Math. Prog. Comp. (2010) 2:203–230
DOI 10.1007/s12532-010-0017-1

FULL LENGTH PAPER

Alternating direction augmented Lagrangian methods for semidefinite programming

Zaiwen Wen · Donald Goldfarb · Wotao Yin

Received: 21 September 2009 / Accepted: 11 September 2010 / Published online: 28 September 2010

Abstract We present an alternating direction dual augmented Lagrangian method for solving semidefinite programming (SDP) problems in standard form. At each iteration, our basic algorithm minimizes the augmented Lagrangian function for the dual SDP problem sequentially, first with respect to the dual variables corresponding to the linear constraints, and then with respect to the dual slack variables, while in each minimization keeping the other variables fixed, and then finally it updates the Lagrange multipliers (i.e., primal variables). Convergence is proved by using a fixed-point argument. For SDPs with inequality constraints and positivity constraints, our algorithm is extended to separately minimize the dual augmented Lagrangian function over four sets of variables. Numerical results for frequency assignment, maximum stable set and binary integer quadratic programming problems demonstrate that our algorithms

The work of Z. Wen and D. Goldfarb was supported in part by NSF Grant DMS 06-06712, ONR Grant N00014-08-1-1118 and DOE Grant DE-FG02-08ER58562. The work of W. Yin was supported in part by NSF CAREER Award DMS-07-48839, ONR Grant N00014-08-1-1101, the US Army Research Laboratory and the US Army Research Office grant W911NF-09-1-0383, and an Alfred P. Sloan Research Fellowship.

Z. Wen (✉) · D. Goldfarb
Department of Industrial Engineering and Operations Research,
Columbia University, New York, NY 10027, USA
e-mail: zw2109@columbia.edu

D. Goldfarb
e-mail: goldfarb@columbia.edu

W. Yin
Department of Computational and Applied Mathematics,
Rice University, Houston, TX 77005, USA
e-mail: wotao.yin@rice.edu

are robust and very efficient due to their ability or exploit special structures, such as sparsity and constraint orthogonality in these problems.

Keywords Semidefinite programming · Alternating direction method · Augmented Lagrangian method

Mathematics Subject Classification (2000) 90C06 · 90C22 · 90C30 · 90C35

1 Introduction

In this paper we present an alternating direction dual augmented Lagrangian method for solving semidefinite programming (SDP) problems. These convex optimization problems are solvable in polynomial time by interior point methods [26,29,33]. However, if the number of constraints m in an SDP is of order $O(n^2)$ when the unknown positive semidefinite matrix is $n \times n$, interior point methods become impractical both in terms of the time ($O(n^6)$) and the amount of memory ($O(m^2)$) required at each iteration to form the $m \times m$ positive definite Schur complement matrix M and compute the search direction by finding the Cholesky factorization of M. On the other hand, first-order methods do not form or factorize these large dense matrices; consequently, they typically require much less computation and storage at each iteration. Furthermore, some first-order methods are able to take advantage of problem structure such as sparsity. Hence they are often more suitable, and sometimes the only practical choice for solving large-scale SDPs.

Most existing first-order methods for SDP are based on the augmented Lagrangian method (which is also referred to as the method of multipliers). Specific methods differ in how the positive semidefinite constraints are handled. In [3,4], the positive definite variable X is replaced by $RR^\top$ in the primal augmented Lagrangian function, where R is a low rank matrix, and then nonlinear programming approaches are used. In [2,6], a coordinate descent method and an eigenvalue decomposition are used to minimize the primal augmented Lagrangian function. In [31], by fixing any $(n-1)$-dimensional principal submatrix of X and using its Schur complement, the positive semidefinite constraint is reduced to a simple second-order cone constraint and then a sequence of second-order cone programming problems constructed from the primal augmented Lagrangian function are minimized. In [41], the positive semidefinite constraint is represented implicitly by using a projection operator and a semismooth Newton approach combined with the conjugate gradient method is proposed to minimize the dual augmented Lagrangian function. The regularization methods [22] (and the related boundary point method [24]) are also based on a dual augmented Lagrangian approach and they use an eigenvalue decomposition to maintain complementarity. In fact, one variant of these methods is the same as our basic alternating direction method.

Alternating direction methods have been extensively studied to minimize the augmented Lagrangian function for optimization problems arising from partial differential equations (PDEs) [11,12]. In these methods, the variables are partitioned into several blocks according to their roles, and then the augmented Lagrangian function is minimized with respect to each block by fixing all other blocks at each inner iteration.

This simple idea has been applied to many other problem classes, such as, variational inequality problems [16,17,37], linear programming [9], nonlinear convex optimization [1,7,13,14,20,21,28], maximal monotone operators [10] and nonsmooth ℓ_1 minimization arising from compressive sensing [30,35,36,40]. In [38], an alternating direction method for SDP is presented by reformulating the complementary condition as a projection equation.

Our algorithm applies the alternating direction method within a dual augmented Lagrangian framework. When our method is applied to an SDP in standard form, at each iteration it first minimizes the dual augmented Lagrangian function with respect to the dual variables corresponding to the linear constraints, and then with respect to the dual slack variables while keeping the other variables fixed, after which it updates the primal variables. This algorithm is very closely related to the regularization method in [22]. While the theoretical algorithm in [22] updates the primal variable X only after a certain condition is satisfied, the actual algorithm implemented in [22] (i.e., Algorithm 5.1 in [22]) updates the primal variable X immediately after all other blocks are updated at each iteration, which is exactly our alternating direction method. The algorithms in [22] cannot be applied directly to SDPs with inequality constraints, and in particular, with positivity constraints, i.e., every component of X is nonnegative. In order to preserve the structure of these inequality constraints, like sparsity and orthogonality, we do not transform them to equality constraints but add extra steps to our alternating direction method to minimize the dual augmented Lagrangian function with respect to the dual variables corresponding to the inequality constraints. Numerical experiments on, for example, binary integer quadratic programming problems, show that the performance of our method can be significantly better than those reported in [22,41] if the special structures in these problems are exploited.

Our contributions are as follows. Although the techniques for analyzing the alternating direction methods for variational inequalities in [17] and a class of nonlinear optimization problems in [1] can be applied to analyze our algorithm for SDPs in standard form, we present a different and simple convergence proof by formulating our algorithm as a fixed point method. We note that the convergence properties of the actual implementation of the regularization method in [22] have not been studied. We present a multiple block variable extension (whose convergence has not been established) for solving SDPs with inequality constraints directly without transforming them into SDPs in standard form by introducing auxiliary variables and constraints, thus enabling exploitation of certain constraint structures.

The rest of this paper is organized as follows. We present an alternating direction augmented Lagrangian method for SDP in standard form in Subsect. 2.1 and analyze its convergence in Subsect. 2.2, and then extend this method to an expanded problem with inequality and positivity constraints in Subsect. 2.3. We discuss practical issues related to the eigenvalue decomposition performed at each iteration, strategies for adjusting the penalty parameter, the use of a step size for updating the primal variable X, termination rules and how to detect stagnation to enhance the performance of our methods in Sect. 3. Finally, numerical results for frequency assignment, maximum stable set and binary integer quadratic programming problems are presented in Sect. 4 to demonstrate the robustness and efficiency of our algorithm.

1.1 Preliminaries

The set of $n \times n$ symmetric matrices is denoted by S^n and the set of $n \times n$ symmetric positive semidefinite (positive definite) matrices is denoted by S^n_+ (S^n_{++}). The notation $X \succeq 0$ ($X \succ 0$) means that the matrix $X \in S^n$ is positive semidefinite (positive definite). The notation $X \geq 0$ ($X > 0$) means that every component of the matrix X is nonnegative (positive). The trace of X, i.e., the sum of the diagonal elements of X, is denoted by $\mathbf{Tr}(X)$. The inner product between two matrices $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{m\times n}$ is defined as $\langle A, B\rangle := \sum_{jk} A_{j,k}B_{j,k} = \mathbf{Tr}(A^\top B)$. The Frobenius norm of a matrix $A \in \mathbb{R}^{m\times n}$ is defined as $\|A\|_F := \sqrt{\sum_{i,j} A^2_{i,j}}$.

2 Alternating direction augmented Lagrangian methods

2.1 A two-splitting scheme for standard form SDPs

Consider the standard form SDP

$$\min_{X\in S^n} \quad \langle C, X\rangle \quad \text{s.t.} \quad \mathcal{A}(X) = b, \quad X \succeq 0, \tag{1}$$

where the linear map $\mathcal{A}(\cdot) : S^n \to \mathbb{R}^m$ is defined as

$$\mathcal{A}(X) := \left\langle A^{(1)}, X\right\rangle, \ldots, \left\langle A^{(m)}, X\right\rangle^\top, \tag{2}$$

the matrices $C, A^{(i)} \in S^n, i = 1, \ldots, m$, and the vector $b \in \mathbb{R}^m$ are given, and the unknown matrix $X \in S^n_+$. Let $\mathrm{vec}(X)$ be a vector that contains the columns of the matrix X, stacked each on top of the next in the order that they appear in the matrix, and $\mathrm{mat}(x)$ be a matrix X such that $x = \mathrm{vec}(X)$. Note that the equation $\mathcal{A}(X) = b$ is equivalent to $A\,\mathrm{vec}(X) = b$, where

$$A := \left(\mathrm{vec}(A^{(1)}), \ldots, \mathrm{vec}(A^{(m)})\right)^\top \in \mathbb{R}^{m\times n^2}.$$

The adjoint operator $\mathcal{A}^* : \mathbb{R}^m \to S^n$ of $\mathcal{A}$ is defined as $\mathcal{A}^*(y) := \sum_{i=1}^m y_i A^{(i)} = \mathrm{mat}(A^\top y)$. The operator $\mathcal{A}\mathcal{A}^* : \mathbb{R}^m \to \mathbb{R}^m$ is defined as $\mathcal{A}(\mathcal{A}^*(y)) = (AA^\top)y$ and the operator $\mathcal{A}^*\mathcal{A} : S^n \to S^n$ is defined as $\mathcal{A}^*(\mathcal{A}(X)) = \mathrm{mat}((A^\top A)\mathrm{vec}(X))$.

We make the following assumption throughout our presentation.

Assumption 1 *The matrix A has full row rank and the Slater condition holds for (1); that is, there exists a matrix $\bar{X} \succ 0$ satisfying $\mathcal{A}(\bar{X}) = b$.*

The dual problem of (1) is

$$\min_{y\in\mathbb{R}^m, S\in S^n} \quad -b^\top y \quad \text{s.t.} \quad \mathcal{A}^*(y) + S = C, \quad S \succeq 0. \tag{3}$$

The augmented Lagrangian function for the dual SDP (3) corresponding to the linear constraints is defined as:

$$\mathcal{L}_\mu(X, y, S) := -b^\top y + \langle X, \mathcal{A}^*(y) + S - C\rangle + \frac{1}{2\mu}\|\mathcal{A}^*(y) + S - C\|_F^2,$$

where $X \in S^n$ and $\mu > 0$. Starting from $X^0 = \mathbf{0}$, the augmented Lagrangian method solves on the k-th iteration

$$\min_{y\in\mathbb{R}^m, S\in S^n} \mathcal{L}_\mu(X^k, y, S), \quad \text{s.t.} \quad S \succeq 0, \tag{4}$$

for y^{k+1} and S^{k+1}, and then updates the primal variable X^{k+1} by

$$X^{k+1} := X^k + \frac{\mathcal{A}^*(y^{k+1}) + S^{k+1} - C}{\mu}. \tag{5}$$

Since solving problem (4) requires jointly minimizing $\mathcal{L}_\mu(X^k, y, S)$ with respect to y and S, it can be very time consuming. Instead, we minimize the augmented Lagrangian function with respect to y and S, one after the other, and hence no longer solving problem (4) exactly. Specifically, we replace (4) and (5) by the following:

$$y^{k+1} := \arg\min_{y\in\mathbb{R}^m} \mathcal{L}_\mu(X^k, y, S^k), \tag{6a}$$

$$S^{k+1} := \arg\min_{S\in S^n} \mathcal{L}_\mu(X^k, y^{k+1}, S), \quad S \succeq 0, \tag{6b}$$

$$X^{k+1} := X^k + \frac{\mathcal{A}^*(y^{k+1}) + S^{k+1} - C}{\mu}. \tag{6c}$$

The above order of (6a)–(6c) is not important. Similar convergence properties and numerical performance, which we will study below for (6a)–(6c), apply to any other order as well.

