ALTERNATING DIRECTION AUGMENTED LAGRANGIAN METHODS FOR SEMIDEFINITE PROGRAMMING

ZAIWEN WEN[†], DONALD GOLDFARB[†], AND WOTAO YIN [‡]

August 10, 2009

Abstract. We present an alternating direction method based on an augmented Lagrangian framework for solving semidefinite programming (SDP) problems in standard form. At each iteration, the algorithm, also known as a two-splitting scheme, minimizes the dual augmented Lagrangian function sequentially with respect to the Lagrange multipliers corresponding to the linear constraints, then the dual slack variables and finally the primal variables, while in each minimization keeping the other variables fixed. Convergence is proved by using a fixed-point argument. A multiple-splitting algorithm is then proposed to handle SDPs with inequality constraints and positivity constraints directly without transforming them to the equality constraints in standard form. Finally, numerical results for frequency assignment, maximum stable set and binary integer quadratic programming problems are presented to demonstrate the robustness and efficiency of our algorithm.

Key words. semidefinite programming, alternating direction method, augmented Lagrangian method

AMS subject classifications. 90C06, 90C22, 90C30, 90C35

1. Introduction.

In this paper we present an alternating direction method based on an augmented Lagrangian framework for solving semidefinite programming (SDP) problems. These convex optimization problems are solvable in polynomial time by interior point methods [24, 27, 31]. 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. In comparison, the computational cost of each iteration of our method is much cheaper, particularly, if any sparsity in the SDP constraints is exploited. This enables it to solve very large SDPs efficiently.

First-order augmented Lagrangian approaches have been proposed for both the primal and dual formulations of SDPs, and different ways have been used to minimize the augmented Lagrangian function depending on 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 eigenvalue decomposition are used to minimize the primal augmented Lagrangian function. In [29], 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 [35], 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 [20] (and the related boundary point method [22]) 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.

[†]Department of Industrial Engineering and Operations Research, Columbia University, New York, 10027, U.S.A. (zw2109@columbia.edu, goldfarb@columbia.edu). Research supported in part by NSF Grant DMS 06-06712, ONR Grant N00014-08-1-1118 and DOE Grant DE-FG02-08ER58562.

[‡]Department of Computational and Applied Mathematics, Rice University, Texas, 77005, U.S.A. (wotao.yin@rice.edu). Research supported in part by NSF CAREER Award DMS-07-48839 and ONR Grant N00014-08-1-1101.

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 [14, 15], linear programming [9], nonlinear convex optimization [1, 19, 7, 18, 26], maximal monotone operators [10] and nonsmooth ℓ_1 minimization arising from compressive sensing [28, 32, 34]. In [33], 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 Lagrange multipliers 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 [20]. While the theoretical algorithm in [20] updates the primal variable X only after a certain condition is satisfied, the actual algorithm implemented in [20] (i.e., Algorithm 5.1 in [20]) 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 [20] 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 some extra steps to our alternating direction method to minimize the dual augmented Lagrangian function with respect to the Lagrange multipliers corresponding to the inequality constraints. This gives us a multiple-splitting alternating direction method. Numerical experiments on, for example, frequency assignment problems show that the performance of our method is significantly better than those in [20, 35].

Our contributions are as follows. Although the techniques for analyzing the alternating direction methods for variational inequalities in [15] 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 [20] has not been studied. Moreover, we present a multiple-splitting alternating direction method to solve SDPs with inequality constraints directly without transforming them into SDPs in standard form by introducing a lot of auxiliary variables and constraints.

The rest of this paper is organized as follows. We present an alternating direction augmented Lagrangian method for SDP in standard form in subsection 2.1 and analyze its convergence in subsection 2.2, and then extend this method to an expanded problem with inequality and positivity constraints in subsection 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 section 3. Finally, numerical results for frequency assignment, maximum stable set and binary integer quadratic programming problems are presented in section 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 $\|A\|_F := \sqrt{\sum_{i,j} A_{i,j}^2}$.

2. Alternating direction augmented Lagrangian methods.

2.1. A two-splitting scheme for standard form SDPs. Consider the standard form SDP

$$\min_{X\in S^n} \quad \langle C, X\rangle \quad \text{s.t.} \quad \mathcal{A}(X) = b, \quad X \succeq 0, \tag{2.1}$$

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

$$\mathcal{A}(X) := \left(\left\langle A^{(1)}, X\right\rangle, \cdots, \left\langle A^{(m)}, X\right\rangle\right)^\top, \tag{2.2}$$

the matrices $C, A^{(i)} \in S^n$, $i = 1, \cdots, m$, and the vector $b \in \mathbb{R}^m$ are given, and the unknown matrix $X \in S^n_+$. Let $\mathbf{vec}(X)$ 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 $\mathbf{mat}(x)$ be a matrix X such that $x = \mathbf{vec}(X)$. Note that the equation $\mathcal{A}(X) = b$ is equivalent to $A\,\mathbf{vec}(X) = b$, where

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

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

We make the following assumption throughout our presentation.

Assumption 2.1. *The matrix A has full row rank and the Slater condition holds for* (2.1)*; that is, there exists a matrix $\bar{X} \succ 0$ satisfying $\mathcal{A}(\bar{X}) = b$.*

The dual problem of (2.1) is

$$\min_{y\in\mathbb{R}^m, S\in S^n} \quad -b^\top y \quad \text{s.t.} \quad \mathcal{A}^*(y) + S = C, \quad S \succeq 0. \tag{2.3}$$

The augmented Lagrangian function for the dual SDP (2.3) 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

$$\min_{y\in\mathbb{R}^m, S\in S^n} \mathcal{L}_\mu(X^k, y, S), \quad \text{s.t.} \quad S \succeq 0, \tag{2.4}$$

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

$$X^{k+1} := X^k + \frac{\mathcal{A}^*(y^{k+1}) + S^{k+1} - C}{\mu}. \tag{2.5}$$

Since solving problem (2.4) exactly is very expensive, we consider instead an alternating direction method. Starting from X^0 and S^0 at the first iteration, we update on the kth iteration the variables y, S and X by first minimizing $\mathcal{L}_\mu(X,\ y,\ S)$ with respect to y to obtain y^{k+1} with $X := X^k$ and $S := S^k$ fixed; then minimizing $\mathcal{L}_\mu(X,\ y,\ S)$ with respect to S subject to $S \succeq 0$ to obtain S^{k+1} with $X := X^k$ and $y := y^{k+1}$ fixed; and finally, updating X^k by (2.5); that is, we compute

$$y^{k+1} := \arg\min_{y\in\mathbb{R}^m}\ \mathcal{L}_\mu(X^k,\ y,\ S^k), \tag{2.6a}$$

$$S^{k+1} := \arg\min_{S\in S^n}\ \mathcal{L}_\mu(X^k,\ y^{k+1},\ S), \quad S \succeq 0, \tag{2.6b}$$

$$X^{k+1} := X^k + \frac{\mathcal{A}^*(y^{k+1}) + S^{k+1} - C}{\mu}. \tag{2.6c}$$

The first-order optimality conditions for (2.6a) 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 2.1 $\mathcal{A}\mathcal{A}^*$ is invertible , we obtain $y^{k+1} := y(S^k, X^k)$, where

$$y(S,X) := -(\mathcal{A}\mathcal{A}^*)^{-1}\left(\mu(\mathcal{A}(X) - b) + \mathcal{A}(S - C)\right). \tag{2.7}$$

By rearranging the terms of $\mathcal{L}_\mu(X^k, y^{k+1}, S)$, it is easily verified that problem (2.6b) is equivalent to

$$\min_{S\in S^n}\quad \left\|S - V^{k+1}\right\|_F^2, \quad S \succeq 0, \tag{2.8}$$

