly random vector on the unit sphere in $\mathbf{R}^m$. For each perturbed dual y and for each rank estimate $r \in \{r_\star, 3r_\star\}$, we first solve (MinFeasSDP) to obtain a solution X_{infeas}, and then solve (MinObjSDP) with $\delta = 1.1\left\|\mathcal{A}X_{\text{infeas}} - b\right\|_2$. We measure the suboptimality of the perturbed dual using the relative suboptimality $\frac{|p^\star + g_\alpha(y)|}{|p^\star|}$. We measure the distance of the recovered primal to the solution in three ways: relative suboptimality $|\,\mathbf{tr}(CX) - p_\star|/p_\star$, relative infeasibility $\|\mathcal{A}X - b\|/\|b\|$, and relative distance to the solution set $\|X - X_\star\|_{\mathrm{F}} / \|X_\star\|_{\mathrm{F}}$.



(a) (MinFeasSDP).



(b) (MinObjSDP).

FIG. 1. *The plots show the primal recovery performance of* (MinFeasSDP) *(upper) and* (MinObjSDP) *(lower) in terms of (relative) primal suboptimality* $|\mathbf{tr}(CX) - p_\star|/p_\star$*, (relative) infeasibility gap* $\|\mathcal{A}X - b\|/\|b\|$*, and (relative) distance to solution* $\|X - X_\star\|_F \,/\, \|X_\star\|_F$*. The horizontal axis is (relative) dual suboptimality* $\frac{|p^\star + g_\alpha(y)|}{|p^\star|}$*. The blue dots corresponds to the choice* $r = r_\star$ *and the red dots corresponds to the choice* $r = 3r_\star$ *in Algorithm* 6.1*.*

Figure 1 shows distance of the recovered primal to the solution. The blue dots show the primal recovered using $r = r_\star$, while the red dots show the primal recovered using $r = 3r_\star$. The blue and red curves are fit to the dots of the same color to provide a visual guide. The red line ($r = 3r_\star$) generally lies below the blue line ($r = r_\star$), which confirms that larger ranks produce more accurate primal reconstructions.

These plots show that the recovered primal approaches the true primal solution as the dual suboptimality approaches zero, as expected from our theory.[4] From Table 3, recall that we expect the primal solution recovered from an ϵ suboptimal dual solution to converge to the true primal solution as $\mathcal{O}(\sqrt{\epsilon})$ with respect to all three measures. The plots confirm this scaling for distance to solution and infeasibility, while suboptimality decays even faster than predicted by our theory. By construction, the primal suboptimality of (MinObjSDP) is smaller than that of (MinFeasSDP); however, the plots measure primal suboptimality by its absolute value. The kink in the curves describing primal suboptimality for (MinObjSDP) separates suboptimal primal solutions (to the left) from superoptimal solutions (to the right). Finally, notice that $3r_\star = 39$ is close to the Barvinok–Pataki bound. Interestingly, (MinFeasSDP) still performs better with this large feasible set ($r = 3r_\star$) than with a smaller one ($r = r_\star$), although our theory does not apply.

[4]To be precise, the theory we present in Theorems 4.1 and 4.6 requires the approximate dual solution to be feasible, while y may be infeasible in our experiments. An extension of our results can show similar bounds when y is infeasible but $g_\alpha(y)$ is close to $-d_\star$.

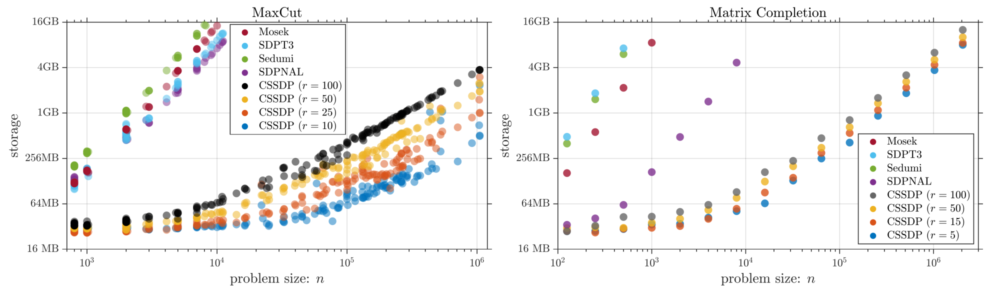


FIG. 2. *Storage considerations. We compare storage cost of Algorithm* 6.1 *with different choices of* r *against other SDP solvers: Mosek, SDPT*3*, SeDuMi, and SDPNAL+. Each marker represents one problem instance. See subsection* 7.2 *for details.*

7.2. Storage considerations. In this section, we demonstrate that our methods require significantly less storage compared to existing algorithms. Below, our method Algorithm 6.1 is labeled as CSSDP (complementary slackness SDP).

We compare Algorithm 6.1 with AcceleGrad as the dual solver against Mosek [49], SDPT3 [66], SeDuMi [62], and SDPNAL+ [64]. Figure 2 shows that the storage required for these methods scales with the side length n of the primal variable. Each datapoint corresponds to a problem instance from GSet [2] and DIMACS10 [1] groups for MaxCut, and a simulated problem for matrix completion.[5]

The plots show that other methods all exceed our storage limit when $n \sim 10^4$. In contrast, our method scales linearly with the dimension (for any r) and can tackle problems with $n = 10^6$.