The first-order optimality conditions for (6a) are

$$\nabla_y \mathcal{L}_\mu(X^k, y^{k+1}, S^k) := \mathcal{A}(X^k) - b + \frac{1}{\mu}\mathcal{A}(\mathcal{A}^*(y^{k+1}) + S^k - C) = 0.$$

Since by Assumption 1 $\mathcal{A}\mathcal{A}^*$ is invertible , we obtain $y^{k+1} := y(S^k, X^k)$, where

$$y(S, X) := -(\mathcal{A}\mathcal{A}^*)^{-1}\left(\mu(\mathcal{A}(X) - b) + \mathcal{A}(S - C)\right). \tag{7}$$

By rearranging the terms of $\mathcal{L}_\mu(X^k, y^{k+1}, S)$, it is easily verified that problem (6b) is equivalent to

$$\min_{S\in S^n} \left\|S - V^{k+1}\right\|_F^2, \quad S \succeq 0, \tag{8}$$

where $V^{k+1} := V(S^k, X^k)$ and the function $V(S, X)$ is defined as

$$V(S, X) := C - \mathcal{A}^*(y(S, X)) - \mu X. \tag{9}$$

Hence, we obtain the solution $S^{k+1} := V_\dagger^{k+1} := Q_\dagger \Sigma_+ Q_\dagger^\top$, where

$$Q \Sigma Q^\top = (Q_\dagger \; Q_\ddagger) \begin{pmatrix} \Sigma_+ & \mathbf{0} \\ \mathbf{0} & \Sigma_- \end{pmatrix} \begin{pmatrix} Q_\dagger^\top \\ Q_\ddagger^\top \end{pmatrix}$$

is the spectral decomposition of the matrix V^{k+1}, and Σ_+ and Σ_- are the nonnegative and negative eigenvalues of V^{k+1}. It follows from the updating Eq. (6c) that

$$X^{k+1} := X^k + \frac{\mathcal{A}^*(y^{k+1}) + S^{k+1} - C}{\mu} = \frac{1}{\mu}(S^{k+1} - V^{k+1}) = \frac{1}{\mu} V_\ddagger^{k+1}, \tag{10}$$

where $V_\ddagger^{k+1} := -Q_\ddagger \Sigma_- Q_\ddagger^\top$. Note that X^{k+1} is also the optimal solution of

$$\min_{X \in S^n} \quad \left\| \mu X + V^{k+1} \right\|_F^2, \quad X \succeq 0. \tag{11}$$

From the above observation, we arrive at the alternating direction augmented Lagrangian method in Algorithm 1, below.

Algorithm 1: Alternating direction augmented Lagrangian method for SDP

Set $X^0 \succeq 0$ and $S^0 \succeq 0$.;
for $k = 0, 1, \cdots$ **do**
  Compute y^{k+1} according to (7).
  Compute V^{k+1} and its eigenvalue decomposition, and set $S^{k+1} := V_\dagger^{k+1}$.
  Compute $X^{k+1} = \frac{1}{\mu}(S^{k+1} - V^{k+1})$.

Remark 1 Algorithm 1 is related to the boundary point method in [24] and the regularization method in [22], in which X^k is fixed until $\frac{1}{\mu}(S^{k+1} - V^{k+1})$ is nearly feasible. However, the actual regularization method implemented in the numerical experiments in [22] is exactly Algorithm 1.

Remark 2 If $\mathcal{A}\mathcal{A}^* = I$, as is the case for many SDP relaxations of combinatorial optimization problems, such as the maxcut SDP relaxation, the SDP relaxation of the maximum stable set problem and the bisection SDP relaxation, step (6a), i.e., (7), is very inexpensive. If $\mathcal{A}\mathcal{A}^* \neq I$, we can compute $\mathcal{A}\mathcal{A}^*$ and its inverse (or its Cholesky factorization) prior to executing Algorithm 1. If computing the Cholesky factorization of $\mathcal{A}\mathcal{A}^*$ is very expensive, an iterative method for solving the system of linear equations corresponding to (7) can be used and the strategies described in [7,10] for

controlling the accuracy of the subproblem may provide a provable convergence guarantee. Moreover, as in the approaches used in [34,35], it might be possible to replace the step (6a) by a single steepest descent gradient step:

$$y^{k+1} := \arg\min_{y\in\mathbb{R}^m} (g^k)^\top (y - y^k) + \frac{1}{\tau^k}\|y - y^k\|^2, \tag{12}$$

where $\tau^k = \frac{\mu\|g^k\|_2^2}{\|\mathcal{A}^*(g^k)\|_F^2}$ and $g^k := \nabla_y \mathcal{L}_\mu(y^k, S^k, X^k)$; that is, $y^{k+1} := y^k - \tau^k g^k$.

2.2 Convergence analysis of algorithm 1

The convergence results obtained for the alternating direction methods for optimization problems arising from PDEs in [12], variational inequalities in [17] and a class of nonlinear optimization problems in [1] can be applied to our algorithm. However, here we present a simpler direct proof by formulating our algorithm as a fixed point method. Some of our results shed light on rules for monitoring iteration progress and choosing stopping criteria.

For any matrix $V \in S^n$, let the matrix $V_\dagger$, $V_\ddagger$ be denoted by $\mathcal{P}(V)$. Hence, each iteration of Algorithm 1 can be expressed as

$$y^{k+1} := y(S^k, X^k) \text{ and } \left(S^{k+1}, \mu X^{k+1}\right) := \mathcal{P}(V^{k+1}) = \mathcal{P}(V(S^k, X^k)). \tag{13}$$

We first describe a result of the orthogonal decomposition.

Lemma 1 (Theorem 3.2.5 in [18], J.-J. Moreau) *Let K be a closed convex cone and $K^\diamond$ be the polar cone of K, that is, $K^\diamond := \{s \in \mathbb{R}^n : \langle s, x\rangle \le 0 \text{ for all } x \in K\}$. For the three elements x, x_1 and x_2 in $\mathbb{R}^n$, the properties below are equivalent:*

(i) $x = x_1 + x_2$ *with* $x_1 \in K$ *and* $x_2 \in K^\diamond$ *and* $\langle x_1, x_2\rangle = 0$;
(ii) $x_1 = P_K(x)$ *and* $x_2 = P_{K^\diamond}(x)$,

where $P_K(x)$ is the projection of x on K.

The next lemma shows that any optimal solution of (1) is a fixed point of the Eq. (13).

Lemma 2 *Suppose Assumption 1 holds. Then, there exists primal and dual optimal solutions (X, y, S) for (1) and (3) and the following two statements are equivalent:*

(i) (X, y, S) *satisfies the KKT optimality conditions*

$$\mathcal{A}(X) = b, \quad \mathcal{A}^*(y) + S = C, \quad SX = 0, \quad X \succeq 0, \quad Z \succeq 0.$$

(ii) (X, y, S) *satisfies*

$$y = y(S, X) \quad \textit{and} \quad (S, \mu X) = \mathcal{P}(V(S, X)).$$

Proof The proof here is similar to Proposition 2.6 in [22]. Since the Slater condition holds, there exists primal and dual optimal solution (X, y, S) for (1) and (3) so that statement (i) is true. Direct algebraic manipulation shows that $y = y(S, X)$ from statement (i). It follows from Lemma 1 that

$$S - \mu X = V(S, X), \quad X \succeq 0, \quad S \succeq 0, \quad SX = 0,$$

is equivalent to $S = V_\dagger(S, X)$ and $\mu X = V_\ddagger(S, X)$. Hence, statement (i) implies statement (ii). Now, suppose statement (ii) is true; i.e., $S = V_\dagger(S, X)$ and $\mu X = V_\ddagger(S, X)$. Since $S - \mu X = V(S, X)$, it follows from $V(S, X) = C - \mathcal{A}^*(y) - \mu X$ that $S = C - \mathcal{A}^*(y)$. From $y = y(S, X)$, we obtain

$$(\mathcal{A}\mathcal{A}^*)y = \mu(b - \mathcal{A}(X)) + \mathcal{A}(C - S) = \mu(b - \mathcal{A}(X)) + (\mathcal{A}\mathcal{A}^*)y,$$

which implies that $\mathcal{A}(X) = b$. Hence, statement (ii) implies statement (i). □

We now show that the operator $\mathcal{P}(V)$ is nonexpansive.

Lemma 3 *For any $V, \widehat{V} \in S^n$,*

$$\left\|\mathcal{P}(V) - \mathcal{P}(\widehat{V})\right\|_F \leq \|V - \widehat{V}\|_F, \tag{14}$$

with equality holding if and only if $V_\dagger^\top \widehat{V}_\ddagger = 0$ and $V_\ddagger^\top \widehat{V}_\dagger = 0$.

Proof We denote $V_\dagger - \widehat{V}_\dagger$ by $W_\dagger$ and $V_\ddagger - \widehat{V}_\ddagger$ by $W_\ddagger$. Since $V_\dagger^\top V_\ddagger = 0$ and $\widehat{V}_\dagger^\top \widehat{V}_\ddagger = 0$, we obtain $-W_\dagger^\top W_\ddagger = V_\dagger^\top \widehat{V}_\ddagger + \widehat{V}_\dagger^\top V_\ddagger$. The positive semidefinitess of the matrices $V_\dagger, V_\ddagger, \widehat{V}_\dagger, \widehat{V}_\ddagger \succeq 0$ implies that $\mathbf{Tr}(V_\dagger^\top \widehat{V}_\ddagger) \geq 0$ and $\mathbf{Tr}(\widehat{V}_\dagger^\top V_\ddagger) \geq 0$. Expanding terms of $V - \widehat{V}$, we obtain

$$\begin{aligned}
\|V - \widehat{V}\|_F^2 &= \mathbf{Tr}\left((W_\dagger - W_\ddagger)^\top (W_\dagger - W_\ddagger)\right) = \mathbf{Tr}\left(W_\dagger^\top W_\dagger + W_\ddagger^\top W_\ddagger\right) \\
&\quad -2\,\mathbf{Tr}\left(W_\dagger^\top W_\ddagger\right) \\
&= \left\|\mathcal{P}(V) - \mathcal{P}(\widehat{V})\right\|_F^2 + 2\,\mathbf{Tr}\left(V_\dagger^\top \widehat{V}_\ddagger + \widehat{V}_\dagger^\top V_\ddagger\right) \\
&\geq \left\|\mathcal{P}(V) - \mathcal{P}(\widehat{V})\right\|_F^2,
\end{aligned}$$

which proves (14). □

The following lemma shows that the operator $V(S, X)$ is nonexpansive.

Lemma 4 *For any $S, X, \widehat{S}, \widehat{X} \in S_+^n$,*

$$\|V(S, X) - V(\widehat{S}, \widehat{X})\|_F \leq \left\|\left(S - \widehat{S}, \mu(X - \widehat{X})\right)\right\|_F \tag{15}$$

with equality holding if and only if $V - \widehat{V} = (S - \widehat{S}) - \mu(X - \widehat{X})$.

Proof From the definition of $y(S, X)$ in (7), we have

$$y(\widehat{S}, \widehat{X}) - y(S, X) = (\mathcal{A}\mathcal{A}^*)^{-1}\left(\mu\mathcal{A}(X - \widehat{X}) + \mathcal{A}(S - \widehat{S})\right),$$

which together with (9) gives

$$\begin{aligned} V(S, X) - V(\widehat{S}, \widehat{X}) &= (C - \mathcal{A}^*(y(S, X)) - \mu X) - (C - \mathcal{A}^*(y(\widehat{S}, \widehat{X})) - \mu\widehat{X}) \\ &= \mathcal{A}^*(y(\widehat{S}, \widehat{X}) - y(S, X)) + \mu(\widehat{X} - X) \\ &= \mathrm{mat}\left(-\mu\,(I - M)\,\mathrm{vec}(X - \widehat{X}) + M\mathrm{vec}(S - \widehat{S})\right), \end{aligned} \tag{16}$$

where $M := A^\top(AA^\top)^{-1}A$. Since M is an orthogonal projection matrix whose spectral radius is 1, we obtain from (16) that

$$\begin{aligned} \left\|V(S, X) - V(\widehat{S}, \widehat{X})\right\|_F^2 &= \left\|\mu\,(I - M)\,\mathrm{vec}(X - \widehat{X}) - M\mathrm{vec}(S - \widehat{S})\right\|_2^2 \\ &\le \|\mu\mathrm{vec}(X - \widehat{X})\|_2^2 + \|\mathrm{vec}(S - \widehat{S})\|_2^2 \\ &= \left\|S - \widehat{S}, \mu(X - \widehat{X})\right\|_F^2, \end{aligned} \tag{17}$$

which proves (15).