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

$$V(S,X) := C - \mathcal{A}^*(y(S,X)) - \mu X. \tag{2.9}$$

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

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

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 equation (2.6c) that

$$X^{k+1} := X^k + \frac{\mathcal{A}^*(y^{k+1}) + S^{k+1} - C}{\mu} = \frac{1}{\mu}(S^{k+1} - V^{k+1}) = \frac{1}{\mu}V_\ddagger^{k+1}, \tag{2.10}$$

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

$$\min_{X\in S^n}\quad \left\|\mu X + V^{k+1}\right\|_F^2, \quad X \succeq 0. \tag{2.11}$$

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

Remark 2.2. *In the boundary point method [22] and regularization method [20], X^k is fixed until*

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 (2.7).
    Compute $V^{k+1}$ and its eigenvalue decomposition, and set $S^{k+1} := V_{\dagger}^{k+1}$.
    Compute $X^{k+1} = \frac{1}{\mu}(S^{k+1} - V^{k+1})$.
```

$\frac{1}{\mu}(S^{k+1} - V^{k+1})$ is nearly feasible. However, the actual regularization method implemented in the numerical experiments in [20] is exactly Algorithm 1.

Remark 2.3. *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* (2.6a)*, i.e.,* (2.7)*, 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, the strategies in [7, 10] can be used to compute an approximate minimizer y^{k+1} in step* (2.6a)*.*

2.2. Convergence analysis of Algorithm 1. Although the techniques for analyzing the convergence properties of the alternating direction methods for optimization problems arising from PDEs in [12], variational inequalities in [15] and a class of nonlinear optimization problems in [1] can be applied to our algorithm, we present here a different and simple argument by formulating our algorithm as a fixed point method. For any matrix $V \in S^n$, let the matrix $\left(V_{\dagger}, V_{\ddagger}\right)$ be denoted by $\mathcal{P}(V)$. Hence, each iteration of Algorithm 1 can be expressed as

$$y^{k+1} := y(S^k, X^k) \text{ and } \left(S^{k+1}, \mu X^{k+1}\right) := \mathcal{P}(V^{k+1}) = \mathcal{P}(V(S^k, X^k)). \tag{2.12}$$

We first describe a result of the orthogonal decomposition.

Lemma 2.4. *(Theorem 3.2.5 in [16], J.-J. Moreau) Let K be a closed convex cone and $K^{\diamond}$ be the polar cone of K, that is, $K^{\diamond} := \{s \in \mathbb{R}^n : \langle s, x\rangle \le 0$ 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 (2.1) is a fixed point of the equations (2.12).

Lemma 2.5. *Suppose Assumption 2.1 holds. Then, there exists primal and dual optimal solutions (X, y, S) for* (2.1) *and* (2.3) *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*

$$y = y(S, X) \text{ and } \left(S, \mu X\right) = \mathcal{P}(V(S, X)).$$

Proof. The proof here is similar to Proposition 2.6 in [20]. Since the Slater condition holds, there exists primal and dual optimal solution (X, y, S) for (2.1) and (2.3) so that statement (i) is true. Direct algebraic manipulation shows that $y = y(S, X)$ from statement (i). It follows from Lemma 2.4 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). $\square$

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

LEMMA 2.6. *For any $V, \widehat{V} \in S^n$,*

$$\left\| \mathcal{P}(V) - \mathcal{P}(\widehat{V}) \right\|_F \le \| V - \widehat{V} \|_F, \tag{2.13}$$

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 $\mathbf{Tr}(V_\dagger^\top \widehat{V}_\ddagger) \ge 0$ and $\mathbf{Tr}(\widehat{V}_\dagger^\top V_\ddagger) \ge 0$. Expanding terms of $V - \widehat{V}$, we obtain

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

which proves (2.13). $\square$

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

LEMMA 2.7. *For any $S, X, \widehat{S}, \widehat{X} \in S_+^n$,*

$$\| V(S, X) - V(\widehat{S}, \widehat{X}) \|_F \le \left\| \left(S - \widehat{S}, \mu(X - \widehat{X}) \right) \right\|_F \tag{2.14}$$

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 (2.7), 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 (2.9) gives

$$\begin{aligned}
V(S, X) - V(\widehat{S}, \widehat{X}) &= (C - \mathcal{A}^*(y(S, X)) - \mu X) - (C - \mathcal{A}^*(y(\widehat{S}, \widehat{X})) - \mu \widehat{X}) \\
&= \mathcal{A}^*(y(\widehat{S}, \widehat{X}) - y(S, X)) + \mu(\widehat{X} - X) \\
&= \mathbf{mat}\left(-\mu\,(I - M)\,\mathbf{vec}(X - \widehat{X}) + M\mathbf{vec}(S - \widehat{S})\right),
\end{aligned} \tag{2.15}$$

where $M := A^\top(AA^\top)^{-1}A$. Since M is an orthogonal projection matrix whose spectral radius is 1, we obtain from (2.15) that

$$\begin{aligned}\left\|V(S,X)-V(\widehat{S},\widehat{X})\right\|_F^2 &= \left\|\mu\,(I-M)\,\mathbf{vec}(X-\widehat{X})-M\mathbf{vec}(S-\widehat{S})\right\|_2^2\\ &\le \|\mu\mathbf{vec}(X-\widehat{X})\|_2^2+\|\mathbf{vec}(S-\widehat{S})\|_2^2\\ &= \left\|\left(S-\widehat{S},\mu(X-\widehat{X})\right)\right\|_F^2,\end{aligned} \tag{2.16}$$

which proves (2.14).

If the equality in (2.14) holds, it also holds in (2.16); that is,

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

This implies that $M\mathbf{vec}(X-\widehat{X})=0$ and $(I-M)\mathbf{vec}(S-\widehat{S})=0$. Using this relations in (2.15), we obtain

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

which proves the second statement. □

Lemma 2.8. *Let (X^*,y^*,S^*), where $y^*=y(S^*,X^*)$, be an optimal solution of* (2.1) *and* (2.3). *Under Assumption 2.1, if*