7.3. Detailed comparison with other SDP solvers. In this section, we compare the empirical performance of our methods against other SDP solvers for medium size MaxCut problems. We solve the MaxCut problem with all 67 datasets from the GSet group. We present only problems with dimension $n \geq 7000$; additional comparisons can be found in [26, subsection SM5.7]. We use Algorithm 6.1 with AcceleGrad as the dual solver. We solve both (MinFeasSDP) and (MinObjSDP) (with $\delta = 1.1 \|\mathcal{A}X_{\text{infeas}} - b\|_2$) with $r = \{10, \lceil\sqrt{2n}\rceil\}$. We compare with other SDP solvers: Mosek, SDPT3, SeDuMi, and SDPNAL+. We set all solver tolerances to 10^{-4}; for AcceleGrad, we stop when the norm of the average of all past subgradients is $< 10^{-4}$.

Table 6 reports the DIMACS measures [39],[6] cut value after rounding, total runtime, and storage cost of each method for each dataset. Our method is labeled CSSDP MF (i.e., (MinFeasSDP)) or MO (i.e., (MinObjSDP)). For larger problems, our method significantly outperforms other solvers in terms of storage as well as runtime. For the `G67` dataset ($n = 10^4$), our methods are more than $\sim$10 times faster, and they use $\sim$100 times less storage.

[5]We generate rank 5 random matrices $\bar{X} = UV \in \mathbf{R}^{n_1 \times n_2}$ where $U \in \{\pm 1\}^{n_1 \times 5}$ and $V \in \{\pm 1\}^{5 \times n_2}$ are random sign matrices. We vary the dimensions by setting $n = 75c$, $m = 50c$ and varying $c = 10, 100, 1000, \ldots$. The $25(n_1 + n_2)\log(n_1 + n_2)$ observations are chosen uniformly at random.

[6]DIMACS measures are basically KKT errors and common in comparing performance of solvers. The quantities err_1 and err_2 measure the primal infeasibility, err_3 and err_4 measure the dual infeasibility, err_5 measures the primal-dual gap, and err_6 measures the violation in complementary slackness. See [39] and [26, SM5.2.1] for more details.

TABLE 6
MaxCut experiments. err_1 and err_2 measure the primal infeasibility, err_3 and err_4 measure the dual infeasibility, err_5 measures the primal-dual gap, and err_6 measures the violation in complementary slackness, [39] *and* [26, SM5.2.1].

Data name size (n)		MinFeas $r=10$	MinObj $r=10$	MinFeas $r=\lceil\sqrt{2n}\rceil$	MinObj $r=\lceil\sqrt{2n}\rceil$	Sedumi	Mosek	SDPNAL	SDPT3	SDPT3 High Acc.
G63 7000	cut value	25990	25957	25922	25888	25929	25982	25931	25998	25988
	err1	3.23E-03	3.56E-03	4.52E-04	5.01E-04	1.54E-03	7.40E-05	3.59E-05	0	2.18E-14
	err2	0	0	0	0	0	0	1.34E-15	0	0
	err3	0	0	0	0	0	5.54E-05	1.41E-04	1.49E-12	5.21E-13
	err4	0	0	0	0	1.54E-05	0	0	0	0
	err5	5.85E-04	-5.50E-04	5.56E-05	-1.89E-04	-8.54E-04	-4.15E-05	-4.90E-05	3.93E-05	6.91E-14
	err6	2.35E-05	2.68E-05	4.54E-05	4.32E-05	3.38E-04	1.69E-05	0	3.93E-05	7.88E-14
	time (s)	257	272	264	1.29E+03	2.50E+04	6.96E+03	2.60E+04	2.26E+03	3.43E+03
	storage (MB)	39	39	57	57	11941	7238	6248	4899	5048
G64 7000	cut value	7741	7792	7669	7700	7640	7732	7690	7735	7746
	err1	2.44E-03	2.69E-03	1.86E-04	2.05E-04	1.51E-03	1.85E-03	2.34E-05	0	2.85E-15
	err2	0	0	0	0	0	0	4.52E-15	0	0
	err3	0	0	0	0	0	1.77E-03	1.21E-04	1.91E-12	1.28E-12
	err4	8.70E-06	8.70E-06	8.70E-06	8.70E-06	5.50E-06	0	0	0	0
	err5	3.43E-04	-1.02E-03	1.35E-04	-2.27E-04	-3.06E-05	-5.69E-05	-3.90E-05	7.00E-05	1.39E-13
	err6	9.85E-06	1.08E-05	1.06E-04	5.35E-05	1.79E-03	2.19E-03	-3.97E-15	7.00E-05	1.54E-13
	time (s)	244	258	262	1.26E+03	2.60E+04	6.78E+03	2.30E+04	2.21E+03	3.52E+03
	storage (MB)	35	37	57	57	11940	7232	5966	5048	5048

TABLE 6
(cont.).