If the equality in (15) holds, it also holds in (17); that is,

$$\begin{aligned} \|\mu(I - M)\mathrm{vec}(X - \widehat{X})\|_2^2 + \|M\mathrm{vec}(S - \widehat{S})\|_2^2 = \|\mu\mathrm{vec}(X - \widehat{X})\|_2^2 \\ +\|\mathrm{vec}(S - \widehat{S})\|_2^2. \end{aligned}$$

This implies that $M\mathrm{vec}(X - \widehat{X}) = 0$ and $(I - M)\mathrm{vec}(S - \widehat{S}) = 0$. Using this relations in (16), we obtain

$$V(S, X) - V(\widehat{S}, \widehat{X}) = \mathrm{mat}\left(\mathrm{vec}(S - \widehat{S}) - \mu\mathrm{vec}(X - \widehat{X})\right),$$

which proves the second statement. □

Lemma 5 *Let (X^*, y^*, S^*), where $y^* = y(S^*, X^*)$, be an optimal solution of (1) and (3). Under Assumption 1, if*

$$\left\|\mathcal{P}(V(S, X)) - \mathcal{P}(V(S^*, X^*))\right\|_F = \left\|S - S^*, \mu(X - X^*)\right\|_F, \tag{18}$$

then, $(S, \mu X)$ is a fixed point, that is, $(S, \mu X) = \mathcal{P}(V(S, X))$, and hence, (X, y, S), where $y = y(S, X)$, is a primal and dual optimal solutions of (1) and (3).

Proof From Lemma 2, we have $(S^*, \mu X^*) = \mathcal{P}(V(S^*, X^*))$. From Lemmas (3) and (4), we have

$$V(S, X) - V(S^*, X^*) = (S - S^*) - \mu(X - X^*),$$

which implies that $V(S, X) = S - \mu X$. Since S and X are all positive semidefinite, and $S^\top X = 0$, we obtain from Lemma 1 that $(S, \mu X) = \mathcal{P}(V(S, X))$. □

Given Lemmas 3, 4 and 5, we can prove convergence of Algorithm 1 by following the proof of Theorem 4.5 in [15].

Theorem 2 *The sequence* $\{(X^k, y^k, S^k)\}$ *generated by Algorithm 1 from any starting point* (X^0, y^0, S^0) *converges to a solution* $(X^*, y^*, S^*) \in \mathcal{X}^*$, *where* $\mathcal{X}^*$ *is the set of primal and dual solutions of* (1) *and* (3).

Proof Since both $\mathcal{P}(\cdot)$ and $V(\cdot,\cdot)$ are non-expansive, $\mathcal{P}(V(\cdot,\cdot))$ is also nonexpansive. Therefore, $\{(S^k, \mu X^k)\}$ lies in a compact set and must have a limit point, say $\bar{S} = \lim_{j\to\infty} S^{k_j}$ and $\bar{X} = \lim_{j\to\infty} X^{k_j}$. Also, for any $(X^*, y^*, S^*) \in \mathcal{X}^*$,

$$\begin{aligned}\left\|(S^{k+1}, \mu X^{k+1}) - (S^*, \mu X^*)\right\|_F &= \left\|\mathcal{P}(V(S^k, \mu X^k)) - \mathcal{P}(V(S^*, \mu X^*))\right\|_F \\ &\le \left\|V(S^k, \mu X^k) - V(S^*, \mu X^*)\right\|_F \\ &\le \left\|(S^k, \mu X^k) - (S^*, \mu X^*)\right\|_F,\end{aligned}$$

which means that the sequence $\{\|(S^k, \mu X^k) - (S^*, \mu X^*)\|_F\}$ is monotonically nonincreasing. Therefore,

$$\lim_{k\to\infty} \left\|(S^k, \mu X^k) - (S^*, \mu X^*)\right\|_F = \left\|(\bar{S}, \mu\bar{X}) - (S^*, \mu X^*)\right\|_F, \tag{19}$$

where $(\bar{S}, \mu\bar{X})$ can be any limit point of $\{(S^k, \mu X^k)\}$. By the continuity of $\mathcal{P}(V(\cdot,\cdot))$, the image of $(\bar{S}, \mu\bar{X})$,

$$\mathcal{P}(V(\bar{S}, \mu\bar{X})) = \lim_{j\to\infty} \mathcal{P}(V(S^{k_j}, \mu X^{k_j})) = \lim_{j\to\infty} (S^{k_j+1}, \mu X^{k_j+1}),$$

is also a limit of $\{(S^k, \mu X^k)\}$. Therefore, we have

$$\left\|\mathcal{P}(V(\bar{S}, \mu\bar{X})) - \mathcal{P}(V(S^*, \mu X^*))\right\|_F = \left\|(\bar{S}, \mu\bar{X}) - (S^*, \mu X^*)\right\|_F,$$

which allows us to apply Lemma 5 to get that $(\bar{S}, \bar{y}, \mu\bar{X})$, where $\bar{y} = y(\bar{S}, \bar{X})$, is an optimal solution to problems (1) and (3). Finally, by setting $(S^*, \mu X^*) = (\bar{S}, \mu\bar{X})$ in (19), we get that

$$\lim_{k\to\infty} \left\|(S^k, \mu\bar{X}^k) - (\bar{S}, \mu\bar{X})\right\|_F = \lim_{j\to\infty} \left\|(S^{k_j}, \mu\bar{X}^{k_j}) - (\bar{S}, \mu\bar{X})\right\|_F = 0,$$

i.e., $\{(S^k, \mu X^k)\}$ converges to its unique limit of $(\bar{S}, \mu\bar{X})$. □

We now describe the relationship between the primal infeasibility $\|\mathcal{A}(X^{k+1}) - b\|_2$ and dual infeasibility $\|C - \mathcal{A}^*(y^{k+1}) - S^{k+1}\|_F$ and the difference between the matrices $\{V^k\}$.

Corollary 1 *Let* $\{(X^k, y^k, S^k)\}$ *be a sequence generated by Algorithm 1. Then*

1. $\mathcal{A}(X^{k+1}) - b = \frac{1}{\mu}\mathcal{A}(S^{k+1} - S^k)$ *and* $\|\mathcal{A}(X^{k+1}) - b\|_2 \le \frac{\|A\|_2}{\mu}\|V^{k+1} - V^k\|_F$.
2. $C - \mathcal{A}^*(y^{k+1}) - S^{k+1} = \mu(X^k - X^{k+1})$ *and* $\|C - \mathcal{A}^*(y^{k+1}) - S^{k+1}\|_F \le \|V^{k+1} - V^k\|_F$.
3. $\|V^{k+1} - V^k\|_F \le \|V^k - V^{k-1}\|_F$.

Proof (1) It follows from (7), (9) and Assumption 1 that

$$\begin{aligned}\mathcal{A}(X^{k+1}) - b &= \frac{1}{\mu}\mathcal{A}(S^{k+1} - V^{k+1}) - b \\ &= \frac{1}{\mu}\mathcal{A}(S^{k+1} - C) + \mathcal{A}(X^k) - b + (b - \mathcal{A}(X^k)) + \frac{1}{\mu}\mathcal{A}(C - S^k) \\ &= \frac{1}{\mu}\mathcal{A}(S^{k+1} - S^k).\end{aligned}$$

Since the projection $V_\dagger^{k+1}$ is nonexpansive, we obtain

$$\|\mathcal{A}(X^{k+1}) - b\|_2 \le \frac{\|A\|_2}{\mu}\|\mathrm{vec}(S^{k+1} - S^k)\|_2 \le \frac{\|A\|_2}{\mu}\|V^{k+1} - V^k\|_F.$$

(2) Rearranging the terms of (6c), we obtain the first part of statement 2. The nonexpansiveness of the projection $V_\ddagger^{k+1}$ gives

$$\|C - \mathcal{A}^*(y^{k+1}) - S^{k+1}\|_2 = \|V_\ddagger^{k+1} - V_\ddagger^k\|_F \le \|V^{k+1} - V^k\|_F.$$

(3) From Lemmas 3 and 4, we obtain

$$\begin{aligned}\left\|V^{k+1} - V^k\|_F \le \|S^k - S^{k-1}, \mu(X^k - X^{k-1})\right\|_F \\ = \left\|V_\dagger^k - V_\dagger^{k-1}, V_\ddagger^k - V_\ddagger^{k-1}\right\|_F \le \|V^k - V^{k-1}\|_F.\end{aligned}$$

□

2.3 A multiple-splitting scheme for an expanded problem

If there are linear inequality constraints in an SDP, one can transform them to equality constraints by introducing non-negative slack variables and thus, apply Algorithm 1 without any change. However, in order to preserve favorable constraint structures such as sparsity and orthogonality, we shall avoid applying such a transformation. We now extend our alternating direction method (6a)–(6c) to solve an expanded SDP problem, that includes, in particular, positivity constraints on the elements of the matrix X; i.e.,

$$\min_{X \in S^n} \langle C, X\rangle, \quad \text{s.t. } \mathcal{A}(X) = b, \quad \mathcal{B}(X) \ge d, \quad X \succeq 0, \quad X \ge 0, \tag{20}$$

where $d \in \mathbb{R}^q$ and the linear map $\mathcal{B}(\cdot) : S^n \to \mathbb{R}^q$ is defined as

$$\mathcal{B}(X) := \left(\left\langle B^{(1)}, X\right\rangle, \ldots, \left\langle B^{(q)}, X\right\rangle\right)^\top, \quad B^{(i)} \in S^n, i = 1, \ldots, q. \tag{21}$$

As we did for the operator $\mathcal{A}$, we define the operators $\mathcal{B}^*$, $\mathcal{B}\mathcal{B}^*$ and $\mathcal{B}^*\mathcal{B}$ by introducing

$$B = \left(\mathrm{vec}(B^{(1)}), \ldots, \mathrm{vec}(B^{(q)})\right)^\top \in \mathbb{R}^{q \times n^2}.$$

We also make the following assumption.

Assumption 3 *The matrices A and B have full row rank and a refined Slater condition holds for (20); that is, there exists a positive definite matrix $\bar{X}$ satisfying $\mathcal{A}(\bar{X}) = b$, $\mathcal{B}(\bar{X}) \geq d$ and $\bar{X} \geq 0$.*

The dual of problem (20) is

$$\begin{aligned} \min_{y \in \mathbb{R}^m, v \in \mathbb{R}^q, S \in S^n, Z \in S^n} \quad & -b^\top y - d^\top v, \quad \text{s.t.} \quad \mathcal{A}^*(y) + \mathcal{B}^*(v) + S + Z = C, \\ & v \geq 0, \quad S \succeq 0, \quad Z \geq 0. \end{aligned} \tag{22}$$

The augmented Lagrangian function for the dual SDP (22) corresponding to the linear constraints is defined as:

$$\begin{aligned} \mathcal{L}_\mu(X, y, v, Z, S) := & -b^\top y - d^\top v + \left\langle X, \mathcal{A}^*(y) + \mathcal{B}^*(v) + S + Z - C\right\rangle \\ & + \frac{1}{2\mu}\|\mathcal{A}^*(y) + \mathcal{B}^*(v) + S + Z - C\|_F^2, \end{aligned} \tag{23}$$

where $X \in S^n$ and $\mu > 0$. Starting from $X^0 \succeq 0$, $v^0 \geq 0$, $Z^0 \geq 0$ and $S^0 \succeq 0$, our alternating direction method computes new iterates similar to the procedure (6a)–(6c) as follows

$$y^{k+1} := \arg\min_{y \in \mathbb{R}^m} \ \mathcal{L}_\mu(X^k, \ y, \ v^k, \ Z^k, \ S^k), \tag{24a}$$

$$v^{k+1} := \arg\min_{v \in \mathbb{R}^q} \ \mathcal{L}_\mu(X^k, \ y^{k+1}, \ v, \ Z^k, \ S^k), \quad v \geq 0, \tag{24b}$$

$$Z^{k+1} := \arg\min_{Z \in S^n} \ \mathcal{L}_\mu(X^k, \ y^{k+1}, \ v^{k+1}, \ Z, \ S^k), \quad Z \geq 0, \tag{24c}$$

$$S^{k+1} := \arg\min_{S \in S^n} \ \mathcal{L}_\mu(X^k, \ y^{k+1}, \ v^{k+1}, \ Z^{k+1}, \ S), \quad S \succeq 0, \tag{24d}$$

$$X^{k+1} := X^k + \frac{\mathcal{A}^*(y^{k+1}) + \mathcal{B}^*(v^{k+1}) + S^{k+1} + Z^{k+1} - C}{\mu}. \tag{24e}$$