$$\|\mathcal{P}(V(S,X))-\mathcal{P}(V(S^*,X^*))\|_F=\left\|\left(S-S^*,\mu(X-X^*)\right)\right\|_F, \tag{2.17}$$

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 (2.1) *and* (2.3).

Proof. From Lemma 2.5, we have $(S^*,\mu X^*)=\mathcal{P}(V(S^*,X^*))$. From Lemmas 2.6 and 2.7, 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 2.4 that $(S,\mu X)=\mathcal{P}(V(S,X))$. □

Given Lemmas 2.6, 2.7 and 2.8, we can prove convergence of Algorithm 1 by following the proof of Theorem 4.5 in [13].

Theorem 2.9. *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* (2.1) *and* (2.3).

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 non-increasing. Therefore,

$$\lim_{k\to\infty} \left\|(S^k, \mu X^k) - (S^*, \mu X^*)\right\|_F = \left\|(\bar{S}, \mu\bar{X}) - (S^*, \mu X^*)\right\|_F, \tag{2.18}$$

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 2.8 to get that $(\bar{S}, \bar{y}, \mu\bar{X})$, where $\bar{y} = y(\bar{S}, \bar{X})$, is an optimal solution to problems (2.1) and (2.3). Finally, by setting $(S^*, \mu X^*) = (\bar{S}, \mu\bar{X})$ in (2.18), we get that

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

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

Proof. 1). It follows from (2.7), (2.9) and Assumption 2.1 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

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

2) Rearranging the terms of (2.6c), we obtain the first part of statement 2. The nonexpansiveness of the projection $V_\ddagger^{k+1}$ gives

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

3) From Lemmas 2.6 and 2.7, we obtain

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

□

2.3. A multiple-splitting scheme for an expanded problem. Although SDP problems with inequality constraints can be put into the standard form (2.1), it is often more convenient to treat the inequality constraints directly in order to preserve special structure of the constraints, like sparsity and orthogonality. We now extend our alternating direction method (2.6a)-(2.6c) to solve an expanded SDP problem, that includes, in particular, positivity constraints on the elements of the matrix X; i.e.,

$$\min_{X\in S^n} \langle C, X\rangle, \quad \text{s.t.} \quad \mathcal{A}(X)=b, \quad \mathcal{B}(X)\geq d, \quad X\succeq 0, \quad X\geq 0, \tag{2.19}$$

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

$$\mathcal{B}(X) := \left(\left\langle B^{(1)}, X\right\rangle, \cdots, \left\langle B^{(q)}, X\right\rangle\right)^\top, \quad B^{(i)}\in S^n, i=1,\cdots,q. \tag{2.20}$$

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

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

We also make the following assumption.

ASSUMPTION 2.11. *The matrices A and B have full row rank and a refined Slater condition holds for* (2.19)*; 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 (2.19) is

$$\min_{y\in\mathbb{R}^m, v\in\mathbb{R}^q, S\in S^n, Z\in S^n} -b^\top y - d^\top v, \quad \text{s.t.} \quad \mathcal{A}^*(y)+\mathcal{B}^*(v)+S+Z=C, \quad v\geq 0, \quad S\succeq 0, \quad Z\geq 0. \tag{2.21}$$

The augmented Lagrangian function for the dual SDP (2.21) corresponding to the linear constraints is defined as:

$$\begin{aligned}\mathcal{L}_\mu(X,y,v,Z,S) := & -b^\top y - d^\top v + \langle X, \mathcal{A}^*(y)+\mathcal{B}^*(v)+S+Z-C\rangle \\ & + \frac{1}{2\mu}\|\mathcal{A}^*(y)+\mathcal{B}^*(v)+S+Z-C\|_F^2,\end{aligned} \tag{2.22}$$

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 (2.6a)-(2.6c) as follows

$$y^{k+1} := \arg\min_{y\in\mathbb{R}^m} \mathcal{L}_\mu(X^k,\ y,\ v^k,\ Z^k,\ S^k), \tag{2.23a}$$

$$v^{k+1} := \arg\min_{v\in\mathbb{R}^q} \mathcal{L}_\mu(X^k,\ y^{k+1},\ v,\ Z^k,\ S^k), \quad v\geq 0, \tag{2.23b}$$

$$Z^{k+1} := \arg\min_{Z\in S^n} \mathcal{L}_\mu(X^k,\ y^{k+1},\ v^{k+1},\ Z,\ S^k), \quad Z\geq 0, \tag{2.23c}$$

$$S^{k+1} := \arg\min_{S\in S^n} \mathcal{L}_\mu(X^k,\ y^{k+1},\ v^{k+1},\ Z^{k+1},\ S), \quad S\succeq 0, \tag{2.23d}$$

$$X^{k+1} := X^k + \frac{\mathcal{A}^*(y^{k+1})+\mathcal{B}^*(v^{k+1})+S^{k+1}+Z^{k+1}-C}{\mu}. \tag{2.23e}$$

Note that y^{k+1}, S^{k+1} and X^{k+1} can be computed in the same fashion as in (2.7), (2.8) and (2.10), 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 (2.23b) is equivalent to the strictly convex quadratic program

$$\min_{v\in\mathbb{R}^n} \quad \left(\mathcal{B}\left(X^k+\frac{1}{\mu}Y^{k+1}\right)-d\right)^{\top} v+\frac{1}{2\mu}v^{\top}(\mathcal{B}\mathcal{B}^*)v, \quad v\geq 0, \tag{2.24}$$

where $Y^{k+1} := \mathcal{A}^*(y^{k+1})+S^k+Z^k-C$. 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 (2.23c) 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 (2.23c) is $Z^{k+1}=U_+^{k+1}$, the positive part of U^{k+1}, that is, $(U_+^{k+1})_{i,j} := \max(U_{ij}^{k+1}, 0)$.

3. Practical Issues.

In this section, 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. Our focus is on procedures (2.6a)-(2.6c) for problem (2.1), but our discussion applies equally to the expanded problem (2.19).

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 $\kappa_+(V^{k-1})\leq \frac{n}{2}$, 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 5000 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 2.10 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. We next present a strategy for doing this dynamically. Since the complementary condition $X^kS^k = 0$ 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 2.10 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 [11, 12, 15, 19, 7, 18, 26], numerical performance is often improved if a step size is added to the update of the Lagrange multiplier. Here, we replace step (2.6c) by