	MinFeas $r = 10$	MinObj $r = 10$	MinFeas $r = \lceil\sqrt{2n}\rceil$	MinObj $r = \lceil\sqrt{2n}\rceil$	Sedumi	Mosek	SDPNAL	SDPT3	SDPT3 High Acc.
G65 8000	5096	5092	5060	5046	5210	5202	5064	5086	5010
	7.79E-03	8.57E-03	1.33E-03	1.47E-03	8.86E-04	1.03E-03	2.15E-05	0	6.74E-14
	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	1.02E-03	1.63E-04	4.40E-11	1.25E-12
	0	0	0	0	0	0	0	0	0
	6.32E-03	3.71E-03	1.54E-04	-8.70E-04	-1.75E-05	-4.73E-05	-2.01E-05	8.73E-05	5.26E-14
	9.81E-06	1.51E-05	4.02E-05	4.42E-05	9.39E-04	1.08E-03	0	8.73E-05	1.37E-13
	206	214	239	597	1.72E+04	6.11E+03	3.53E+04	3.09E+03	3.95E+03
	35	36	63	63	15064	9433	7985	6021	6021
G66 9000	5872	5858	5776	5772	—	5926	5808	5838	5766
	7.41E-03	8.18E-03	1.60E-03	1.77E-03	—	1.00E-03	1.92E-05	0	0
	0	0	0	0	—	0	0	0	0
	0	0	0	0	—	9.89E-04	1.55E-04	4.07E-11	1.39E-13
	0	0	0	0	—	0	0	0	0
	6.22E-03	3.70E-03	2.83E-04	-8.43E-04	—	-5.75E-05	-1.71E-05	8.49E-05	1.10E-14
	9.82E-06	1.55E-05	4.11E-05	4.58E-05	—	1.05E-03	0	8.49E-05	1.16E-14
	219	224	232	667	—	8.60E+03	5.18E+04	4.62E+03	6.13E+03
	38	38	84	84	—	11639	10240	7614	7614
G67 10000	6352	6352	6306	6328	—	6490	6310	6344	6250
	7.13E-03	7.87E-03	1.61E-03	1.78E-03	—	1.08E-03	1.83E-05	0	3.61E-14
	0	0	0	0	—	0	0	0	0
	0	0	0	0	—	1.06E-03	1.49E-04	8.60E-11	1.97E-12
	0	0	0	0	—	0	0	0	0
	5.96E-03	3.63E-03	2.90E-04	-6.63E-04	—	-4.72E-05	-1.69E-05	7.11E-05	7.92E-13
	9.08E-06	1.44E-05	4.01E-05	4.41E-05	—	1.12E-03	0	7.11E-05	8.48E-13
	205	210	245	751	—	1.19E+04	6.03E+04	7.61E+03	8.68E+03
	38	39	93	93	—	14670	12645	9399	9399

With these stopping criteria, the DIMACS measures [39] for each solver reach $10^{-3} \sim 10^{-5}$. We argue this accuracy suffices: recall that the MaxCut SDP is a relaxation from which a rounding scheme[7] generates a cut from the SDP solution $X_\star$. To support our claim, we evaluate and compare the cut values after rounding and compare them to the cut values after rounding a high accuracy solution obtained with SDPT3 by setting its stopping criteria `gaptol` and `inftol` to 10^{-12}. After rounding, we find the moderately accurate solution from our solver provides a better cut than the high-accuracy solution obtained with SDPT3!

7.4. Convergence rate and runtime for large-scale SDP. We investigate the empirical convergence of our algorithm for the MaxCut problem with the small-world graph from the DIMACS10 group with $n = 10^5$ nodes and for the matrix completion problem with simulated data (as described in footnote 5) with $n = n_1 + n_2 =$ 125,000 and $m \sim 3.6 \times 10^7$ constraints. We work with large problems beyond the limits of other solvers in subsection 7.2 to demonstrate the scalability of our methods.

We run AcceleGrad to solve the dual problem (6.1) and save the dual variable y at iterations $1, 10, 10^2, 10^3, \ldots$. For each case, we consider both (MinFeasSDP) and (MinObjSDP) for recovering the primal variable. Since we do not know the optimal solution, we track performance using the following relative feasible (feas.) gap and relative primal-dual (p-d) gap:

(7.1)
$$\text{relative feas. gap:} \quad \frac{\|\mathcal{A}(X) - b\|_2}{\|b\|_\infty + 1}; \quad \text{relative p-d gap:} \quad \frac{|\,\mathbf{tr}(CX) + g_\alpha(y)|}{|\,\mathbf{tr}(CX)| + |g_\alpha(y)| + 1}.$$

We note that $\mathbf{tr}(CX) + g_\alpha(y)$ bounds primal suboptimality due to Lemma 6.1. It is traditional to use $\mathbf{tr}(CX) - b^\top y$; however, here y is not necessarily dual feasible and so this simpler measure does not bound primal suboptimality. Note these two measures completely characterize the convergence: a primal dual pair (X, y) is optimal if and only if their relative feasibility gap and primal-dual gap are 0. We also measure the total runtime: the time spent to solve the dual problem (up to that iteration) *plus* the time spent to recover the primal variable (at that iteration).

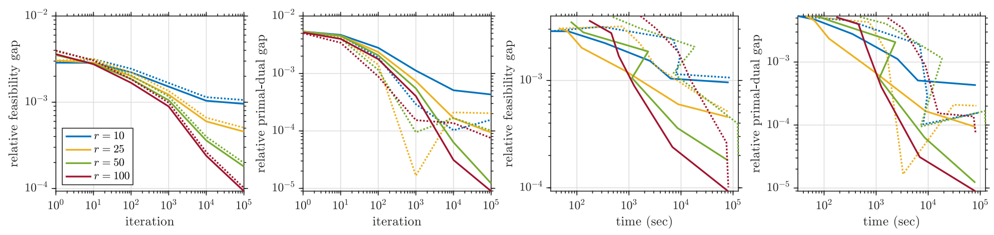
We present the results of this experiment in Figure 3.[8] The solid lines and dotted lines represent (MinFeasSDP) and (MinObjSDP), respectively. As can be seen, the proposed method solves each problem to moderate accuracy in a reasonable time when the rank parameter is sufficiently large; ranks of 10–100 suffice. These ranks are far smaller than the Barvinok–Pataki bound. Once again (MinFeasSDP) empirically outperforms (MinObjSDP).

In the plots, we connect the points according to the iteration counter in Figure 3: the top point on each line corresponds to iteration 1 (of the dual solver). We empirically observe that the primal recovery is faster from more accurate dual iterates, so earlier iterates sometimes correspond to longer runtimes.