Note that y^{k+1}, S^{k+1} and X^{k+1} can be computed in the same fashion as in (7), (8) and (10), respectively. By rearranging the terms of $\mathcal{L}_\mu(X^k, y^{k+1}, v, Z^k, S^k)$, it is easily

verified that problem (24b) is equivalent to the strictly convex quadratic program

$$\min_{v\in\mathbb{R}^n} \quad \left(\mathcal{B}\left(X^k+\frac{1}{\mu}Y^{k+1}\right)-d\right)^{\top} v+\frac{1}{2\mu}v^{\top}(\mathcal{B}\mathcal{B}^*)v, \quad v\geq 0, \tag{25}$$

where $Y^{k+1} := \mathcal{A}^*(y^{k+1}) + S^k + Z^k - C$. If $\mathcal{B}\mathcal{B}^*$ is the identity, the solution v^{k+1} is

$$v^{k+1} := \max\left(-\mu\left(\mathcal{B}\left(X^k+\frac{1}{\mu}Y^{k+1}\right)-d\right), \mathbf{0}\right);$$

Otherwise, the solution is not as straightforward, and one can choose a method based on the comprehensive summaries of software "http://plato.asu.edu/guide.html" and "http://www.numerical.rl.ac.uk/qp/qp.html" according to available computational resources. By rearranging the terms of $\mathcal{L}_\mu(X^k, y^{k+1}, v^{k+1}, Z, S^k)$, it is easily verified that problem (24c) is equivalent to

$$\min_{Z\in S^n} \quad \left\|Z-U^{k+1}\right\|_F^2, \quad Z\geq 0,$$

where $U^{k+1} := C-\mathcal{A}^*(y^{k+1})-\mathcal{B}^*(v^{k+1})-S^k-\mu X^k$. Hence, the solution of problem (24c) is $Z^{k+1} = U_+^{k+1}$, the positive part of U^{k+1}, that is, $(U_+^{k+1})_{i,j} := \max(U_{ij}^{k+1}, 0)$.

Our numerical experiments show that the convergence of the alternating direction method is not related to the order in which subproblems (24a)–(24e) are solved. The scheme (24) is a special case of an alternating direction augmented Lagrangian method (ADM) for three or more separable blocks of variables. Although this version of ADM has been applied to solving a wide range of applications successfully, its theoretical convergence property is still unknown.

3 Algorithmic issues

In this section, we discuss algorithmic issues related to the eigenvalue decomposition performed at each iteration, strategies for adjusting the penalty parameter, the use of a step size for updating the primal variable X, termination rules and how to detect stagnation to enhance the performance of our methods. Our discussion focuses on procedures (6a)–(6c) for problem (1), but it extends in an obvious way to (24a)–(24e) and the expanded problem (20).

3.1 Eigenvalue decomposition

One of the bottlenecks of our alternating direction method is the computation of the eigenvalue decomposition. Fortunately, for many problems in practice, either the primal solution X or the dual solution S is a low rank matrix. For example, the primal variable X in the maxcut SDP relaxation often has low rank while the dual variable S in frequency assignment problem often has low rank. Moreover, since the optimal solution pair (X, y, S) satisfies the complementary condition $XS = 0$, the matrices X

and S share the same set of eigenvectors and the positive eigenvalues of X (S) correspond to zero eigenvalues of S (X). Therefore, we only need to compute either $V_\dagger^k$, the part corresponding to the positive eigenvalues of V^k, or $V_\ddagger^k$, the part corresponding to the negative eigenvalues of V^k, at iteration k. Specifically, the following adaptive scheme can be used at the end of iteration $k-1$ to decide whether $V_\dagger^k$ or $V_\ddagger^k$ should be computed. Suppose that $V_\dagger^{k-1}$ has been computed and let $\kappa_+(V^{k-1})$ be the total number of the positive eigenvalues of V^{k-1}. If $\kappa_+(V^{k-1}) \le \frac{n}{2}$, this suggests that S^k might have low rank and we compute $V_\dagger^k$. Otherwise, it is possible that X^k has low rank and we compute $V_\ddagger^k$. If the total number of the negative eigenvalues $\kappa_-(V^{k-1})$ of V^{k-1} is known, a similar strategy can be used.

There are two types of methods, direct and iterative, for computing selected eigenvalues and eigenvectors of a real symmetric matrix V. Direct methods, which reduce V to tridiagonal form and then compute the required eigenvalues and eigenvectors from the tridiagonal matrix, are suitable for small and medium sized matrices. Since n is less than 5,000 in our numerical experiments, the code "DSYEVX" in LAPACK works fairly well. Iterative methods, like the Lanczos algorithm (for example, "ARPACK"), require only matrix-vector products and hence they are suitable for sparse matrices or matrices for which the required matrix-vector products are cheap. If the matrices C and $\mathcal{A}^*(y)$ are sparse or have low rank, then advantage can be taken of these structures since $V := C - \mathcal{A}^*(y) - \mu X$. Since iterative methods require $O(n^3)$ operations if $O(n)$ eigenvalues need to be computed, they are not competitive with direct methods in this case. Hence, iterative methods should be used only if either $V_\dagger^k$ or $V_\ddagger^k$ is expected to have low rank.

Note from Corollary 1 that the primal infeasibility $\|\mathcal{A}(X^{k+1})-b\|_2$ and dual infeasibility $\|C-\mathcal{A}^*(y^{k+1})-S^{k+1}\|_F$ are bounded by the difference $\|V^k - V^{k+1}\|_F$ which is nonincreasing. Hence, when the alternative direction method converges slowly, $\|V^k - V^{k+1}\|_F$ is often quite small. Hence, the spectral decomposition of V^{k+1} is close to that of V^k. However, neither the direct nor the iterative methods mentioned above can take advantage of a good initial guess. Assume that $V_\dagger^k := Q_\dagger^k \Sigma_+^k (Q_\dagger^k)^\top$ has low rank. Since $V_\dagger^k$ is the optimal solution of $\min_{S\succeq 0} \|S - V^k\|_F^2$, we can use nonlinear programming approaches, like the limited-memory BFGS method, to obtain $R^{k+1} := \arg\min_{R\in\mathbb{R}^{n\times\kappa}} \|RR^\top - V^{k+1}\|_F^2$ starting from $R := Q_\dagger^k(\Sigma_+^k)^{\frac{1}{2}}$, and set $S^{k+1} := R^{k+1}(R^{k+1})^\top$, where $\kappa := \kappa_+(V^k)$. Similarly, since $V_\ddagger^k$ is the optimal solution of $\min_{S\succeq 0} \|S + V^k\|_F^2$, we can compute $V_\ddagger^{k+1}$ from the optimal solution of $\min_{R\in\mathbb{R}^{n\times\kappa}} \|RR^\top + V^{k+1}\|_F^2$, where $\kappa := \kappa_-(V^k)$.

3.2 Updating the penalty parameter

Although our analysis shows that our alternating direction method Algorithm 1 converges for any fixed penalty parameter $\mu > 0$, numerical performance can be improved by adjusting the value of μ (see also [17,39]). We next present a strategy for doing this dynamically. Since the complementary condition $X^k S^k = 0$ is always satisfied, the pair

(X^k, y^k, S^k) is close to optimal if the primal and dual infeasibilities $\|\mathcal{A}(X^{k+1}) - b\|_2$ and $\|C - \mathcal{A}^*(y^{k+1}) - S^{k+1}\|_F$ are small. Hence, we can tune μ so that the primal and dual infeasibilities are balanced, that is, $\|\mathcal{A}(X^{k+1}) - b\|_2 \approx \|C - \mathcal{A}^*(y^{k+1}) - S^{k+1}\|_F$. Specifically, we have from Corollary 1 that

$$\mathcal{A}(X^{k+1}) - b = \frac{1}{\mu}\mathcal{A}(S^{k+1} - S^k) \text{ and } C - \mathcal{A}^*(y^{k+1}) - S^{k+1} = \mu(X^k - X^{k+1}),$$

which suggests that the primal and dual infeasibilities are proportional to $\frac{1}{\mu}$ and μ, respectively. Therefore, we decrease (increase) μ by a factor γ ($\frac{1}{\gamma}$), $0 < \gamma < 1$, if the primal infeasibility is less than (greater than) a multiple of the dual infeasibility for a number of consecutive iterations. In addition, μ is required to remain within an interval $[\mu_{\min}, \mu_{\max}]$, where $0 < \mu_{\min} < \mu_{\max} < \infty$.

3.3 Step size for updating the primal variable X

For many alternating direction methods [7,11,12,17,20,21,28], numerical performance is often improved if a step size is added to the update of the Lagrange multiplier. Here, we replace step (6c) by

$$\begin{aligned} X^{k+1} &:= X^k + \rho\frac{\mathcal{A}^*(y^{k+1}) + S^{k+1} - C}{\mu} = (1-\rho)X^k + \frac{\rho}{\mu}(S^{k+1} - V^{k+1}) \\ &:= (1-\rho)X^k + \rho\bar{X}^{k+1}, \end{aligned} \tag{26}$$

where $\rho \in (0, \frac{1+\sqrt{5}}{2})$ and $\bar{X}^{k+1} := \frac{1}{\mu}(S^{k+1} - V^{k+1})$. To prove convergence of this variant one can use an argument similar to the one in [17] to obtain the following theorem.

Theorem 4 *Let (X^*, y^*, S^*) be an optimal solution of (1) and (3), $\rho \in (0, \frac{1+\sqrt{5}}{2})$ and $T = 2 - \frac{1}{2}(1 + \rho - \rho^2)$. Then, we have*

$$\begin{aligned} &\|X^{k+1} - X^*\|_F^2 + \frac{\rho}{\mu^2}\|S^{k+1} - S^*\|_F^2 + \frac{\rho(T-\rho)}{\mu^2}\|\mathcal{A}^*(y^{k+1}) + S^{k+1} - C\|_F^2 \\ &\le \|X^k - X^*\|_F^2 + \frac{\rho}{\mu^2}\|S^k - S^*\|_F^2 + \frac{\rho(T-\rho)}{\mu^2}\|\mathcal{A}^*(y^k) + S^k - C\|_F^2 \\ &\quad - \frac{(1+\rho-\rho^2)\rho}{3\mu^2}\left(\|\mathcal{A}^*(y^{k+1}) + S^{k+1} - C\|_F^2 + \|S^k - S^{k+1}\|_F^2\right). \end{aligned} \tag{27}$$

Hence, we obtain

$$\lim_{k\to\infty}\left(\|\mathcal{A}^*(y^{k+1}) + S^{k+1} - C\|_F^2 + \|S^k - S^{k+1}\|_F^2\right) = 0. \tag{28}$$

Based on Theorem 4, we can show that both the primal and dual infeasibilities and the violation of the complementary condition converge to zero. From statement 1 of Corollary (1), we have

$$\begin{aligned}\|\mathcal{A}(X^{k+1}) - b\|_2 &\le \|\mathcal{A}(\bar{X}^{k+1}) - b\|_2 + \|\mathcal{A}(X^{k+1}) - \mathcal{A}(\bar{X}^{k+1})\|_2 \\ &\le \frac{1}{\mu}\|A\|_2 \, \|S^{k+1} - S^k\|_F + \|A\|_2 \, \|X^{k+1} - \bar{X}^{k+1}\|_F. \end{aligned} \quad (29)$$

We obtain from (26) that

$$\begin{aligned}\|X^{k+1}S^{k+1}\|_F &= (1-\rho)\|X^k S^{k+1}\|_F \\ &\le (1-\rho)\left(\|(X^k - \bar{X}^{k+1})S^{k+1}\|_F + \|\bar{X}^{k+1}S^{k+1}\|_F\right) \\ &= (1-\rho)\|(X^k - \bar{X}^{k+1})S^{k+1}\|_F \\ &\le (1-\rho)\|(X^k - \bar{X}^{k+1})\|_F \|S^{k+1}\|_F. \end{aligned} \quad (30)$$