$$X^{k+1} := X^k + \rho\frac{\mathcal{A}^*(y^{k+1}) + S^{k+1} - C}{\mu} = (1-\rho)X^k + \frac{\rho}{\mu}(S^{k+1} - V^{k+1}) := (1-\rho)X^k + \rho\bar{X}^{k+1}, \tag{3.1}$$

where $\rho \in (0, \frac{1+\sqrt{5}}{2})$ and $\bar{X}^{k+1} := \frac{1}{\mu}(S^{k+1} - V^{k+1})$. Convergence of this variant of the algorithm can be proved in the same fashion as in [15]. Specifically, the following property holds.

THEOREM 3.1. *Let (X^*, y^*, S^*) be an optimal solution of* (2.1) *and* (2.3)*, $\rho \in (0, \frac{1+\sqrt{5}}{2})$ and $T = 2 - \frac{1}{2}(1 + \rho - \rho^2)$. Then, we have*

$$\begin{aligned} &\|X^{k+1} - X^*\|_F^2 + \frac{\rho}{\mu^2}\|S^{k+1} - S^*\|_F^2 + \frac{\rho(T-\rho)}{\mu^2}\|\mathcal{A}^*(y^{k+1}) + S^{k+1} - C\|_F^2 \\ \le\ &\|X^k - X^*\|_F^2 + \frac{\rho}{\mu^2}\|S^k - S^*\|_F^2 + \frac{\rho(T-\rho)}{\mu^2}\|\mathcal{A}^*(y^k) + S^k - C\|_F^2 \\ &-\frac{(1+\rho-\rho^2)\rho}{3\mu^2}\left(\|\mathcal{A}^*(y^{k+1}) + S^{k+1} - C\|_F^2 + \|S^k - S^{k+1}\|_F^2\right). \end{aligned} \tag{3.2}$$

Hence, we obtain

$$\lim_{k\to\infty}\left(\|\mathcal{A}^*(y^{k+1}) + S^{k+1} - C\|_F^2 + \|S^k - S^{k+1}\|_F^2\right) = 0. \tag{3.3}$$

Based on Theorem 3.1, 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 (2.10), we have

$$\begin{aligned}\|\mathcal{A}(X^{k+1})-b\|_2 &\leq \|\mathcal{A}(\bar{X}^{k+1})-b\|_2+\|\mathcal{A}(X^{k+1})-\mathcal{A}(\bar{X}^{k+1})\|_2\\ &\leq \frac{1}{\mu}\|A\|_2\,\|S^{k+1}-S^k\|_F+\|A\|_2\,\|X^{k+1}-\bar{X}^{k+1}\|_F.\end{aligned}\tag{3.4}$$

We obtain from (3.1) that

$$\begin{aligned}\|X^{k+1}S^{k+1}\|_F=(1-\rho)\|X^kS^{k+1}\|_F &\leq (1-\rho)\left(\|(X^k-\bar{X}^{k+1})S^{k+1}\|_F+\|\bar{X}^{k+1}S^{k+1}\|_F\right)\\ &=(1-\rho)\|(X^k-\bar{X}^{k+1})S^{k+1}\|_F\leq(1-\rho)\|(X^k-\bar{X}^{k+1})\|_F\|S^{k+1}\|_F.\end{aligned}\tag{3.5}$$

It follows from (3.1) and (3.3) that

$$\lim_{k\infty}\|X^{k+1}-X^k\|_F=0,\quad \lim_{k\infty}\|\bar{X}^{k+1}-X^k\|_F=0,\quad \lim_{k\infty}\|X^{k+1}-\bar{X}^{k+1}\|_F=0.\tag{3.6}$$

Combining (3.3)-(3.6), we obtain

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

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

$$\text{pinf}=\frac{\|\mathcal{A}(X)-b\|_2}{1+\|b\|_2},\quad \text{dinf}=\frac{\|C+S+Z-\mathcal{A}^*(y)\|_F}{1+\|C\|_1},\quad \text{gap}=\frac{|b^\top y-\langle C,X\rangle\,|}{1+|b^\top y|+\langle C,X\rangle}.\tag{3.7}$$

We stop our algorithm when

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

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:

$$(\text{it\_stag}>h_1 \text{ and } \delta<=10\epsilon) \text{ or } (\text{it\_stag}>h_2 \text{ and } \delta<=10^2\epsilon) \text{ or } (\text{it\_stag}>h_3 \text{ and } \delta<=10^3\epsilon),\tag{3.8}$$

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 $S^0 = \mathbf{0}$. Note that SDPAD can be extended naturally to the expanded problem (2.19); we shall also refer to the resulting algorithm as SDPAD .

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 (2.7) and $V^{k+1}$ according to (2.9).
          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 $\delta := \max\{\text{pinf}, \text{dinf}, \text{gap}\}$.
          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 (3.8) is satisfied then return the solution.
S3      Update penalty parameter $\mu$:
          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. Numerical Results. Although the numerical results that we present in this section are limited to three special classes of SDP problems, they illustrate the effectiveness of our alternating direction methods. The main parts of our code were written in C Language MEX-files in MATLAB (Release 7.3.0), and all experiments were performed on a Dell Precision 670 workstation with an Intel Xeon 3.4GHZ CPU and 6GB of RAM.

4.1. Frequency assignment relaxation. In this subsection, we demonstrate the effectiveness of SDPAD on the SDP relaxations of frequency assignment problems and compare the results with results obtained using the code SDPNAL [35]. These problems (see equations (4) and (5) on page 363 in [5], or equation (2) on page 5 of [21]) arise from wireless communication networks and contain both equality and inequality constraints. Let $G = (V, E)$ be an undirected graph with vertex set $V = \{1, \cdots, r\}$ 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:

$$
\begin{array}{ll}
\min & \left\langle \frac{1}{2k}\operatorname{diag}(We) + \frac{k-1}{2k}W, X \right\rangle \\
s.t. & X_{i,j} \geq \frac{-1}{k-1}, \quad \forall (i,j) \in E \backslash T, \\
& X_{i,j} = \frac{-1}{k-1}, \quad \forall (i,j) \in T, \\
& \operatorname{diag}(X) = e, \quad X \succeq 0.
\end{array} \tag{4.1}
$$

Using the matrices corresponding to the edges in T and the constraint $\operatorname{diag}(X) = e$ 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 (4.1) as the expanded problem (2.19) 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 (2.23a) and (2.23b) are explicitly available:

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

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.

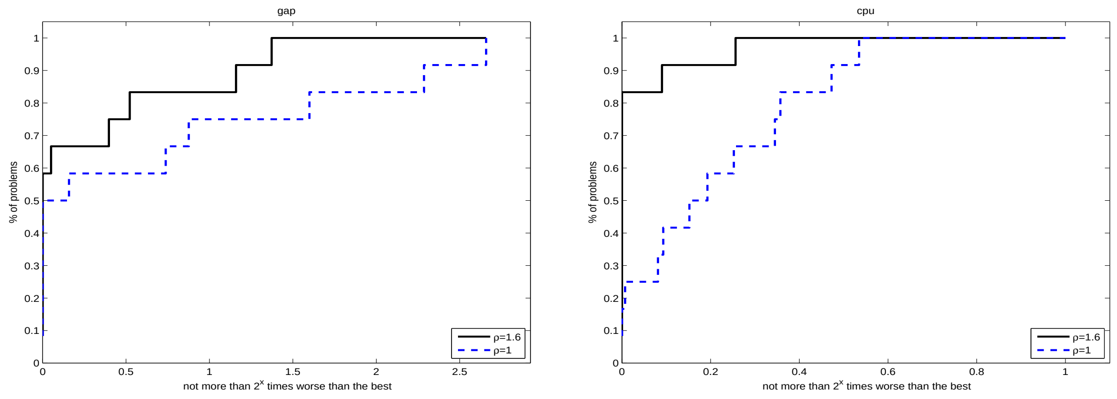