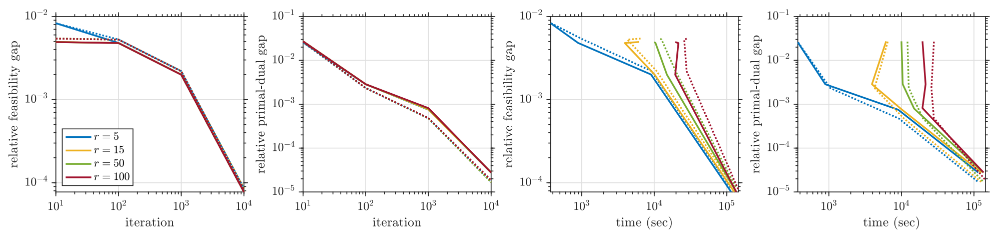
Last, we remark that the method works well on matrix completion problems, although the problem has multiple dual solutions [25, Theorem 5.1]. Hence the numerical evidence suggests that our method is effective even when our assumptions fail.

[7]Let $u_\star$ be an eigenvector that corresponds to the largest eigenvector of $X_\star$. In practice, $\mathbf{sign}(u_\star)$ provides an excellent solution to the MaxCut problem, while the randomized rounding [32] yields a near-optimal solution with guarantees.

[8]Results for the MaxCut problem and the matrix completion in terms of DIMACS measures can be found in Figure 4 and [26, subsection SM5.8], respectively.

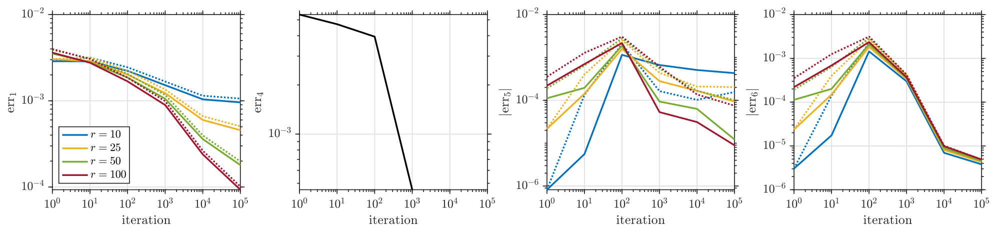


(a) MaxCut: smallworld from DIMACS10 ($n = m = 10^5$).

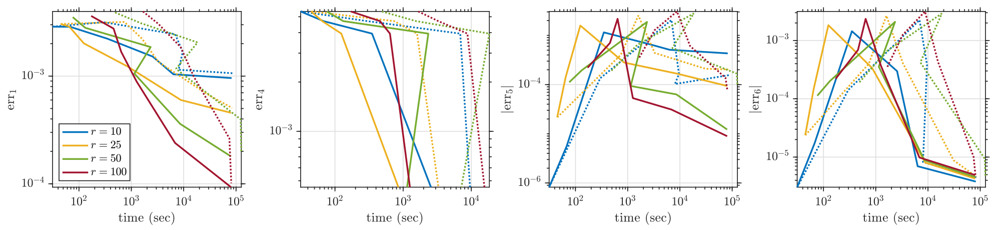


(b) Matrix Completion: simulated data ($n = 125'000,\ m \approx 3.6 \times 10^7$).

FIG. 3. *Convergence rate and runtime. Convergence of Algorithm* 6.1 *with* (MinFeasSDP) *as the solid line and* (MinObjSDP) *as the dotted line. Primal recovery from accurate dual iterates is both more accurate and faster, so primal iterates recovered from early dual iterates can be dominated by those recovered from later iterates.*



(a) MaxCut, DIMACS measures vs. iteration counter.



(b) MaxCut, DIMACS measures vs. time.

FIG. 4. *Convergence of Algorithm* 6.1 *measured by DIMACS errors with MinFeasSDP as the solid line and MinObjSDP as the dotted line for the MaxCut problem in subsection* 7.4.

7.5. Additional numerics. Interested readers may find details of the implementation, discussion on how to measure the solution quality, and additional numerics with alternate dual solvers (6.1), performance on other datasets, and a detailed comparison of our methods to existing solvers in [26, section SM5].

8. Conclusion and discussion.

This paper presents a new theoretically justified method to recover an approximate solution to a primal SDP from an approximate solution to a dual SDP, using complementarity between the primal and dual optimal solutions. We present two concrete algorithms for primal recovery, which offer guarantees on the suboptimality, infeasibility, and distance to the solution set of the recovered primal under the regular conditions on the SDP. The proposed method gives an advantage over existing solvers for large-scale, weakly constrained SDPs with an efficient data representation.

We use this primal recovery method to develop a storage-optimal algorithm to solve regular SDP: use any first-order algorithm to solve a penalized version of the dual problem, and recover a primal solution from the dual. This method requires $O(m + nr)$ storage: the storage is linear in the number of constraints m and in the side length n of the SDP variable, when the target rank r of the solution is fixed. These storage requirements improve on the storage requirements that guarantee convergence for nonconvex factored (Burer–Monteiro) methods to solve the SDP, which scale as $O(\sqrt{m}n)$. Furthermore, we show that no method can use less storage without a more restrictive data access model or a more restrictive representation of the solution. We demonstrate numerically that our algorithm is able to solve SDPS of practical interest including MaxCut and matrix completion.

In this work, we focused on SDPs with equality constraints only. Generalizing these ideas to handle SDPs with inequality constraints constitutes important future work.

Appendix A. Lemmas for section 1. To establish Lemma 3.1, we prove a lemma concerning the operator $\mathcal{A}_{V_\star}$.