It follows from (26) and (28) that

$$\lim_{k\to\infty} \|X^{k+1} - X^k\|_F = 0, \quad \lim_{k\to\infty} \|\bar{X}^{k+1} - X^k\|_F = 0, \quad \lim_{k\to\infty} \|X^{k+1} - \bar{X}^{k+1}\|_F = 0. \quad (31)$$

Combining (28)–(31), we obtain

$$\lim_{k\to\infty} \|\mathcal{A}^*(y^{k+1}) + S^{k+1} - C\|_F^2 = 0, \quad \lim_{k\to\infty} \|\mathcal{A}(X^{k+1}) - b\|_F^2 = 0,$$
$$\lim_{k\to\infty} \|X^{k+1}S^{k+1}\|_F = 0.$$

3.4 Termination rules and detection of stagnation

Since the rate of convergence of first-order methods can slow down as the iterates approach an optimal solution, it is critical to detect this stagnation and stop properly. However, it is difficult to predict whether an algorithm can get out of a region in which it is temporarily trapped and then resume a fast rate of convergence. Hence, it is usually beneficial to allow some flexibility in the termination rules. Similar to the rules used in the Seventh DIMACS Implementation Challenge, we measure the infeasibilities and closeness to optimality for the primal and dual problems as follows

$$\begin{aligned}\text{pinf} &= \frac{\|\mathcal{A}(X) - b\|_2}{1 + \|b\|_2}, \quad \text{dinf} = \frac{\|C + S + Z - \mathcal{A}^*(y)\|_F}{1 + \|C\|_1}, \\ \text{gap} &= \frac{|b^\top y - \langle C, X\rangle|}{1 + |b^\top y| + \langle C, X\rangle}. \end{aligned} \quad (32)$$

We stop our algorithm when

$$\delta := \max\{\text{pinf}, \text{dinf}, \text{gap}\} \leq \epsilon,$$

for $\epsilon > 0$. We use "it_stag" to count the number of consecutive iterations that δ does not decrease below the best value obtained thus far and stop if the following criteria are satisfied:

$$\begin{aligned}&(\text{it\_stag} > h_1 \text{ and } \delta <= 10\epsilon) \text{ or } (\text{it\_stag} > h_2 \text{ and } \delta <= 10^2\epsilon) \text{ or}\\ &(\text{it\_stag} > h_3 \text{ and } \delta <= 10^3\epsilon),\end{aligned} \tag{33}$$

where $1 < h_1 < h_2 < h_3$ are integers representing different levels of difficulty.

The complete pseduo-code for our algorithm SDPAD (SDP Alternating Direction) is presented in Algorithm 2. SDPAD uses eleven parameters, the values of only a few of which are critical to its convergence and performance. The default value of the termination tolerance ϵ is 10^{-6}, and h_1, h_2 and h_3 are used to detect stagnation. The following parameters are for adjusting the penalty parameter μ: the initial value of μ is 5, and its minimal and maximal values $\mu_{\min}$ and $\mu_{\max}$ are set to 10^{-4} and 10^4, respectively, the factor for reducing (increasing) μ is $\gamma = 0.5$, and this is done if there are h_4 consecutive iterations in which the ratio of the primal to the dual infeasibility is less than or equal to η_1 (greater than η_2). The step size ρ for updating X is set to 1.6. We choose the initial iterate $X^0 = I$ and $S^0 = \mathbf{0}$. By default, the decomposition $V_\dagger^{k+1}$ or $V_\ddagger^{k+1}$ is computed by the code "DSYEVX" in LAPACK for all eigenvalues in the interval $[10^{-12}, 10^{15}]$ and their corresponding eigenvectors. Note that SDPAD can be extended naturally to the expanded problem (20); we shall also refer to the resulting algorithm as SDPAD.

4 Numerical results

In this section, we report numerical results of our alternating direction algorithm SDPAD and the Newton-CG augmented Lagrangian method, whose implementation is called SDPNAL [41], on the SDP relaxations of frequency assignment, maximum stable set and binary integer quadratic programming problems. The main parts of our code were written in C Language MEX-files in MATLAB (Release 7.3.0). The main parts of SDPNAL were implemented in Matlab with a couple of tasks written in C Language MEX-files. All experiments were performed on a Dell Precision 670 workstation with an Intel Xeon 3.4 GHZ CPU and 6 GB of RAM.

We should point out that the current implementation of SDPAD is not aimed at SDPs with general linear constraints since inverting the matrix $\mathcal{A}\mathcal{A}^*$ can be expensive even if done only once. The comparisons are not meant to be a rigorous assessment of the performance of the solvers tested, as this would require very careful handling of subtle details such as parameter tuning and comparable termination criteria. Our main purpose is to demonstrate that alternating direction methods can be extremely robust and efficient if underlying special structures in the test problems are exploited.

Algorithm 2: SDPAD

Set $0 < \mu_{\min} \le \mu \le \mu_{\max} < +\infty, \epsilon > 0, \gamma \in (0,1), 0 < \eta_1 \le \eta_2 < \infty, \rho \in (0, \frac{1+\sqrt{5}}{2}), 1 < h_1 < h_2 < h_3$ and $h_4 > 1$. Set $X^0 \succeq 0$ and $S^0 \succeq 0$. Set
`eigS` $=$ `true`, `ref` $= +\infty$, `it_stag` $= 0$, `it_pinf` $= 0$ and `it_dinf` $= 0$.
for $k = 0, 1, \cdots$ **do**

S1 *Update y^{k+1}, S^{k+1} and X^{k+1}:*
Compute y^{k+1} according to (7) and V^{k+1} according to (9).
if `eigS` $==$ `true` **then**
Compute $V_\dagger^{k+1}$, set $S^{k+1} := V_\dagger^{k+1}$ and $X^{k+1} = \frac{1}{\mu}(S^{k+1} - V^{k+1})$.
if $\kappa(V_\dagger^{k+1}) \ge \frac{n}{2}$ **then** set `eigS` $=$ `false`.
else
Compute $V_\ddagger^{k+1}$, set $S^{k+1} := V^{k+1} + V_\ddagger^{k+1}$ and $X^{k+1} = \frac{1}{\mu} V_\ddagger^{k+1}$.
if $\kappa(V_\ddagger^{k+1}) \ge \frac{n}{2}$ **then** set `eigS` $=$ `true`.
Set $X^{k+1} := (1-\rho)X^k + \rho X^{k+1}$.

S2 *Check optimality and detect stagnation:*
Compute $\delta := \max\{$`pinf`, `dinf`, `gap`$\}$.
if $\delta \le \epsilon$ **then** return the solution.
if $\delta \le$ `ref` **then** set `ref` $:= \delta$ and `it_stag` $= 0$ **else** set `it_stag` $=$ `it_stag` $+ 1$.
if *condition* (33) *is satisfied* **then** return the solution.

S3 *Update penalty parameter μ:*
if `pinf/dinf` $\le \eta_1$ **then**
Set `it_pinf` $=$ `it_pinf` $+ 1$ and `it_dinf` $= 0$.
if `it_pinf` $\ge h_4$ **then** set $\mu = \max(\gamma\mu, \mu_{\min})$ and `it_pinf` $= 0$.
else if `pinf/dinf` $> \eta_2$ **then**
Set `it_dinf` $=$ `it_dinf` $+ 1$ and `it_pinf` $= 0$.
if `it_dinf` $\ge h_4$ **then** set $\mu = \min(\frac{1}{\gamma}\mu, \mu_{\max})$ and `it_dinf` $= 0$.

4.1 Frequency assignment relaxation

In this subsection, we demonstrate the effectiveness of SDPAD on the SDP relaxations of frequency assignment problems. These problems (see equations (4) and (5) on page 363 in [5], or equation (2) on page 5 of [23]) arise from wireless communication networks and contain both equality and inequality constraints. Let $G = (V, E)$ be an undirected graph with vertex set $V = \{1, \ldots, r\}$ and edge set $E \subseteq V \times V$, and let $W \in S^r$ be a weight matrix for G such that $w_{i,j} = w_{j,i}$ is the weight associated with edge $(i, j) \in E$. For those edges $(i, j) \notin E$, we assume $w_{i,j} = w_{j,i} = 0$. Let $T \subseteq E$ be a given edge subset. These problems can be formulated as:

$$\begin{aligned} \min \quad & \left\langle \frac{1}{2k}\mathrm{diag}(We) + \frac{k-1}{2k}W, X \right\rangle \\ s.t. \; & X_{i,j} \ge \tfrac{-1}{k-1}, \quad \forall (i,j) \in E \backslash T, \\ & X_{i,j} = \tfrac{-1}{k-1}, \quad \forall (i,j) \in T, \\ & \mathrm{diag}(X) = e, \quad X \succeq 0. \end{aligned} \tag{34}$$

Using the matrices corresponding to the edges in T and the constraint $\mathrm{diag}(X) = e$ to construct the operator $\mathcal{A}$, and the matrices corresponding to the edges in $E\backslash T$ to construct the operator $\mathcal{B}$, we can formulate (34) as the expanded problem (20) without the positivity constraints $X \geq 0$. We replaced the constraints $X_{ij} = \frac{-1}{k-1}$ by $X_{ij}/\sqrt{2} = \frac{-1}{\sqrt{2}(k-1)}$ and $X_{ij} \geq \frac{-1}{k-1}$ by $X_{ij}/\sqrt{2} \geq \frac{-1}{\sqrt{2}(k-1)}$, hence, $\mathcal{A}\mathcal{A}^* = I, \mathcal{B}\mathcal{B}^* = I$ and $\mathcal{A}$ and $\mathcal{B}$ are orthogonal to each other. Therefore, the optimal solution of the subproblems (24a) and (24b) are explicitly available:

$$y^{k+1} := -\Big(\mu(\mathcal{A}(X^k) - b) + \mathcal{A}(S^k - C)\Big) \text{ and } v^{k+1} := \max\Big(-\Big(\mu(\mathcal{B}(X^k) - d) + \mathcal{B}(S^k - C)\Big), \mathbf{0}\Big).$$

To accommodate the inequality constraints, the primal infeasibility was measured by

$$\text{pinf} = \frac{\|\mathcal{A}(X) - b\|_2 + \|\min(\mathcal{B}(X) - d, \mathbf{0})\|_2}{1 + \|b\|_2}.$$

Since the matrices corresponding to the operators $\mathcal{A}$ and $\mathcal{B}$ do not have to be stored, the memory required by our implementation is quite small.

The parameters of SDPNAL were set to their default values. The iteration counter set points h_1, h_2 and h_3 were set to 20, 150 and 300, respectively, the iteration counter h_4 for changing μ was set to 50 and the ratios η_1 and η_2 were set to 1. We stopped SDPAD when the total number of iterations reached 5,000. All other parameters were set to their default values. A summary of the computational results is presented in Table 1. In that table, $\hat{m}$ denotes the total number $m+q$ of equality and inequality constraints, "itr" denotes the total number of iterations performed and "cpu" denotes the CPU time reported in the format of "hours:minutes:seconds". In particular, "cpu(p)" and "cpu(f)" denote the cpu time obtained by SDPAD using partial ("DSYEVX") and full ("DSYEVD") eigenvalue decomposition, respectively. The advantage of using partial eigen-value decomposition was very obvious on the small instances from "fap01" to "fap12". However, its performance deteriorated on the large problems "fap25" and "fap36". Since both of the codes "DSYEVX" and "DSYEVD" reduce the targeted matrix to tridiagonal form and their difference is in how they compute the eigenvalues and eigenvectors from the resulting tridiagonal matrix, it may be possible to improve the performance of SDPAD with partial eigenvalue decomposition if a sophisticated strategy for specifying the lower and upper bounds of the interval to be searched for eigenvalues in "DSYEVX" can be found. Although SDPAD converged slowly on problems "fap12", "fap25" and "fap36", it was relatively easy to find solutions whose accuracy was of order 10^{-4}.