The parameters of SDPNAL were set to their default values. The iteration counter set points h_1, h_2 and h_3 were set to 20, 150 and 300, respectively, the iteration counter h_4 for changing μ was set to 50 and the ratios η_1 and η_2 were set to 1. We stopped SDPAD when the total number of iterations reached 2000. All other parameters were set to their default values. A summary of the computational results is presented in Table 4.1. 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". Since running SDPNAL on "fap25" and "fap36" is very time consuming (for example, in the results reported in [35], 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 [35]. Note that the numerical results of SDPNAL on problems from "fap01" to "fap12" in Table 4.1 are slightly different from those reported in Table 3 in [35]. Since the results in [35] were obtained from a PC with Intel Xeon 3.2GHZ and 4GB 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 [35] are very similar and the comparison between them is meaningful. From these two tables, we can see that SDPAD is faster than SDPNAL for achieving a duality gap of almost the same order. The results of the boundary point method "mprw.m" reported in Table 4 in [35] 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 Figures 4.1(a) and (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.

Table 4.1
Computational results on computing frequency assignment problems

			SDPAD							SDPNAL	
name	n	$\hat{m}$	pobj	dobj	pinf	dinf	itr	gap	cpu	gap	cpu
fap01	52	1378	3.2873028e-2	3.2882685e-2	2.78e-7	3.94e-7	607	9.06e-6	0.35	1.39e-7	6.31
fap02	61	1866	1.0769673e-3	1.0239101e-3	2.18e-6	1.94e-5	666	5.29e-5	0.49	1.15e-5	4.12
fap03	65	2145	4.9412374e-2	4.9410752e-2	5.89e-6	4.30e-6	840	1.48e-6	0.75	2.27e-6	7.44
fap04	81	3321	1.7487540e-1	1.7484595e-1	3.09e-6	2.21e-6	718	2.18e-5	1.21	1.53e-5	19.69
fap05	84	3570	3.0828829e-1	3.0830258e-1	3.49e-6	3.76e-6	768	8.84e-6	1.32	1.09e-5	31.59
fap06	93	4371	4.5936245e-1	4.5937955e-1	5.69e-6	5.72e-6	506	8.91e-6	1.07	1.73e-5	29.84
fap07	98	4851	2.1176348e+	2.1176865e+	4.54e-6	4.95e-6	543	9.89e-6	1.26	5.75e-6	29.88
fap08	120	7260	2.4358398e+	2.4363324e+	1.76e-5	8.74e-6	424	8.39e-5	1.57	5.93e-6	25.25
fap09	174	15225	1.0797819e+1	1.0797814e+1	7.99e-7	9.98e-7	505	2.06e-7	4.77	2.86e-6	59.67
fap10	183	14479	9.2596367e-3	9.7648925e-3	3.22e-6	5.46e-6	1250	4.96e-4	14.93	7.93e-5	1:50
fap11	252	24292	2.9313742e-2	2.9842171e-2	3.26e-6	3.21e-6	1654	4.99e-4	49.57	1.89e-4	5:15
fap12	369	26462	2.7277446e-1	2.7376889e-1	2.59e-6	3.19e-6	2000	6.43e-4	2:34	1.60e-4	13:14
fap25*	2118	322924	1.2863215e+1	1.2880243e+1	1.40e-5	1.63e-5	2000	6.37e-4	7:17:08	1.1e-4	10:53:22
fap36*	4110	1154467	6.9828490e+1	6.9859402e+1	2.03e-5	1.48e-5	2000	2.20e-4	53:14:12	2.5e-5	65:25:07

Fig. 4.1. *Performance profiles of two variants of SDPAD for frequency assignment problems*



(a) gap (b) cpu

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

$$\theta(G) = \max\{\langle C^\top, X\rangle, \quad X_{ij} = 0, \quad (i,j) \in E, \quad \langle I, X\rangle = 1, \quad X \succeq 0\}, \tag{4.2}$$

$$\theta_+(G) = \max\{\langle C^\top, X\rangle, \quad X_{ij} = 0, \quad (i,j) \in E, \quad \langle I, X\rangle = 1, \quad X \succeq 0, \quad X \geq 0\}, \tag{4.3}$$

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 [17, 23, 25]. 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 1000. All other parameters were set to their default values. Summaries of the computational results for $\theta(G)$ and $\theta_+(G)$ are presented in Tables 4.2 and 4.3, respectively. In these tables, the duality "gap" was measured in the original scale, but "pinf" and "dinf" were computed for the scaled problems. Since running SDPNAL on all the test problems is very time consuming (for example, in the results reported in [35], 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.

Table 4.2
Computational results on computing $\theta(G)$

			SDPAD							SDPNAL	
name	n	$\hat{m}$	pobj	dobj	pinf	dinf	itr	gap	cpu	gap	cpu
theta102	500	37467	-3.8390626e+1	-3.8390551e+1	8.76e-7	6.27e-7	256	9.67e-7	47	1.6e-8	50
theta103	500	62516	-2.2528588e+1	-2.2528572e+1	2.85e-7	9.42e-7	257	3.33e-7	57	4.6e-8	1:00
theta104	500	87245	-1.3336159e+1	-1.3336141e+1	3.40e-7	9.72e-7	260	6.53e-7	48	7.6e-8	58
theta123	600	90020	-2.4668678e+1	-2.4668655e+1	3.42e-7	9.71e-7	263	4.57e-7	1:43	4.1e-8	1:34
MANN-a27	378	703	-1.3276623e+2	-1.3276388e+2	8.45e-7	3.50e-6	503	8.81e-6	27	8.3e-8	07
sanr200-0.7	200	6033	-2.3836177e+1	-2.3836162e+1	7.83e-7	9.98e-7	219	3.04e-7	04	1.4e-7	04
c-fat200-1	200	18367	-1.2000003e+1	-1.1999980e+1	1.00e-6	1.76e-7	302	9.15e-7	04	8.5e-8	09
ham-10-2	1024	23041	-1.0239734e+2	-1.0239930e+2	4.72e-7	3.16e-6	597	9.51e-6	22:8	9.0e-8	02
ham-8-3-4	256	16129	-2.5599950e+1	-2.5599909e+1	9.92e-8	9.94e-7	199	8.04e-7	05	1.3e-8	10
ham-9-5-6	512	53761	-8.5332165e+1	-8.5334776e+1	5.70e-7	4.51e-6	1000	1.52e-5	2:48	1.4e-6	1:33
brock400-1	400	20078	-3.9701971e+1	-3.9701904e+1	9.75e-7	8.06e-7	254	8.30e-7	25	1.7e-8	26
keller4	171	5101	-1.4012231e+1	-1.4012258e+1	5.06e-7	9.88e-7	249	9.32e-7	03	1.3e-8	05
p-hat300-1	300	33918	-1.0067984e+1	-1.0067963e+1	7.58e-7	6.81e-7	764	9.96e-7	37	5.3e-7	1:45
G43	1000	9991	-2.8063120e+2	-2.8062688e+2	2.84e-6	3.91e-6	935	7.68e-6	21:17	4.2e-8	1:33
G44	1000	9991	-2.8058951e+2	-2.8058568e+2	3.65e-6	4.21e-6	933	6.82e-6	21:02	3.3e-7	2:59
G45	1000	9991	-2.8017918e+2	-2.8018294e+2	4.02e-6	3.88e-6	957	6.70e-6	21:27	5.6e-8	2:51