LEMMA A.1. *Instate the hypothesis of subsection* 1.1. *Then* $\mathbf{null}(\mathcal{A}_{V^\star}) = \{0\}$.

Proof of Lemma A.1. Suppose by way of contradiction that $\ker(\mathcal{A}_{V_\star}) \neq \{0\}$. Let $S \in \ker(\mathcal{A}_{V_\star})$, so $\mathcal{A}_{V_\star}(S) = 0$. Recall $X_\star = V_\star S_\star (V_\star)^\top$ for some unique $S_\star \succ 0$. Hence for some $\alpha_0 > 0$, $S_\star + \alpha S \succeq 0$ for all $|\alpha| \leq \alpha_0$. Now pick any α with $|\alpha| \leq \alpha_0$ to see $\mathcal{A}(X_\alpha) = \mathcal{A}_{V_\star}(S_\star + \alpha S) = \mathcal{A}_{V_\star}(S_\star) + 0 = b$. Hence the matrix X_α is feasible for all $|\alpha| \leq \alpha_0$. But we can always find some $|\alpha| \leq \alpha_0$, $\alpha \neq 0$, so that $\mathbf{tr}(CX_\alpha) = p_\star + \alpha\, \mathbf{tr}(CV_\star(SV_\star)^\top) \leq p_\star$. This contradicts the assumption that $X_\star$ is unique. Hence we must have $\mathbf{null}(\mathcal{A}_{V_\star}) = \{0\}$. ∎

Proof of Lemma 3.1. Consider the linear operator $\mathcal{D}$ defined in Lemma 3.1. An argument similar to the proof of Lemma A.1 shows $\ker(\mathcal{D}) = \{0\}$ [26, Lemma SM4.1]. Hence

$$\|(Z(y), y) - (Z(y_\star), y_\star)\| \leq \frac{1}{\sigma_{\min}(\mathcal{D})} \|\mathcal{D}(Z(y) - Z(y_\star), y - y_\star)\|.$$

By utilizing Lemma 4.3 with $X = Z(y)$ and $Z = X_\star$ and noting $\epsilon = \mathbf{tr}(Z(y)X_\star) = \epsilon_d(y) = b^\top y_\star - b^\top y$ (from strong duality), we see that

$$\left\|Z(y) - (U^\star)(U^\star)^\top Z(y)(U^\star)(U^\star)^\top\right\|_{\mathrm{F}} \leq \frac{\epsilon}{\lambda_{\min>0}(X_\star)} + \sqrt{\frac{2\epsilon}{\lambda_{\min>0}(X_\star)}\, \|Z(y)\|_{\mathrm{op}}}.$$

We also have $\mathcal{D}(Z(y) - Z(y_\star), y - y_\star) = (Z(y) - (U^\star)(U^\star)^\top Z(y)(U^\star)(U^\star)^\top, 0)$. Combining this equality and the above pieces, we get the results in Lemma 3.1. ∎

Appendix B. Lemmas from section 4. We first prove Lemma 4.3 concerning the distance to subspaces and the inner product.

Proof of Lemma 4.3. Complete V to form a basis $W = [V; U]$ for $\mathbf{R}^n$, where $U = [v_1, \ldots, v_{n-r+1}] \in \mathbf{R}^{n\times(n-r)}$ and where v_i is the eigenvector of Z associated with the ith largest eigenvalue.

Rotating into the coordinate system formed by $W = [V; U]$, let us compare X and its projection into the space spanned by V, $P_V(X) := VV^\top XVV^\top$,

$$W^\top XW = \begin{bmatrix} U^\top XU & U^\top XV \\ V^\top XU & V^\top XV \end{bmatrix}, \quad \text{and} \quad W^\top P_V(X)W = \begin{bmatrix} 0 & 0 \\ 0 & V^\top XV \end{bmatrix}.$$

Let $X_1 = U^\top XU$, $B = U^\top XV$, and $X_2 = V^\top XV$. Using the unitary invariance of $\|\cdot\|_{\mathrm{F}}$, we have $P_{V^\perp}(X) := X - VV^\top XVV^\top$ satisfying

$$\begin{aligned} \left\|X - VV^\top XVV^\top\right\|_{\mathrm{F}} &= \|P_{V^\perp}(X)\|_{\mathrm{F}} = \left\|W^\top XW - W^\top VV^\top XVV^\top W\right\|_{\mathrm{F}} \\ &= \left\|\begin{bmatrix} X_1 & B \\ B & 0 \end{bmatrix}\right\|_{\mathrm{F}}. \end{aligned} \tag{B.1}$$

A similar equality holds for $\|\cdot\|_*$. Thus we need only bound the terms X_1 and B.

Let us first get a bound on X_1. Since all the vectors in U have corresponding eigenvalues at least as large as the threshold $T = \lambda_{n-r}(Z) > 0$, and $Z \succeq 0$ by assumption, we have

(B.2)
$$\epsilon \overset{(a)}{=} \sum_{i=1}^{n} \lambda_i(Z) v_i^\top X v_i \geq T \sum_{i=1}^{n-r+1} v_i^\top X v_i = \mathbf{tr}(UXU^\top) = \mathbf{tr}(X_1) \overset{(b)}{=} \|X_1\|_* \geq \|X_1\|_{\mathrm{F}},$$

where we use $\epsilon = \mathbf{tr}(ZX)$ in step (a) and $X_1 \succeq 0$ in step (b). Next, we show a bound on B:

$$\left\|BB^\top\right\|_* \overset{(a)}{\leq} \|X_2\|_{\mathrm{op}} \mathbf{tr}(X_1) \overset{(b)}{\leq} \frac{\epsilon}{T} \|X_2\|_{\mathrm{op}} \overset{(c)}{\leq} \frac{\epsilon}{T} \|X\|_{\mathrm{op}}. \tag{B.3}$$