Since running SDPNAL on "fap25" and "fap36" is very time consuming (for example, in the results reported in [41], SDPNAL took more than 65 hours to solve problem "fap36"), we did not run SDPNAL on our own computer on these two problems and the results presented here were taken from Table 3 in [41]. Note that the numerical results of SDPNAL on problems from "fap01" to "fap12" in Table 1 are slightly different from those reported in Table 3 in [41]. Since the results in [41]

Table 1 Computational results on computing frequency assignment problems

Name	n	$\hat{m}$	SDPAD						SDPNAL			
			itr	gap	cpu(p)	cpu(f)	pinf	dinf	gap	cpu	pinf	dinf
fap01	52	1,378	607	9.06e−6	0.35	0.69	2.02e−7	1.83e−7	1.39e−7	6.31	8.28e−7	1.02e−7
fap02	61	1,866	666	5.29e−5	0.49	0.94	1.62e−6	1.47e−5	1.15e−5	4.12	8.38e−7	3.51e−7
fap03	65	2,145	840	1.48e−6	0.75	1.42	4.43e−6	4.30e−6	2.27e−6	7.44	1.14e−7	2.50e−7
fap04	81	3,321	718	2.18e−5	1.21	2.77	2.28e−6	1.08e−6	1.53e−5	19.69	2.07e−7	6.38e−7
fap05	84	3,570	768	8.84e−6	1.32	3.04	2.50e−6	1.69e−6	1.09e−5	31.59	5.21e−6	6.13e−7
fap06	93	4,371	506	8.91e−6	1.07	2.39	4.11e−6	2.66e−6	1.73e−5	29.84	6.87e−7	6.83e−7
fap07	98	4,851	543	9.89e−6	1.26	1.98	3.26e−6	4.95e−6	5.75e−6	29.88	9.93e−7	4.88e−7
fap08	120	7,260	424	8.39e−5	1.57	2.82	1.27e−5	5.29e−6	5.93e−6	25.25	2.80e−7	9.90e−7
fap09	174	15,225	505	2.06e−7	4.77	6.74	5.80e−7	5.08e−7	2.86e−6	59.67	8.92e−7	9.60e−7
fap10	183	14,479	1,250	4.96e−4	14.93	29.48	2.31e−6	9.17e−7	7.93e−5	1:50	1.49e−7	9.33e−7
fap11	252	24,292	1,654	4.99e−4	49.57	1:18	2.34e−6	4.51e−7	1.89e−4	5:15	9.48e−7	5.96e−7
fap12	369	26,462	4,207	1.36e−4	5:33	6:41	5.54e−7	3.11e−7	1.60e−4	13:14	6.07e−7	7.82e−7
fap25	2,118	322,924	4,752	7.47e−5	37:07:18	24:25:18	1.71e−6	2.41e−7	1.1e−4	10:53:22	3.2e−6	5.0e−7
fap36	4,110	1,154,467	5,000	1.68e−5	152:04:46	116:28:06	1.38e−6	1.26e−7	2.5e−5	65:25:07	7.7e−7	6.7e−7
fap12*	369	26,462	2,000	6.43e−4	2:34	3:51	1.88e−6	1.41e−6	1.60e−4	13:14	6.07e−7	7.82e−7
fap25*	2,118	322,924	2,000	6.97e−4	15:06:58	7:17:08	1.29e−5	1.27e−6	1.1e−4	10:53:22	3.2e−6	5.0e−7
fap36*	4,110	1,154,467	2,000	2.20e−4	78:37:23	53:14:12	1.48e−5	1.04e−6	2.5e−5	65:25:07	7.7e−7	6.7e−7

were obtained from a PC with Intel Xeon 3.2 GHZ and 4 GB of RAM which has a very similar performance profile to the computer that we used, the numbers reported in our table and Table 3 in [41] are very similar and the comparison between them is meaningful. From these two tables, we can see that SDPAD is quite competitive with SDPNAL for achieving a duality gap of almost the same order. Although the infeasibilities are not directly comparable since our formulation is different from the standard form used by SDPNAL, we can observe that the infeasibilities achieved by SDPAD were still satisfactory. The results of the boundary point method "mprw.m" reported in Table 4 in [41] are much worse than those of SDPAD. Since the implementation in "mprw.m" is essentially an alternating direction method applied to SDPs in standard form, we can conclude that treating inequality constraints directly can greatly improve the performance of such methods.

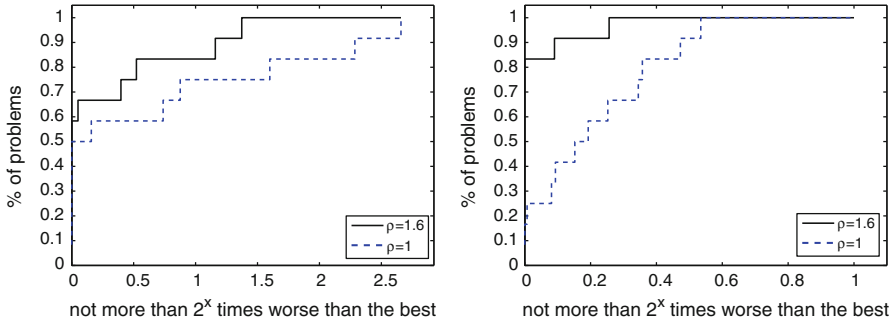
We now compare the numerical results of our alternating direction methods obtained using $\rho = 1$ and $\rho = 1.6$ by using performance profiles as proposed in [8]. Specifically, performance plots for the duality gap and the CPU time are presented in Fig. 1a, b, respectively. These figures show that the variant using $\rho = 1.6$ is both faster than the variant using $\rho = 1$ and achieves a smaller duality gap.

4.2 The SDP relaxation of the maximum stable set problem

Given a graph G with edge set E, two SDPs relaxations of the maximum stable set problem are

$$\theta(G) = \max\{\langle C, X\rangle, X_{ij} = 0, (i, j) \in E, \langle I, X\rangle = 1, X \succeq 0\}, \tag{35}$$

$$\theta_+(G) = \max\{\langle C, X\rangle, X_{ij} = 0, (i, j) \in E, \langle I, X\rangle = 1, X \succeq 0, X \geq 0\}, \tag{36}$$



(a) gap **(b)** cpu

Fig. 1 Performance profiles of two variants of SDPAD for frequency assignment problems

where $C = ee^\top$. We scaled the constraints so that $\mathcal{A}\mathcal{A}^* = I$, i.e., we replaced the constraints $X_{ij} = 0$ by $X_{ij}/\sqrt{2} = 0$ and $\langle I, X\rangle = 1$ by $\frac{1}{\sqrt{n}}\langle I, X\rangle = \frac{1}{\sqrt{n}}$. The matrix C was also scaled by n. The tests problems were taken from [19,25,27]. The numbers h_1, h_2 and h_3 were set to 20, 50 and 150, respectively, the iteration counter h_4 for changing μ was set to 100 and the ratios η_1 and η_2 were set to 1. We stopped SDPAD when the total number of iterations reached 5,000. All other parameters were set to their default values. Summaries of the computational results for $\theta(G)$ and $\theta_+(G)$ are presented in Tables 2 and 3, respectively. Since running SDPNAL on all the test problems is very time consuming (for example, in the results reported in [41], SDPNAL took almost 81 hours to compute $\theta_+(G)$ on problem "1et.2048"), we did not run SDPNAL on our own computer on any of the $\theta(G)$ and $\theta_+(G)$ or BIQ (see next subsection) problems. Hence, the SDPNAL results in Tables 2 and 3 are taken from Tables 5 and 6 in [41]. Because the computer used to obtain the results in [41] has very similar performance characteristics to the computer that we used, the comparison presented in Tables 2 and 3 is meaningful. Specifically, when we run SDPNAL on smaller and easier problems, such as "fap01"–"fap12", on our computer, the cpu time differed from those reported in [41] by an insignificant amount. From Table 2, we can see that SDPNAL performs better than SDPAD on most problems. However, SDPAD is faster than SDPNAL on "2dc.512", and it achieves a duality gap not much worse than that of SDPNAL on problems like "theta102" to "theta123", "sanr200-0.7", "c-fat200-1" and "brock400-1". From Table 3, we can see that SDPAD is quite competitive with SDPNAL. Specifically, it is faster than SDPNAL on problems like "2dc.512", "1tc.1024", "2dc.1024", "1tc.2048" and "1zc.2048" while achieving smaller duality gap. We should point out that SDPNAL is a generic code for SDPs and its performance may be significantly improved if its parameters are carefully tuned on the maximum stable set problems. Finally, performance plots for numerical results obtained using $\rho = 1$ and $\rho = 1.6$ for computing $\theta(G)$ and $\theta_+(G)$ are presented in Figs. 2a, b, and 3a, b, respectively. When both the final duality gap and CPU time are considered, these plots again show that using a fixed step size of $\rho = 1.6$ is preferable to a step size of $\rho = 1$.

Table 2 Computational results on computing $\theta(G)$

Name	n	$\hat{m}$	SDPAD					SDPNAL			
			itr	gap	cpu	pinf	dinf	gap	cpu	pinf	dinf
theta102	500	37,467	256	9.67e-7	47	6.47e−7	1.25e−6	1.6e−8	50	6.9e−8	4.8e−7
theta103	500	62,516	257	3.33e−7	57	2.10e−7	1.88e−6	4.6e−8	1:00	4.4e−8	5.8e−7
theta104	500	87,245	260	6.53e−7	48	2.51e−7	1.94e−6	7.6e−8	58	6.1e−8	6.5e−7
theta123	600	90,020	263	4.57e−7	1:43	2.52e−7	1.94e−6	4.1e−8	1:34	3.3e−8	5.2e−7
MANN-a27	378	703	503	8.81e−6	27	2.93e−6	6.99e−6	8.3e−8	07	9.4e−11	7.0e−7
sanr200-0.7	200	6,033	219	3.04e−7	04	5.93e−7	1.99e−6	1.4e−7	04	2.3e−7	3.1e−7
c-fat200-1	200	18,367	302	9.15e−7	04	8.39e−7	3.50e−7	8.5e−8	09	1.5e−7	8.3e−7
ham-8-3-4	256	16,129	199	8.04e−7	05	8.24e−7	1.98e−6	9.0e−8	02	7.2e−9	8.0e−7
ham-9-5-6	512	53,761	1,154	3.16e−8	3:09	1.66e−6	1.84e−6	1.3e−8	10	1.2e−10	2.4e−7
ham-10-2	1,024	23,041	597	9.51e−6	22:8	4.95e−6	6.31e−6	1.4e−6	1:33	7.1e−8	7.1e−7
brock400-1	400	20,078	254	8.30e−7	25	7.24e−7	1.61e−6	1.7e−8	26	5.4e−7	9.9e−7
keller4	171	5,101	249	9.32e−7	03	3.87e−7	1.97e−6	1.3e−8	05	1.3e−7	4.4e−7
p-hat300-1	300	33,918	764	9.96e−7	37	5.67e−7	1.36e−6	5.3e−7	1:45	5.5e−7	9.4e−7
G43	1,000	9,991	935	7.68e−6	21:17	5.90e−6	7.82e−6	4.2e−8	1:33	3.0e−8	4.6e−7
G44	1,000	9,991	933	6.82e−6	21:02	5.02e−6	8.41e−6	3.3e−7	2:59	3.6e−7	9.2e−7
G45	1,000	9,991	957	6.70e−6	21:27	1.15e−5	7.76e−6	5.6e−8	2:51	3.6e−8	5.8e−7
G46	1,000	9,991	927	9.69e−6	21:02	9.00e−6	1.09e−5	2.3e−7	2:53	3.2e−7	9.1e−7
G47	1,000	9,991	880	8.91e−6	19:41	7.23e−6	1.20e−5	1.3e−7	2:54	7.0e−8	9.3e−7
2dc.512	512	54,896	2,254	1.31e−4	12:42	5.82e−6	1.25e−5	1.7e−4	32:16	2.4e−5	5.0e−7
1dc.1024	1,024	24,064	747	2.79e−4	24:26	5.70e−5	1.02e−4	2.9e−6	41:26	1.4e−6	4.9e−7
1et.1024	1,024	9,601	603	7.11e−4	20:03	1.42e−4	2.64e−4	1.8e−6	1:01:14	2.5e−6	3.5e−7
1tc.1024	1,024	7,937	611	9.93e−4	21:47	3.28e−4	1.04e−3	2.2e−6	1:48:04	1.7e−6	6.3e−7
1zc.1024	1,024	16,641	608	4.16e−7	23:15	1.10e−6	7.56e−7	3.3e−8	4:15	2.8e−8	3.0e−7
2dc.1024	1,024	1,69,163	3,802	4.75e−5	3:01:37	2.93e−6	8.18e−6	9.9e−5	2:57:56	7.8e−6	6.8e−7
1dc.2048	2,048	58,368	473	9.99e−4	2:50:44	2.13e−4	4.33e−4	1.5e−6	6:11:11	7.7e−7	4.0e−7
1et.2048	2,048	22,529	904	9.93e−4	4:54:57	1.36e−4	8.10e−4	8.8e−7	7:13:55	6.9e−7	6.3e−7
1tc.2048	2,048	18,945	1,011	9.98e−04	5:37:33	1.26e−4	8.37e−4	7.9e−6	9:52:09	3.3e−6	3.7e−7
1zc.2048	2,048	39,425	941	9.17e−6	6:39:07	6.16e−6	6.20e−6	1.2e−6	45:16	1.5e−7	7.3e−7
2dc.2048	2,048	5,04,452	3,434	5.79e−5	21:15:29	2.56e−6	5.67e−6	4.4e−5	15:13:19	3.7e−6	4.5e−7