G46	1000	9991	-2.7984557e+2	-2.7984014e+2	3.18e-6	5.46e-6	927	9.69e-6	21:02	2.3e-7	2:53
G47	1000	9991	-2.8190252e+2	-2.8189748e+2	4.86e-6	6.01e-6	880	8.91e-6	19:41	1.3e-7	2:54
2dc.512	512	54896	-1.1777815e+1	-1.1770773e+1	2.24e-5	2.57e-5	1000	2.87e-4	5:51	1.7e-4	32:16
1dc.1024	1024	24064	-9.6053190e+1	-9.5999280e+1	7.82e-5	5.10e-5	747	2.79e-4	24:26	2.9e-6	41:26
1et.1024	1024	9601	-1.8460646e+2	-1.8434339e+2	1.93e-4	1.32e-4	603	7.11e-4	20:03	1.8e-6	1:01:14
1tc.1024	1024	7937	-2.0705051e+2	-2.0663863e+2	4.16e-4	5.21e-4	611	9.93e-4	21:47	2.2e-6	1:48:04
1zc.1024	1024	16641	-1.2866647e+2	-1.2866658e+2	9.72e-7	3.78e-7	608	4.16e-7	23:15	3.3e-8	4:15
2dc.1024	1024	169163	-1.8654205e+1	-1.8641209e+1	1.85e-5	2.71e-5	1000	3.39e-4	49:05	9.9e-5	2:57:56
1dc.2048	2048	58368	-1.7527338e+2	-1.7492237e+2	2.82e-4	2.16e-4	473	9.99e-4	2:50:44	1.5e-6	6:11:11
1et.2048	2048	22529	-3.4297575e+2	-3.4229432e+2	1.78e-4	4.05e-4	904	9.93e-4	4:54:57	8.8e-7	7:13:55
1tc.2048	2048	18945	-3.7567281e+2	-3.7486271e+2	1.79e-4	4.48e-4	1000	1.08e-3	5:15:14	7.9e-6	9:52:09
1zc.2048	2048	39425	-2.3739409e+2	-2.3739845e+2	1.42e-6	3.10e-6	941	9.17e-6	6:39:07	1.2e-6	45:16
2dc.2048	2048	504452	-3.0698999e+1	-3.0679246e+1	1.88e-5	8.37e-6	1000	3.17e-4	7:12:50	4.4e-5	15:13:19

Hence, the SDPNAL results in Tables 4.2 and 4.3 are taken from Tables 5 and 6 in [35]. Because the computer used to obtain the results in [35] has very similar performance characteristics to the computer that we used, the comparison presented in Tables 4.2 and 4.3 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 [35] by an insignificant amount. From Table 4.2, we can see that SDPAD achieves approximately the same level of duality gap as SDPNAL on problems like "theta102" to "theta123", "c-fat200-1" and "brock400-1". Although SDPNAL is faster than SDPAD on problems like "hamming-10-2" and "G43" to "G47", SDPAD is faster than SDPNAL on "2dc.512". From Table 4.3, we can see that SDPAD is faster than SDPNAL on most problems except "hamming-9-5-6", "hamming-10-2", "1zc.1024" and "1zc.2048" while achieving almost the same level of duality gap. Finally, performance plots for numerical results obtained using $\rho = 1$ and $\rho = 1.6$ for computing $\theta(G)$ and $\theta_+(G)$ are presented in Figures 4.2(a) and (b), and Figures 4.3(a) and (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 $\rho = 1$.

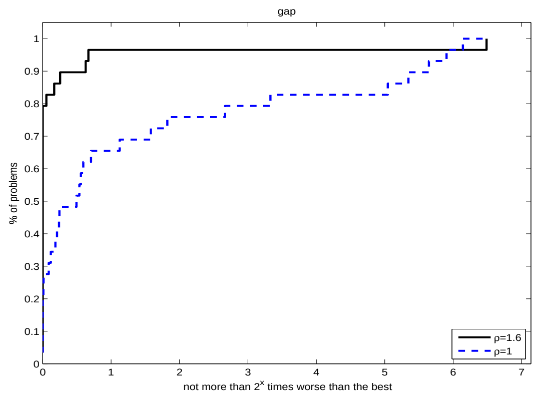
4.3. Binary Integer Quadratic Programming Problem. In this subsection, we report on how SDPAD performs on SDP relaxations of binary integer quadratic programming problems and compare these results to those obtained using SDPNAL. These problems have the form:

$$\begin{aligned} \min \quad & \left\langle \begin{pmatrix} Q & 0 \\ 0 & 0 \end{pmatrix}, X \right\rangle \\ s.t. \quad & X_{ii} - X_{n,i} = 0, i = 1, \cdots, n-1, \\ & X_{nn} = 1, \quad X \succeq 0, \quad X \geq 0, \end{aligned} \tag{4.4}$$

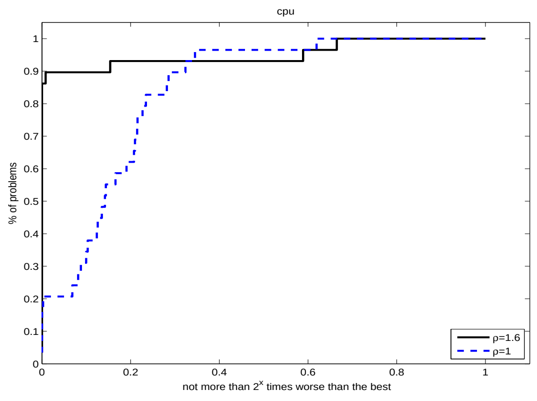
TABLE 4.3
Computational results on computing $\theta_+(G)$

			SDPAD							SDPNAL	
name	n	$\hat{m}$	pobj	dobj	pinf	dinf	itr	gap	cpu	gap	cpu
theta102	500	37467	-3.8066274e+1	-3.8066252e+1	2.94e-7	9.58e-7	281	2.84e-7	1:01	8.4e-8	3:31
theta103	500	62516	-2.2377445e+1	-2.2377422e+1	3.27e-7	9.52e-7	262	4.85e-7	1:01	2.3e-8	3:28
theta104	500	87245	-1.3282631e+1	-1.3282610e+1	3.57e-7	9.55e-7	266	7.35e-7	52	1.6e-7	2:35
theta123	600	90020	-2.4495182e+1	-2.4495152e+1	3.77e-7	9.43e-7	267	6.02e-7	1:39	1.2e-7	6:44
MANN-a27	378	703	-1.3275956e+2	-1.3276174e+2	8.98e-7	4.08e-6	530	8.17e-6	32	1.6e-7	35
sanr200-0.7	200	6033	-2.3633314e+1	-2.3633293e+1	8.61e-7	9.57e-7	228	4.47e-7	05	2.9e-7	11
c-fat200-1	200	18367	-1.2000012e+1	-1.1999991e+1	4.71e-7	2.04e-7	306	8.40e-7	04	2.1e-7	36
ham-8-3-4	256	16129	-2.5599951e+1	-2.5599909e+1	9.91e-8	9.94e-7	199	8.05e-7	05	2.7e-10	05
ham-9-5-6	512	53761	-5.8666560e+1	-5.8666522e+1	6.49e-8	7.58e-7	472	3.25e-7	2:19	2.6e-7	42
ham-10-2	1024	23041	-8.5333069e+1	-8.5333237e+1	4.99e-8	5.49e-7	653	9.78e-7	27:58	4.2e-7	4:35
brock400-1	400	20078	-3.9331005e+1	-3.9330926e+1	9.98e-7	7.54e-7	258	9.83e-7	29	3.5e-9	1:45
keller4	171	5101	-1.3466089e+1	-1.3466006e+1	3.61e-6	9.01e-6	331	2.98e-6	05	3.7e-7	43
p-hat300-1	300	33918	-1.0020244e+1	-1.0020212e+1	1.34e-6	5.93e-7	567	1.50e-6	29	7.9e-7	6:50
G43	1000	9991	-2.7972840e+2	-2.7973289e+2	4.40e-6	5.45e-6	864	8.02e-6	20:22	2.1e-7	52:00
G44	1000	9991	-2.7975221e+2	-2.7974864e+2	4.95e-6	4.36e-6	893	6.36e-6	21:14	5.7e-8	49:32
G45	1000	9991	-2.7931027e+2	-2.7931528e+2	5.11e-6	4.08e-6	916	8.96e-6	22:54	2.4e-8	50:25
G46	1000	9991	-2.7904079e+2	-2.7903549e+2	4.34e-6	5.19e-6	886	9.47e-6	22:48	3.3e-8	44:38
G47	1000	9991	-2.8089994e+2	-2.8089501e+2	6.05e-6	5.66e-6	838	8.76e-6	21:11	5.1e-9	40:27
2dc.512	512	54896	-1.1385823e+1	-1.1383769e+1	9.43e-6	9.09e-6	1000	8.64e-5	5:10	3.8e-4	2:25:15
1dc.1024	1024	24064	-9.5563960e+1	-9.5552471e+1	2.13e-5	8.50e-6	1000	5.98e-5	57:20	1.4e-5	5:03:49
1et.1024	1024	9601	-1.8229378e+2	-1.8210159e+2	1.18e-4	2.16e-4	651	5.26e-4	36:04	1.1e-5	6:45:50
1tc.1024	1024	7937	-2.0440122e+2	-2.0425679e+2	1.65e-4	2.67e-4	799	3.53e-4	49:13	8.7e-4	10:37:57
1zc.1024	1024	16641	-1.2813904e+2	-1.2800286e+2	5.71e-5	1.80e-5	506	5.30e-4	28:22	1.6e-7	40:13
2dc.1024	1024	169163	-1.7711337e+1	-1.7709936e+1	3.81e-6	2.37e-6	1000	3.85e-5	50:34	7.3e-4	11:57:25
1dc.2048	2048	58368	-1.7477173e+2	-1.7442175e+2	2.01e-4	9.45e-5	517	9.99e-4	4:25:17	9.7e-5	35:52:44