Here in step (a), we use Lemma B.1 for $W^\top XW = \left[\begin{smallmatrix} X_1 & B \\ B^\top & X_2 \end{smallmatrix}\right]$. In step (b), we use the bound $\frac{\epsilon}{T} \geq \|X_1\|_*$ proved in (B.2). In step (c), we use $\|X_2\|_{\mathrm{op}} \leq \|X\|_{\mathrm{op}}$. We also have that

$$\left\|\begin{bmatrix} 0 & B \\ B^\top & 0 \end{bmatrix}\right\|_{\mathrm{F}}^2 = \mathbf{tr}\left(\begin{bmatrix} BB^\top & 0 \\ 0 & B^\top B \end{bmatrix}\right) = 2\,\mathbf{tr}(BB^\top) = 2\left\|BB^\top\right\|_*. \tag{B.4}$$

Combining pieces, we bound the error in the Frobenius norm:

$$\left\|X - VV^\top XVV^\top\right\|_{\mathrm{F}} \overset{(a)}{\leq} \|X_1\|_{\mathrm{F}} + \left\|\begin{bmatrix} 0 & B \\ B^\top & 0 \end{bmatrix}\right\|_{\mathrm{F}} \overset{(b)}{\leq} \frac{\epsilon}{T} + \sqrt{\frac{2\epsilon}{T}\|X\|_{\mathrm{op}}}, \tag{B.5}$$

where step (a) uses (B.1) and the triangle inequality; step (b) uses the inequality $\frac{\epsilon}{T} \geq \|X_1\|_{\mathrm{F}}$ in (B.2) for bounding the term $\|X_1\|_{\mathrm{F}}$, and uses (B.3), and (B.4) for the other term. Similarly, we may bound the error in the nuclear norm:

$$\left\|X_\star - VV^\top X_\star VV^\top\right\|_* \overset{(a)}{\leq} \|X_1\|_* + \left\|\begin{bmatrix} 0 & B \\ B^\top & 0 \end{bmatrix}\right\|_* \overset{(b)}{\leq} \frac{\epsilon}{T} + 2\sqrt{\frac{r\epsilon}{T}\|X_\star\|_{\mathrm{op}}}.$$

Step (a) follows step (a) in (B.5). Step (b) first uses $\|\left[\begin{smallmatrix} 0 & B \\ B^\top & 0 \end{smallmatrix}\right]\| \leq \sqrt{2r}\|\left[\begin{smallmatrix} 0 & B \\ B^\top & 0 \end{smallmatrix}\right]\|$ as $\left[\begin{smallmatrix} 0 & B \\ B^\top & 0 \end{smallmatrix}\right]$ has rank at most $2r$, and then uses the same reasoning as step (b) in (B.5). ☐

Based on the Shur complement and Von Neumann's trace inequality, the following lemma is proved in [26, subsection SM4.1].

Lemma B.1. *Suppose* $Y = \left[\begin{smallmatrix} A & B \\ B^\top & D \end{smallmatrix}\right] \succeq 0$. *Then* $\|A\|_{\mathrm{op}} \mathbf{tr}(D) \geq \left\|BB^\top\right\|_*$.

Acknowledgments. Ahmet Alacaoglu suggested we consider the Chambolle–Pock method for solving (MinObjSDP) in numerical experiments. We would also like to thank the editor and reviewers for their feedback.

REFERENCES

[1] *The University of Florida Sparse Matrix Collection: Dimacs*10 *Group*, https://www.cise.ufl.edu/research/sparse/matrices/DIMACS10/index.html.

[2] *The University of Florida Sparse Matrix Collection: Gset Group*, https://www.cise.ufl.edu/research/sparse/matrices/Gset/index.html.

[3] F. Alizadeh, *Combinatorial Optimization with Interior Point Methods and Semi-definite Matrices*, Ph.D. thesis, University of Minnesota, 1991.

[4] F. Alizadeh, *Interior point methods in semidefinite programming with applications to combinatorial optimization*, SIAM J. Optim., 5 (1995), pp. 13–51.

[5] F. Alizadeh, J.-P. A. Haeberly, and M. L. Overton, *Complementarity and nondegeneracy in semidefinite programming*, Math. Program., 77 (1997), pp. 111–128.

[6] F. Alizadeh, J.-P. A. Haeberly, and M. L. Overton, *Primal-dual interior-point methods for semidefinite programming: Convergence rates, stability and numerical results*, SIAM J. Optim., 8 (1998), pp. 746–768.

[7] X. Bai, H. Wei, K. Fujisawa, and Y. Wang, *Semidefinite programming for optimal power flow problems*, Internat. J. Electrical Power Energy Systems, 30 (2008), pp. 383–392.

[8] A. I. Barvinok, *Problems of distance geometry and convex properties of quadratic maps*, Discrete Comput. Geom., 13 (1995), pp. 189–202.

[9] A. Beck and M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM J. Imaging Sci., 2 (2009), pp. 183–202.

[10] R. Bellman and K. Fan, *On systems of linear inequalities in Hermitian matrix variables*, Convexity, 7 (1963), pp. 1–11.

[11] S. J. Benson, Y. Ye, and X. Zhang, *Solving large-scale sparse semidefinite programs for combinatorial optimization*, SIAM J. Optim., 10 (2000), pp. 443–461.

[12] N. Boumal, V. Voroninski, and A. Bandeira, *The non-convex Burer-Monteiro approach works on smooth semidefinite programs*, in Advances in Neural Information Processing Systems, 2016, pp. 2757–2765.