4.3 Binary integer quadratic programming problem

In this subsection, we report on how SDPAD performs on SDP relaxations of binary integer quadratic programming problems and compare these results to those obtained using SDPNAL. These problems have the form:

$$\begin{aligned} \min \quad & \left\langle \begin{pmatrix} Q & 0 \\ 0 & 0 \end{pmatrix}, X \right\rangle \\ s.t. \quad & X_{ii} - X_{n,i} = 0, \quad i = 1, \ldots, n-1, \\ & X_{nn} = 1, \quad X \succeq 0, \quad X \geq 0, \end{aligned} \tag{37}$$

Table 3 Computational results on computing $\theta_+(G)$

			SDPAD					SDPNAL			
Name	n	$\hat{m}$	itr	gap	cpu	pinf	dinf	gap	cpu	pinf	dinf
theta102	500	37,467	281	2.84e−7	1:01	2.17e−7	1.91e−6	8.4e−8	3:31	4.5e−7	9.1e−7
theta103	500	62,516	262	4.85e−7	1:01	2.42e−7	1.90e−6	2.3e−8	3:28	1.0e−7	9.3e−7
theta104	500	87,245	266	7.35e−7	52	2.64e−7	1.91e−6	1.6e−7	2:35	8.1e−7	8.4e−7
theta123	600	90,020	267	6.02e−7	1:39	2.78e−7	1.89e−6	1.2e−7	6:44	7.7e−7	8.5e−7
MANN-a27	378	703	530	8.17e−6	32	2.32e−6	8.15e−6	1.6e−7	35	2.1e−7	6.8e−7
sanr200-0.7	200	6,033	228	4.47e−7	05	6.53e−7	1.91e−6	2.9e−7	11	5.9e−7	4.0e−7
c-fat200-1	200	18,367	306	8.40e−7	04	4.33e−7	4.08e−7	2.1e−7	36	1.3e−7	9.5e−7
ham-8-3-4	256	16,129	199	8.05e−7	05	8.23e−7	1.98e−6	2.7e−10	05	8.5e−9	3.7e−7
ham-9-5-6	512	53,761	472	3.25e−7	2:19	7.31e−7	1.51e−6	2.6e−7	42	1.1e−7	4.4e−7
ham-10-2	1,024	23,041	653	9.78e−7	27:58	3.97e−7	1.10e−6	4.2e−7	4:35	6.0e−8	7.9e−7
brock400-1	400	20,078	258	9.83e−7	29	7.41e−7	1.51e−6	3.5e−9	1:45	9.5e−7	6.5e−7
keller4	171	5,101	331	2.98e−6	05	2.75e−6	1.80e−5	3.7e−7	43	6.1e−7	9.7e−7
p-hat300-1	300	33,918	567	1.50e−6	29	1.00e−6	1.18e−6	7.9e−7	6:50	8.7e−7	7.2e−7
G43	1,000	9,991	864	8.02e−6	20:22	1.30e−5	1.09e−5	2.1e−7	52:00	9.1e−7	8.1e−7
G44	1,000	9,991	893	6.36e−6	21:14	4.40e−6	8.72e−6	5.7e−8	49:32	3.3e−7	6.2e−7
G45	1,000	9,991	916	8.96e−6	22:54	1.38e−5	8.16e−6	2.4e−8	50:25	9.3e−7	8.6e−7
G46	1,000	9,991	886	9.47e−6	22:48	7.36e−6	1.04e−5	3.3e−8	44:38	3.5e−7	9.6e−7
G47	1,000	9,991	838	8.76e−6	21:11	6.56e−6	1.13e−5	5.1e−9	40:27	4.7e−7	6.0e−7
2dc.512	512	54,896	1,521	2.61e−5	5:10	2.14e−6	4.72e−6	3.8e−4	2:25:15	2.1e−4	7.7e−7
1dc.1024	1,024	24,064	1,022	6.23e−5	57:20	1.52e−5	1.18e−5	1.4e−5	5:03:49	1.4e−5	6.9e−7
1et.1024	1,024	9,601	651	5.26e−4	36:04	8.61e−5	4.32e−4	1.1e−5	6:45:50	4.8e−6	7.0e−7
1tc.1024	1,024	7,937	799	3.53e−4	49:13	1.25e−4	5.33e−4	8.7e−4	10:37:57	1.5e−4	7.3e−7
1zc.1024	1,024	16,641	506	5.30e−4	28:22	4.17e−5	3.61e−5	1.6e−7	40:13	6.4e−7	5.7e−7
2dc.1024	1,024	1,69,163	2,052	4.86e−6	1:25:12	5.70e−7	1.57e−6	7.3e−4	11:57:25	1.6e−4	6.2e−7
1dc.2048	2,048	58,368	517	9.99e−4	4:25:17	1.51e−4	1.89e−4	9.7e−5	35:52:44	1.9e−5	7.1e−7
1et.2048	2,048	22,529	659	6.80e−4	5:15:09	1.27e−4	5.33e−4	4.0e−5	80:48:17	6.3e−6	5.7e−7
1tc.2048	2,048	18,945	862	6.09e−4	6:35:33	1.36e−4	7.94e−4	1.4e−3	73:56:01	3.5e−4	7.9e−7
1zc.2048	2,048	39,425	953	8.27e−6	7:14:21	5.10e−6	9.21e−6	2.3e−7	2:13:04	2.5e−7	7.9e−7
2dc.2048	2,048	5,04,452	1,651	9.87e−6	11:00:08	7.99e−7	2.68e−6	2.7e−3	45:21:42	1.3e−4	7.2e−7

where $Q \in \mathbb{R}^{(n-1)\times(n-1)}$. We replaced the constraints $X_{ii} - X_{n,i} = 0$ by $\sqrt{\frac{2}{3}}(X_{ij} - X_{n,i}) = 0$ and the matrix Q was scaled by its Frobenious norm. The computational results obtained on the BIQ instances described in [32] are presented in Table 4, where "best upper bound" is the best known upper bound reported in [32], and "%pgap" and "%dgap" were computed as

$$\%\text{pgap} := \left|\frac{\text{best upper bound} - \text{pobj}}{\text{best upper bound}}\right| \times 100\% \text{ and}$$

$$\%\text{dgap} := \left|\frac{\text{best upper bound} - \text{dobj}}{\text{best upper bound}}\right| \times 100\%.$$

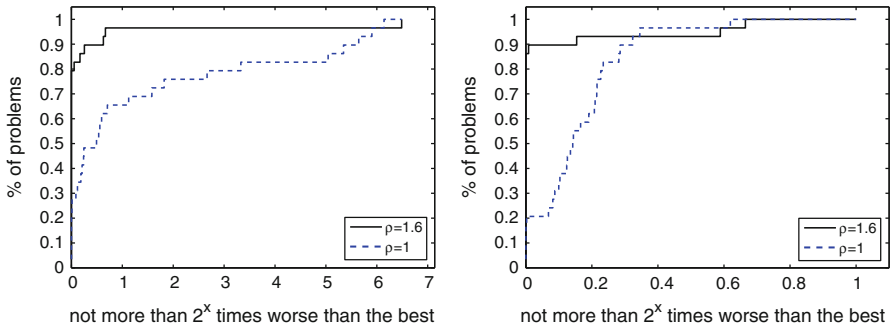


Fig. 2 Performance profiles of two variants of SDPAD for computing $\theta(G)$

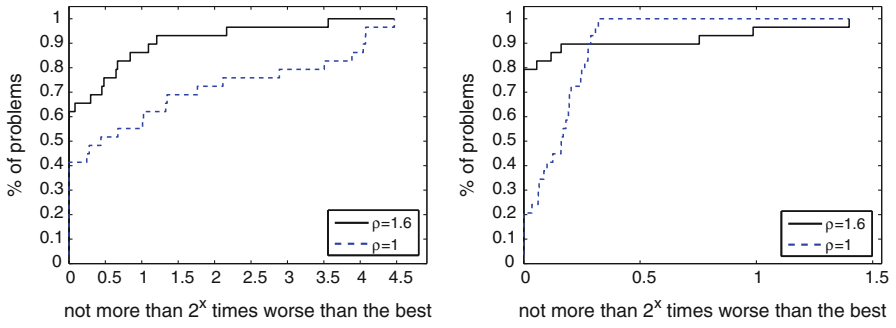


Fig. 3 Performance profiles of two variants of SDPAD for computing $\theta_+(G)$

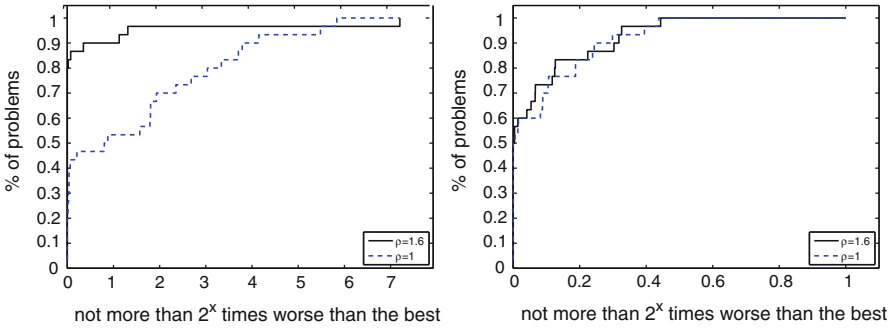
Note that %pgap and %dgap may not be valid since the values pobj and dobj obtained by SDPAD are not generated from primal and dual feasible iterates in general. The numbers h_1, h_2 and h_3 were set to 50,400 and 500, respectively, the iteration counter h_4 for changing μ was set to 0 and the ratios η_1 and η_2 were set to 1 and 100, respectively. We stopped SDPAD when the total number of iterations reached 10,000. The minimum penalty parameter $\mu_{\min}$ is set to 0.1. All other parameters were set to their default values. Again, we did not run SDPNAL on our own computer but presented the results reported in Table 8 in [41] in Table 4. From this table, we can see that SDPAD is faster than SDPNAL for achieving comparable lower bounds. We again emphasize that the performance of SDPNAL may be significantly improved if its parameters are carefully tuned on this class of problems. Finally, performance plots for numerical results obtained using $\rho = 1$ and $\rho = 1.6$ are presented in Fig. 4a, b. When both the final duality gap and CPU time are considered, these plots again show that using a fixed step size of $\rho = 1.6$ is preferable to using a step size of $\rho = 1$.