1et.2048	2048	22529	-3.3883524e+2	-3.3837436e+2	1.76e-4	2.66e-4	659	6.80e-4	5:15:09	4.0e-5	80:48:17
1tc.2048	2048	18945	-3.7115486e+2	-3.7070213e+2	1.86e-4	3.97e-4	862	6.09e-4	6:35:33	1.4e-3	73:56:01
1zc.2048	2048	39425	-2.3739370e+2	-2.3739764e+2	3.58e-7	4.60e-6	953	8.27e-6	7:14:21	2.3e-7	2:13:04
2dc.2048	2048	504452	-2.8789604e+1	-2.8786706e+1	4.84e-6	2.79e-6	1000	4.95e-5	5:45:25	2.7e-3	45:21:42

FIG. 4.2. *Performance profiles of two variants of SDPAD for computing $\theta(G)$*



(a) gap



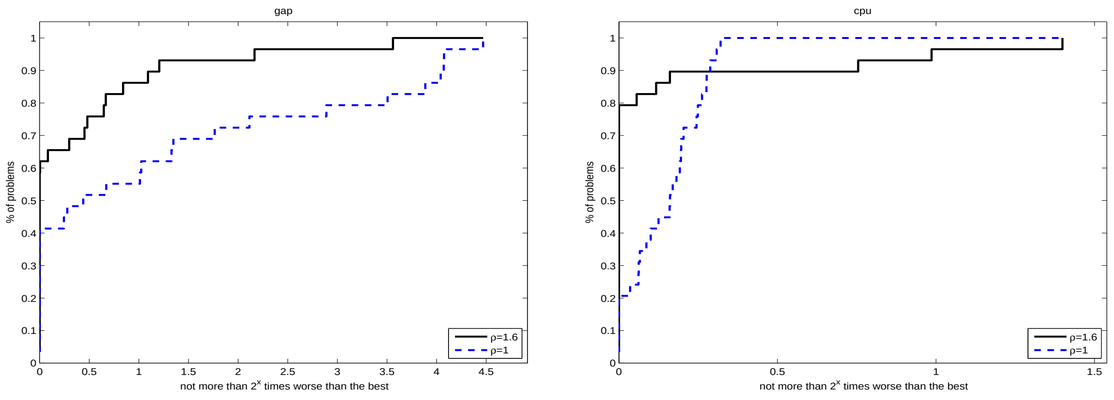
(b) cpu

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 [30] are presented in Table 4.4, where "best upper bound" is the best known upper bound reported in [30], and "%pgap" and "%dgap" were computed as

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

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

Fig. 4.3. *Performance profiles of two variants of SDPAD for computing $\theta_+(G)$*



(a) gap (b) cpu

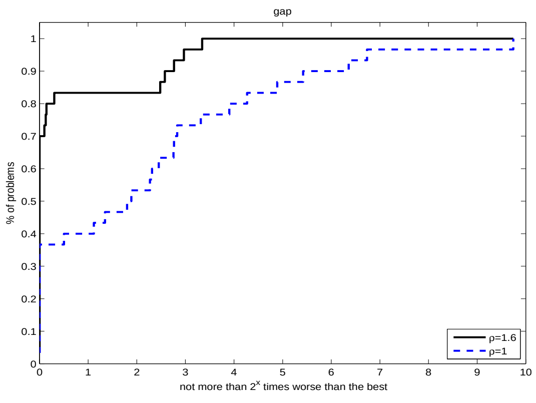
Table 4.4
Computational results on the BIQ problems

		SDPAD								SDPNAL	
name	n	pinf	dinf	gap	itr	best upper bound	%pgap	%dgap	cpu	%dgap	cpu
be200.3.1	201	7.09e-7	5.17e-7	9.99e-7	2484	-2.5453000e+4	8.891	8.891	32	8.891	10:29
be200.3.3	201	8.73e-6	1.42e-6	6.43e-6	2296	-2.8023000e+4	5.194	5.195	31	5.192	12:09
be200.3.5	201	2.66e-6	3.45e-8	3.55e-8	4782	-2.6355000e+4	6.519	6.519	1:04	6.519	10:38
be200.3.7	201	8.88e-6	1.78e-6	9.97e-6	2447	-3.0483000e+4	3.730	3.732	32	3.730	9:43
be200.3.9	201	1.00e-6	1.75e-8	1.50e-8	6940	-2.4683000e+4	7.106	7.106	1:31	7.106	8:28
be200.8.1	201	1.93e-6	3.06e-8	2.77e-8	4267	-4.8534000e+4	4.812	4.812	57	4.811	9:41
be200.8.3	201	8.79e-6	1.03e-6	1.00e-5	2107	-4.3207000e+4	7.051	7.053	28	7.052	10:53
be200.8.5	201	8.60e-6	1.48e-6	9.99e-6	1885	-4.1482000e+4	6.725	6.723	25	6.723	9:53
be200.8.7	201	8.59e-6	4.73e-7	9.97e-6	2186	-4.6828000e+4	5.394	5.391	28	5.392	4:30
be200.8.9	201	5.57e-6	7.95e-7	4.60e-6	2146	-4.3241000e+4	5.213	5.214	29	5.213	12:16
be250.1	251	9.99e-7	3.10e-8	8.74e-9	5923	-2.4076000e+4	4.334	4.334	2:14	4.332	16:41
be250.3	251	6.12e-6	4.58e-6	7.82e-6	2747	-2.2923000e+4	4.698	4.697	1:02	4.698	17:17
be250.5	251	9.98e-7	2.21e-8	1.03e-8	6326	-2.1057000e+4	6.258	6.258	2:24	6.254	14:30
be250.7	251	2.30e-6	5.18e-8	7.88e-9	5256	-2.4095000e+4	4.250	4.250	1:57	4.250	14:00
be250.9	251	1.00e-6	5.61e-8	8.25e-9	6532	-2.0051000e+4	6.713	6.713	2:30	6.713	17:13
bqp250-1	251	6.41e-6	2.57e-6	9.97e-6	3005	-4.5607000e+4	4.509	4.507	1:08	4.508	17:42
bqp250-3	251	3.18e-6	2.06e-6	7.68e-7	3006	-4.9037000e+4	4.159	4.159	1:05	4.160	10:36
bqp250-5	251	7.80e-6	1.10e-6	9.08e-6	3129	-4.7961000e+4	4.260	4.261	1:10	4.260	19:03
bqp250-7	251	5.63e-7	3.21e-7	9.98e-7	4801	-4.6757000e+4	4.630	4.630	1:46	4.630	16:36
bqp250-9	251	6.64e-6	3.47e-6	9.24e-6	3006	-4.8916000e+4	5.277	5.279	1:06	5.276	16:12
bqp500-1	501	2.36e-7	3.92e-7	1.00e-6	8960	-1.1658600e+5	8.044	8.044	19:44	8.045	1:00:59
bqp500-3	501	2.42e-7	3.61e-7	1.00e-6	8824	-1.3081200e+5	5.842	5.841	19:04	5.842	1:01:47
bqp500-5	501	4.44e-7	4.03e-7	9.99e-7	8288	-1.2548700e+5	6.857	6.857	18:41	6.857	1:36:43
bqp500-7	501	2.75e-7	4.83e-7	9.99e-7	9153	-1.2220100e+5	7.603	7.602	20:16	7.603	1:25:26
bqp500-9	501	2.82e-7	4.88e-7	1.00e-6	8439	-1.2079800e+5	7.856	7.856	18:54	7.857	1:24:40
gka2e	201	9.99e-7	1.65e-8	2.07e-8	4375	-2.3395000e+4	6.508	6.508	57	6.506	7:23
gka4e	201	8.35e-6	2.75e-7	3.42e-6	2735	-3.5594000e+4	4.583	4.582	36	4.582	11:25
gka1f	501	4.43e-7	5.83e-7	9.99e-7	8106	-6.1194000e+4	7.133	7.133	17:57	7.133	1:28:54
gka3f	501	3.43e-7	5.27e-7	9.99e-7	7785	-1.3803500e+5	8.778	8.777	17:04	8.778	1:31:34
gka5f	501	3.78e-7	7.40e-7	1.00e-6	8849	-1.9050700e+5	8.612	8.612	18:58	8.613	1:25:48

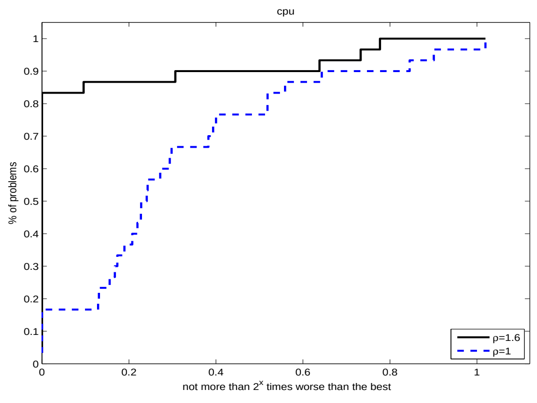
number of iterations reached 10000. The minimum penalty parameter $\mu_{\min}$ is set to 0.1. All other parameters were set to their default values. Again, we did not run SDPNAL on our own computer but presented the results reported in Table 8 in [35] in Table 4.4. From this table, we can see that SDPAD is faster than SDPNAL for achieving comparable lower bounds. Finally, performance plots for numerical results obtained using $\rho = 1$ and $\rho = 1.6$ are presented in Figures 4.4(a) and (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 $\rho = 1$.

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

Fig. 4.4. *Performance profiles of two variants of SDPAD for the BIQ problems*



(a) gap



(b) cpu

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 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. Since the low rank structure of the optimal solution is often exposed after relatively few iterations from our numerical experience, we plan to explore how to take advantage of this to improve the efficiency of our algorithm.

Acknowledgements. We want to thank Kim-Chuan Toh and Defeng Sun for sharing their code SDP-NAL and for providing the data files for the frequency assignment and maximum stable set problems. We also want to thank Samuel Burer for sharing his code SDPLR and Andreas Stathopoulos for his helpful comments on iterative methods for eigenvalue decomposition.

REFERENCES

[1] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

[2] S. Burer, *Optimizing a polyhedral-semidefinite relaxation of completely positive programs*, manuscript, Department of Management Sciences, University of Iowa, Iowa City, IA 52242-1994, USA, December 2008. Submitted to *Mathematical Programming Computation*.