[13] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*, Stud. Appl. Numer. Math. 15, SIAM, Philadelphia, 1994.

[14] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Found. Trends Machine Learning, 3 (2011), pp. 1–122.

[15] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, Cambridge, UK, 2004.

[16] S. Burer and R. D. Monteiro, *A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization*, Math. Program., 95 (2003), pp. 329–357.

[17] S. Burer and R. D. Monteiro, *Local minima and convergence in low-rank semidefinite programming*, Math. Program., 103 (2005), pp. 427–444.

[18] E. J. Candes, Y. C. Eldar, T. Strohmer, and V. Voroninski, *Phase retrieval via matrix completion*, SIAM Rev., 57 (2015), pp. 225–251.

[19] E. J. Candès and B. Recht, *Exact matrix completion via convex optimization*, Found. Comput. Math., 9 (2009), 717.

[20] A. Chai, M. Moscoso, and G. Papanicolaou, *Array imaging using intensity-only measurements*, Inverse Problems, 27 (2010), 015005.

[21] A. Chambolle and T. Pock, *A first-order primal-dual algorithm for convex problems with applications to imaging*, J. Math. Imaging Vis., 40 (2011), pp. 120–145.

[22] Y. Chen, Y. Chi, and A. J. Goldsmith, *Exact and stable covariance estimation from quadratic sampling via convex programming*, IEEE Trans. Inform. Theory, 61 (2015), pp. 4034–4059.

[23] K. L. Clarkson, *Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm*, ACM Trans. Algorithms, 6 (2010), 63.

[24] S. Diamond and S. Boyd, *Matrix-free convex optimization modeling*, in Optimization and Its Applications in Control and Data Sciences, Springer, New York, 2016, pp. 221–264.

[25] L. Ding and M. Udell, *On the Regularity and Conditioning of Low Rank Semidefinite Programs*, preprint, arXiv:2002.10673, 2020.

[26] L. Ding, A. Yurtsever, V. Cevher, J. A. Tropp, and M. Udell, *Supplementary Materials: An Optimal-Storage Approach to Semidefinite Programming Using Approximate Complementarity*, https://people.orie.cornell.edu/ld446/supplement.pdf.

[27] D. Drusvyatskiy and A. S. Lewis, *Error bounds, quadratic growth, and linear convergence of proximal methods*, Math. Oper. Res., 43 (2018), pp. 693–1050.

[28] M. Frank and P. Wolfe, *An algorithm for quadratic programming*, Naval Res. Logist. Quart., 3 (1956), pp. 95–110.

[29] M. P. Friedlander and I. Macedo, *Low-rank spectral optimization via gauge duality*, SIAM J. Sci. Comput., 38 (2016), pp. A1616–A1638.

[30] D. Gabay and B. Mercier, *A Dual Algorithm for the Solution of Non linear Variational Problems via Finite Element Approximation*, Institut de Recherche d'Informatique et d'Automatique, 1975.

[31] R. Glowinski and A. Marroco, *Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de dirichlet non linéaires*, Revue française d'automatique, informatique, recherche opérationnelle. Analyse numérique, 9 (1975), pp. 41–76.

[32] M. X. Goemans and D. P. Williamson, 879-*approximation algorithms for MAX CUT and MAX* 2*SAT*, in Proceedings of the 26th Annual ACM Symposium on Theory of Computing, ACM, 1994, pp. 422–431.

[33] M. X. Goemans and D. P. Williamson, *Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming*, J. ACM, 42 (1995), pp. 1115–1145.

[34] N. Halko, P. G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Rev., 53 (2011), pp. 217–288.

[35] E. Hazan, *Sparse Approximate Solutions to Semidefinite Programs*, in Latin American Symposium on Theoretical Informatics, Springer, New York, 2008, pp. 306–316.

[36] C. Helmberg and F. Rendl, *A spectral bundle method for semidefinite programming*, SIAM J. Optim., 10 (2000), pp. 673–696.

[37] S. Homer and M. Peinado, *Design and performance of parallel and distributed approximation algorithms for maxcut*, J. Parallel Distributed Computing, 46 (1997), pp. 48–61.

[38] M. Jaggi, *Revisiting Frank-Wolfe: Projection-free sparse convex optimization*, in Proceedings of the 30th International Conference on Machine Learning, 2013, pp. 427–435.

[39] D. Johnson, G. Pataki, and F. Alizadeh, *The Seventh DIMACs Implementation Challenge: Semidefinite and Related Optimization Problems*, DIMACS, Piscataway, NJ, 2000.

[40] P. R. Johnstone and P. Moulin, *Faster Subgradient Methods for Functions with Hölderian Growth*, preprint, arXiv:1704.00196, 2017.

[41] R. H. Keshavan, A. Montanari, and S. Oh, *Matrix completion from a few entries*, IEEE Trans. Inform. Theory, 56 (2010), pp. 2980–2998.

[42] J. Kuczyński and H. Woźniakowski, *Estimating the largest eigenvalue by the power and Lanczos algorithms with a random start*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 1094–1122.

[43] A. Lemon, A. M.-C. So, and Y. Ye, *Low-rank semidefinite programming: Theory and applications*, Found. Trends Optim., 2 (2016), pp. 1–156.

[44] E. S. Levitin and B. T. Polyak, *Constrained minimization methods*, Zh. Vychisl. Mat. Mat. Fiz., 6 (1966), pp. 787–823.

[45] K. Y. Levy, A. Yurtsever, and V. Cevher, *Online Adaptive Methods, Universality and Acceleration*, preprint, arXiv:1809.02864, 2018.