Table 4 Computational results on the BIQ problems

		SDPAD							SDPNAL			
Name	n	itr	best upper bound	%pgap	%dgap	cpu	pinf	dinf	%dgap	cpu	pinf	dinf
be200.3.1	201	2,484	−2.5453000e+4	8.891	8.891	32	8.55e−7	1.60e−6	8.891	10:29	5.6e−7	5.0e−7
be200.3.3	201	2,296	−2.8023000e+4	5.194	5.195	31	1.03e−5	4.47e−6	5.192	12:09	5.6e−5	5.7e−7
be200.3.5	201	4,782	−2.6355000e+4	6.519	6.519	1:04	3.25e−6	1.11e−7	6.519	10:38	1.4e−6	5.5e−7
be200.3.7	201	2,447	−3.0483000e+4	3.730	3.732	32	9.11e−6	5.55e−6	3.730	9:43	1.1e−6	5.8e−7
be200.3.9	201	6,940	−2.4683000e+4	7.106	7.106	1:31	1.22e−6	5.51e−8	7.106	8:28	3.2e−5	3.7e−7
be200.8.1	201	4,267	−4.8534000e+4	4.812	4.812	57	2.36e−6	6.83e−8	4.811	9:41	3.7e−5	6.2e−7
be200.8.3	201	2,107	−4.3207000e+4	7.051	7.053	28	9.94e−6	2.34e−6	7.052	10:53	5.8e−7	9.2e−7
be200.8.5	201	1,885	−4.1482000e+4	6.725	6.723	25	9.52e−6	3.33e−6	6.723	9:53	1.7e−5	7.7e−7
be200.8.7	201	2,186	−4.6828000e+4	5.394	5.391	28	8.72e−6	1.07e−6	5.392	4:30	4.7e−7	6.8e−7
be200.8.9	201	2,146	−4.3241000e+4	5.213	5.214	29	6.54e−6	1.75e−6	5.213	12:16	5.8e−6	3.8e−7
be250.1	251	5,923	−2.4076000e+4	4.334	4.334	2:14	1.22e−6	1.49e−7	4.332	16:41	4.0e−5	4.6e−7
be250.3	251	2,747	−2.2923000e+4	4.698	4.697	1:02	7.48e−6	2.18e−5	4.698	17:17	2.9e−5	6.0e−7
be250.5	251	6,326	−2.1057000e+4	6.258	6.258	2:24	1.22e−6	9.80e−8	6.254	14:30	9.3e−5	4.4e−7
be250.7	251	5,256	−2.4095000e+4	4.250	4.250	1:57	2.82e−6	2.22e−7	4.250	14:00	5.9e−5	7.1e−7
be250.9	251	6,532	−2.0051000e+4	6.713	6.713	2:30	1.22e−6	2.66e−7	6.713	17:13	1.1e−4	3.6e−7
bqp250-1	251	3,005	−4.5607000e+4	4.509	4.507	1:08	7.53e−6	1.17e−5	4.508	17:42	3.9e−7	6.6e−7
bqp250-3	251	3,006	−4.9037000e+4	4.159	4.159	1:05	3.68e−6	8.86e−6	4.160	10:36	9.9e−7	7.9e−7
bqp250-5	251	3,129	−4.7961000e+4	4.260	4.261	1:10	9.18e−6	5.05e−6	4.260	19:03	4.4e−5	6.9e−7
bqp250-7	251	4,801	−4.6757000e+4	4.630	4.630	1:46	6.79e−7	1.51e−6	4.630	16:36	8.2e−7	5.9e−7
bqp250-9	251	3,006	−4.8916000e+4	5.277	5.279	1:06	7.58e−6	1.67e−5	5.276	16:12	3.7e−7	3.9e−7
bqp500-1	501	8,960	−1.1658600e+5	8.044	8.044	19:44	2.76e−7	1.89e−6	8.045	1:00:59	2.9e−7	5.5e−7
bqp500-3	501	8,824	−1.3081200e+5	5.842	5.841	19:04	2.78e−7	1.77e−6	5.842	1:01:47	4.4e−7	4.0e−7
bqp500-5	501	8,288	−1.2548700e+5	6.857	6.857	18:41	5.35e−7	2.03e−6	6.857	1:36:43	4.5e−5	2.5e−7
bqp500-7	501	9,153	−1.2220100e+5	7.603	7.602	20:16	3.27e−7	1.99e−6	7.603	1:25:26	8.1e−5	5.7e−7
bqp500-9	501	8,439	−1.2079800e+5	7.856	7.856	18:54	3.33e−7	2.41e−6	7.857	1:24:40	9.5e−5	7.3e−7
gka2e	201	4,375	−2.3395000e+4	6.508	6.508	57	1.22e−6	5.75e−8	6.506	7:23	4.7e−7	4.3e−7
gka4e	201	2,735	−3.5594000e+4	4.583	4.582	36	1.02e−5	7.59e−7	4.582	11:25	1.2e−5	4.2e−7
gka1f	501	8,106	−6.1194000e+4	7.133	7.133	17:57	5.34e−7	2.83e−6	7.133	1:28:54	9.9e−5	5.2e−7
gka3f	501	7,785	−1.3803500e+5	8.778	8.777	17:04	4.05e−7	1.46e−6	8.778	1:31:34	2.8e−5	6.7e−7
gka5f	501	8,849	−1.9050700e+5	8.612	8.612	18:58	4.56e−7	1.58e−6	8.613	1:25:48	6.6e−6	7.1e−7

5 Conclusion

In this paper, we presented alternating direction augmented Lagrangian methods for solving semidefinite programming (SDP) problems. At each inner iteration, the algorithm minimizes the dual augmented Lagrangian function with respect to each block of dual variable separately while other blocks are fixed and then updates the primal variables. For the version of our algorithm that uses a unit step size $\rho = 1$, complementary is enforced by computing partial eigenvalue decompositions at each iteration. The special structure of the constraints, such as sparsity and orthogonality, can often



(a) gap **(b)** cpu

Fig. 4 Performance profiles of two variants of SDPAD for the BIQ problems

be used to simplify the computation when solving the subproblems. Our method can handle SDPs with inequality and positivity constraints directly without transforming them to equality constraints.

Although the numerical results are limited to three special classes of SDP problems, they illustrate the effectiveness of our approach. In order solve SDPs with general linear constraints efficiently, further research on solving each block of subproblems cheaply and partial spectral decomposition is still needed. Since the low rank structure of the optimal solution is often exposed after relatively few iterations from our numerical experience, we also plan to explore how to take advantage of this to improve the efficiency of our algorithms. Moreover, we note that the convergence of alternating direction methods for three or more separable blocks of variables remains an open question.

Acknowledgments We want to thank Kim-Chuan Toh and Defeng Sun for sharing their code SDPNAL and for providing the data files for the frequency assignment and maximum stable set problems, Samuel Burer for sharing his code SDPLR and Andreas Stathopoulos for his helpful comments on iterative methods for eigenvalue decomposition. The authors are grateful to the Area Editor and the four anonymous referees for their detailed and valuable comments and suggestions.

References

1. Bertsekas, D.P., Tsitsiklis, J.N.: Parallel and distributed computation: numerical methods. Prentice-Hall, Upper Saddle River (1989)
2. Burer, S.: Optimizing a polyhedral-semidefinite relaxation of completely positive programs. Tech. rep, Department of Management Sciences, University of Iowa (2008)
3. Burer, S., Monteiro, R.D.C.: A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. Math. Program. **95**, 329–357 (2003)
4. Burer, S., Monteiro, R.D.C.: Local minima and convergence in low-rank semidefinite programming. Math. Program. **103**, 427–444 (2005)
5. Burer, S., Monteiro, R.D.C., Zhang, Y.: A computational study of a gradient-based log-barrier algorithm for a class of large-scale SDPs. Math. Program. **95**, 359–379 (2003)
6. Burer, S., Vandenbussche, D.: Solving lift-and-project relaxations of binary integer programs. SIAM J. Optim. **16**, 726–750 (2006)
7. Chen, G., Teboulle, M.: A proximal-based decomposition method for convex minimization problems. Math. Program **64**, 81–101 (1994)

8. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Program. **91**, 201–213 (2002)
9. Eckstein, J., Bertsekas, D.P.: An alternating direction method for linear programming. LIDS-P, Cambridge, MA, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology (1967)
10. Eckstein, J., Bertsekas, D.P.: On the Douglas–Rachford splitting method and the proximal point algorithm for maximal monotone operators. Math. Program. **55**, 293–318 (1992)
11. Fortin, M., Glowinski, R.: Augmented Lagrangian methods. In: Studies in Mathematics and its Applications, vol.15. North-Holland Publishing Co., Amsterdam (1983) [Applications to the numerical solution of boundary value problems, Translated from the French by Hunt, B.D., Spicer, C.]
12. Glowinski, R., Le Tallec, P.: Augmented Lagrangian and operator-splitting methods in nonlinear mechanics. In: SIAM Studies in Applied Mathematics, vol. 9. Society for Industrial and Applied Mathematics (SIAM), Philadelphia (1989)
13. Goldfarb, D., Ma, S.: Fast alternating linearization methods for minimizing the sum of two convex functions. Tech. rep. IEOR, Columbia University (2009)
14. Goldfarb, D., Ma, S.: Fast multiple splitting algorithms for convex optimization. Tech. rep. IEOR, Columbia University (2009)
15. Hale, E.T., Yin, W., Zhang, Y.: Fixed-point continuation for l_1-minimization: methodology and convergence. SIAM J. Optim. **19**, 1107–1130 (2008)
16. He, B., Liao, L.-Z., Han, D., Yang, H.: A new inexact alternating directions method for monotone variational inequalities. Math. Program. **92**, 103–118 (2002)
17. He, B.S., Yang, H., Wang, S.L.: Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities. J. Optim. Theory Appl. **106**, 337–356 (2000)
18. Hiriart-Urruty, J.-B., Lemaréchal, C.: Convex analysis and minimization algorithms. I. Fundamentals. In: Grundlehren der Mathematischen Wissenschaften. Fundamental Principles of Mathematical Sciences, vol. 305. Springer, Berlin (1993)
19. Johnson, D.S., Trick, M.A. (eds.): Cliques, coloring, and satisfiability. In: DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 26. American Mathematical Society, Providence (1996) [Papers from the workshop held as part of the 2nd DIMACS Implementation Challenge in New Brunswick, NJ, October 11–13, 1993]
20. Kiwiel, K.C., Rosa, C.H., Ruszczyński, A.: Proximal decomposition via alternating linearization. SIAM J. Optim. **9**, 668–689 (1999)
21. Kontogiorgis, S., Meyer, R.R.: A variable-penalty alternating directions method for convex optimization. Math. Program. **83**, 29–53 (1998)
22. Malick, J., Povh, J., Rendl, F., Wiegele, A.: Regularization methods for semidefinite programming. SIAM J Optim. **20**, 336–356 (2009)
23. Pataki, G., Schmieta, S.: The dimacs library of semidefinite-quadratic-linear programs. Tech. rep., Center, Columbia University (1999)
24. Povh, J., Rendl, F., Wiegele, A.: A boundary point method to solve semidefinite programs. Computing **78**, 277–286 (2006)
25. Sloane, N.J.A.: Challenge problems: independent sets in graphs. http://research.att.com/njas/doc/graphs.html
26. Todd, M.J.: Semidefinite optimization. Acta Numer. **10**, 515–560 (2001)
27. Toh, K.-C.: Solving large scale semidefinite programs via an iterative solver on the augmented systems. SIAM J. Optim. **14**, 670–698 (2003)
28. Tseng, P.: Alternating projection-proximal methods for convex programming and variational inequalities. SIAM J. Optim. **7**, 951–965 (1997)
29. Vandenberghe, L., Boyd, S.: Semidefinite programming. SIAM Rev. **38**, 49–95 (1996)
30. Wang, Y., Yang, J., Yin, W., Zhang, Y.: A new alternating minimization algorithm for total variation image reconstruction. SIAM J. Imaging Sci. **1**, 248–272 (2008)
31. Wen, Z., Goldfarb, D., Ma, S., Scheinberg, K.: Row by row methods for semidefinite program ming. Technical report, Department of IEOR, Columbia University (2009)
32. Wiegele, A.: Biq mac library—a collection of max-cut and quadratic 0-1 programming instances of medium size. Technical report (2007)
33. Wolkowicz, H., Saigal, R., Vandenberghe, L. (eds.): Handbook of semidefinite programming: theory, algorithms, and applications. In: International Series in Operations Research & Management Science, vol. 27. Kluwer, Boston (2000)

34. Yang, J., Yuan, X.: An inexact alternating direction method for trace norm regularized least squares problem. Technical report, Department of Mathematics, Nanjing University (2010)
35. Yang, J., Zhang, Y.: Alternating direction algorithms for l1-problems in compressive sensing. Technical report, Rice University (2009)
36. Yang, J., Zhang, Y., Yin, W.: An efficient tvl1 algorithm for deblurring multichannel images corrupted by impulsive noise. SIAM J. Sci. Comput. **31**, 2842–2865 (2008)
37. Ye, C., Yuan, X.: A descent method for structured monotone variational inequalities. Optimi Methods Softw **22**, 329–338 (2007)
38. Yu, Z.: Solving semidefinite programming problems via alternating direction methods. J. Comput. Appl. Math. **193**, 437–445 (2006)
39. Yuan, X.: Alternating direction methods for sparse covariance selection. Technical report, Department of Mathematics, Hong Kong Baptist University (2009)
40. Zhang, Y.: User's guide for yall1: Your algorithms for l1 optimization. Technical report, Rice Univer sity (2009)
41. Zhao, X., Sun, D., Toh, K.: A newton-cg augmented lagrangian method for semidefinite programming. SIAM J. Optim. **20**, 1737–1765 (2010)