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

[4] ———, *Local minima and convergence in low-rank semidefinite programming*, Math. Program., 103 (2005), pp. 427–444.

[5] S. Burer, R. D. C. Monteiro, and Y. Zhang, *A computational study of a gradient-based log-barrier algorithm for a class of large-scale SDPs*, Math. Program., 95 (2003), pp. 359–379. Computational semidefinite and second order cone programming: the state of the art.

[6] S. Burer and D. Vandenbussche, *Solving lift-and-project relaxations of binary integer programs*, SIAM J. Optim., 16 (2006), pp. 726–750 (electronic).

[7] G. Chen and M. Teboulle, *A proximal-based decomposition method for convex minimization problems*, Math. Programming, 64 (1994), pp. 81–101.

[8] E. D. Dolan and J. J. Moré, *Benchmarking optimization software with performance profiles*, Math. Program., 91 (2002), pp. 201–213.

[9] J. Eckstein and D. P. Bertsekas, *An alternating direction method for linear programming.* LIDS-P, 1967. Cambridge,

MA, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology.

[10] J. Eckstein and D. P. Bertsekas, *On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators*, Math. Programming, 55 (1992), pp. 293–318.

[11] M. Fortin and R. Glowinski, *Augmented Lagrangian methods*, vol. 15 of Studies in Mathematics and its Applications, North-Holland Publishing Co., Amsterdam, 1983. Applications to the numerical solution of boundary value problems, Translated from the French by B. Hunt and D. C. Spicer.

[12] R. Glowinski and P. Le Tallec, *Augmented Lagrangian and operator-splitting methods in nonlinear mechanics*, vol. 9 of SIAM Studies in Applied Mathematics, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1989.

[13] E. T. Hale, W. Yin, and Y. Zhang, *Fixed-point continuation for l_1-minimization: methodology and convergence*, SIAM J. Optim., 19 (2008), pp. 1107–1130.

[14] B. He, L.-Z. Liao, D. Han, and H. Yang, *A new inexact alternating directions method for monotone variational inequalities*, Math. Program., 92 (2002), pp. 103–118.

[15] B. S. He, H. Yang, and S. L. Wang, *Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities*, J. Optim. Theory Appl., 106 (2000), pp. 337–356.

[16] J.-B. Hiriart-Urruty and C. Lemaréchal, *Convex analysis and minimization algorithms. I*, vol. 305 of Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences], Springer-Verlag, Berlin, 1993. Fundamentals.

[17] D. S. Johnson and M. A. Trick, eds., *Cliques, coloring, and satisfiability*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 26, American Mathematical Society, Providence, RI, 1996. Papers from the workshop held as part of the 2nd DIMACS Implementation Challenge in New Brunswick, NJ, October 11–13, 1993.

[18] K. C. Kiwiel, C. H. Rosa, and A. Ruszczyński, *Proximal decomposition via alternating linearization*, SIAM J. Optim., 9 (1999), pp. 668–689.

[19] S. Kontogiorgis and R. R. Meyer, *A variable-penalty alternating directions method for convex optimization*, Math. Programming, 83 (1998), pp. 29–53.

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

[21] G. Pataki and S. Schmieta, *The dimacs library of semidefinite-quadratic-linear programs*, tech. rep., Center, Columbia University, 1999.

[22] J. Povh, F. Rendl, and A. Wiegele, *A boundary point method to solve semidefinite programs*, Computing, 78 (2006), pp. 277–286.

[23] N. J. A. Sloane, *Challenge problems: Independent sets in graphs*. http://research.att.com/ njas/doc/graphs.html.

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

[25] K.-C. Toh, *Solving large scale semidefinite programs via an iterative solver on the augmented systems*, SIAM J. Optim., 14 (2003), pp. 670–698 (electronic).

[26] P. Tseng, *Alternating projection-proximal methods for convex programming and variational inequalities*, SIAM J. Optim., 7 (1997), pp. 951–965.

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

[28] Y. Wang, J. Yang, W. Yin, and Y. Zhang, *A new alternating minimization algorithm for total variation image reconstruction*, SIAM J. Imaging Sci., 1 (2008), pp. 248–272.

[29] Z. Wen, D. Goldfarb, S. Ma, and K. Scheinberg, *Row by row methods for semidefinite programming*, tech. rep., Dept of IEOR, Columbia University, 2009.

[30] A. Wiegele, *Biq mac library - a collection of max-cut and quadratic 0-1 programming instances of medium size*, tech. rep., 2007.

[31] H. Wolkowicz, R. Saigal, and L. Vandenberghe, eds., *Handbook of semidefinite programming*, International Series in Operations Research & Management Science, 27, Kluwer Academic Publishers, Boston, MA, 2000. Theory, algorithms, and applications.

[32] J. Yang, Y. Zhang, and W. Yin, *An efficient tvl1 algorithm for deblurring multichannel images corrupted by impulsive noise*, tech. rep., Rice University, 2008.

[33] Z. Yu, *Solving semidefinite programming problems via alternating direction methods*, J. Comput. Appl. Math., 193 (2006), pp. 437–445.

[34] Y. Zhang, *User's guide for yall1: Your algorithms for l1 optimization*, tech. rep., Rice University, 2009.

[35] X. Zhao, D. Sun, and K. Toh, *A newton-cg augmented lagrangian method for semidefinite programming*, tech. rep., Department of Mathematics, National University of Singapore, 2008.