[46] R. Madani, S. Sojoudi, and J. Lavaei, *Convex relaxation for optimal power flow problem: Mesh networks*, IEEE Trans. Power Systems, 30 (2015), pp. 199–211.

[47] J. Malick, J. Povh, F. Rendl, and A. Wiegele, *Regularization methods for semidefinite programming*, SIAM J. Optim., 20 (2009), pp. 336–356.

[48] C. Mathieu and W. Schudy, *Correlation clustering with noisy input*, in Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, Philadelphia, 2010, pp. 712–728.

[49] A. Mosek, *The Mosek Optimization Software*, http://www.mosek.com, 2010.

[50] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, Appl. Optim. 87, Springer, New York, 2013.

[51] Y. Nesterov and A. Nemirovski, *Self-Concordant Functions and Polynomial Time Methods in Convex Programming*, USSR Academy of Sciences, Central Economic & Mathematical Institute, Moscow, 1989.

[52] Y. Nesterov and A. Nemirovskii, *Interior-Point Polynomial Algorithms in Convex Programming*, Stud. Appl. Numer. Math. 13, SIAM, Philadelphia, 1994.

[53] B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd, *Conic optimization via operator splitting and homogeneous self-dual embedding*, J. Optim. Theory Appl., 169 (2016), pp. 1042–1068.

[54] B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd, *SCS: Splitting Conic Solver, Version* 2.0.2, https://github.com/cvxgrp/scs, 2017.

[55] G. Pataki, *On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues*, Math. Oper. Res., 23 (1998), pp. 339–358.

[56] N. Rao, P. Shah, and S. Wright, *Conditional gradient with enhancement and truncation for atomic-norm regularization*, in NIPS Workshop on Greedy Algorithms, 2013.

[57] B. Recht, M. Fazel, and P. A. Parrilo, *Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization*, SIAM Rev., 52 (2010), pp. 471–501.

[58] J. Renegar, *Efficient First-Order Methods for Linear Programming and Semidefinite Programming*, preprint, arXiv:1409.5832, 2014.

[59] R. T. Rockafellar, *Monotone operators and the proximal point algorithm*, SIAM J. Control Optim., 14 (1976), pp. 877–898.

[60] A. P. Ruszczyński and A. Ruszczynski, *Nonlinear Optimization*, Princeton University Press, Princeton, NJ, 2006.

[61] N. Srebro and A. Shraibman, *Rank, Trace-norm and Max-norm*, in International Conference on Computational Learning Theory, Springer, New York, 2005, pp. 545–560.

[62] J. F. Sturm, *Using SeDuMi* 1.02*, a MATLAB toolbox for optimization over symmetric cones*, Optim. Methods Softw., 11 (1999), pp. 625–653.

[63] J. F. Sturm, *Error bounds for linear matrix inequalities*, SIAM J. Optim., 10 (2000), pp. 1228–1248.

[64] D. Sun, K.-C. Toh, Y. Yuan, and X.-Y. Zhao, *SDPNAL+: A MATLAB software for semidefinite programming with bound constraints (version* 1.0*)*, Optim. Methods Softw., 35 (2020), pp. 87–115.

[65] M. J. Todd, *Semidefinite optimization*, Acta Numer., 10 (2001), pp. 515–560.

[66] K.-C. Toh, M. J. Todd, and R. H. Tütüncü, *SDPT*3*–a MATLAB software package for semidefinite programming, version* 1.3, Optim. Methods Softw., 11 (1999), pp. 545–581.

[67] L. Vandenberghe and S. Boyd, *Semidefinite programming*, SIAM Rev., 38 (1996), pp. 49–95.

[68] I. Waldspurger, A. d'Aspremont, and S. Mallat, *Phase recovery, maxcut and complex semidefinite programming*, Math. Program., 149 (2015), pp. 47–81.

[69] I. Waldspurger and A. Waters, *Rank Optimality for the Burer-Monteiro Factorization*, preprint, arXiv:1812.03046, 2018.

[70] Z. Wen, D. Goldfarb, and W. Yin, *Alternating direction augmented lagrangian methods for semidefinite programming*, Math. Program. Comput., 2 (2010), pp. 203–230.

[71] Y. Yu, T. Wang, and R. J. Samworth, *A useful variant of the Davis–Kahan theorem for statisticians*, Biometrika, 102 (2014), pp. 315–323.

[72] A. Yurtsever, O. Fercoq, and V. Cevher, *A conditional gradient-based augmented lagrangian framework*, in International Conference on Machine Learning, 2019, pp. 7272–7281.

[73] A. Yurtsever, O. Fercoq, F. Locatello, and V. Cevher, *A conditional gradient framework for composite convex minimization with applications to semidefinite programming*, in International Conference on Machine Learning, 2018, pp. 5727–5736.

[74] A. Yurtsever, Y.-P. Hsieh, and V. Cevher, *Scalable convex methods for phase retrieval*, in Proceedings of the 6th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing, IEEE, 2015, pp. 381–384.

[75] A. Yurtsever, J. Tropp, O. Fercoq, M. Udell, and V. Cevher, *Scalable Semidefinite Programming*, preprint, arXiv:1912.02949, 2019.

[76] A. Yurtsever, M. Udell, J. Tropp, and V. Cevher, *Sketchy decisions: Convex low-rank matrix optimization with optimal storage*, in International Conference on Artificial Intelligence and Statistics, 2017, pp. 1188–1196.

[77] X.-Y. Zhao, D. Sun, and K.-C. Toh, *A Newton-CG augmented lagrangian method for semidefinite programming*, SIAM J. Optim., 20 (2010), pp. 1737–1765.