Using a Factored Dual in Augmented Lagrangian Methods for Semidefinite Programming

Marianna De Santis\textsuperscript{a}, Franz Rendl\textsuperscript{b}, Angelika Wiegele\textsuperscript{b}

\textsuperscript{a}Dipartimento di Ingegneria Informatica Automatica e Gestionale, Sapienza Università di Roma, Via Ariosto, 25, 00185 Roma, Italy
\textsuperscript{b}Institut für Mathematik, Alpen-Adria-Universität Klagenfurt, Universitätsstraße 65-67, 9020 Klagenfurt, Austria

Abstract

In the context of augmented Lagrangian approaches for solving semidefinite programming problems, we investigate the possibility of eliminating the positive semidefinite constraint on the dual matrix by employing a factorization. Hints on how to deal with the resulting unconstrained maximization of the augmented Lagrangian are given. We further use the approximate maximum of the augmented Lagrangian with the aim of improving the convergence rate of alternating direction augmented Lagrangian frameworks. Numerical results are reported, showing the benefits of the approach.

Keywords: Semidefinite Programming, Alternating Direction Augmented Lagrangian method, theta function

2010 MSC: 90C22, 90C30, 90C06

1. Introduction

Semidefinite Programs (SDP) can be solved in polynomial time to some fixed prescribed precision, but the computational effort grows both with the number \( m \) of constraints and with the order \( n \) of the underlying space of symmetric matrices. Interior point methods to solve SDP become impractical both in terms of computation time and memory requirements, once \( m \geq 10^4 \). Several algorithmic alternatives have been introduced in the literature, including some based on augmented Lagrangian approaches \[ \text{ADAL} \]. It is the purpose of this paper to elaborate on the alternating direction augmented Lagrangian (ADAL) algorithms proposed in \[ \text{ADAL} \] by introducing computational refinements. The key idea will be to eliminate the positive semidefinite constraint on the dual matrix by employing a factorization, so that the maximization of the augmented Lagrangian function with respect to the dual variables can be performed in an unconstrained fashion.

In the remainder of this section we give the problem formulation and state our notations. In Section 2 a description of the ADAL methods for solving semidefinite programs is given. Details on how we maximize the augmented Lagrangian, after the factorization of the dual matrix, are given in Section 3. In Section 4 we outline our new algorithm DADAL: an additional update of the dual variable within one iteration of the ADAL method is used as improvement step. The convergence of DADAL easily follows by the analysis done in 5. DADAL as a fixed point method. We give insights on how DADAL can improve the convergence rate of ADAL in Section 4.1. Section 5 shows numerical results and Section 6 concludes.

1.1. Problem Formulation and Notations

Let \( S_n \) be the set of \( n \)-by-\( n \) symmetric matrices and \( S_n^+ \subset S_n \) be the set of positive semidefinite matrices. Denoting by \( \langle X, Y \rangle = \text{trace}(XY) \) the standard inner product in \( S_n \), we write the standard primal-dual pair of SDP problems as follows:

\[
\begin{align*}
\min \quad & \langle C, X \rangle \\
\text{s.t.} \quad & AX = b, \quad X \in S_n^+ 
\end{align*}
\]

and

\[
\begin{align*}
\max \quad & b^T y \\
\text{s.t.} \quad & C - A^T y = Z \\
& Z \in S_n^+. 
\end{align*}
\]

where \( C \in S_n, b \in \mathbb{R}^m, A : S_n \to \mathbb{R}^m \) is the linear operator \( \langle AX \rangle = \langle A_i, X \rangle \) with \( A_i \in S_n, i = 1, \ldots, m \) and \( A^T : \mathbb{R}^m \to S_n \) is its adjoint, \( A^T y = \sum_i y_i A_i \).

We assume that both problems have strictly feasible points, so that strong duality holds. Under this assumption, \( (X, y, Z) \) is optimal if and only if

\[
X \in S_n^+, \quad AX = b, \quad Z \in S_n^+, \quad C - A^T y = Z, \quad ZX = 0. \tag{3}
\]

We further assume that matrix \( A \) has full rank.

Let \( v \in \mathbb{R}^n \) and \( M \in \mathbb{R}^{mn} \). In the following, we denote by \( \text{vec}(M) \) the \( mn \)-dimensional vector formed by stacking the columns of \( M \) on top of each other (\( \text{vec}^{-1} \) is the inverse operation). We also denote by \( \text{Diag}(v) \) the diagonal matrix having \( v \) in the diagonal. With \( e_i \) we denote the \( i \)-th vector of the standard basis in \( \mathbb{R}^n \). Whenever a norm is used, we consider the Frobenius norm in case of matrices and the Euclidean norm in case of vectors. We denote the projection of some symmetric matrix \( S \) onto the positive semidefinite cone by \( (S)_+ \), and its projection onto the negative semidefinite cone by \( (S)_- \).
2. Augmented Lagrangian Methods for SDP

Let \( X \in S_n \) be the Lagrange multiplier for the dual equation \( Z + C + A^T y = 0 \). In order to solve Problem (2) with the augmented Lagrangian method we introduce

\[
L_\sigma(y, Z; X) := b^T y - (Z + C + A^T y, X) - \frac{\sigma}{2} ||Z + C + A^T y||^2.
\]

To solve (2), we deal with

\[
\max_{y \in \mathbb{R}^m} \ L_\sigma(y, Z; X) \quad \text{s.t.} \quad Z \in S_+^n,
\]

where \( X \) is fixed and \( \sigma > 0 \) is the penalty parameter.

Once Problem (3) is (approximately) solved, the multiplier \( X \) is updated by a first order rule:

\[
X = X + \sigma(Z - C + A^T y)
\]

and the process is iterated until convergence, i.e. until the optimality conditions (5) are satisfied within a certain tolerance (see Chapter 2 in [4] for further details).

Problem (3) is a convex quadratic semidefinite optimization problem, which is tractable but expensive to solve directly. Several simplified versions have been proposed in the literature to quickly get approximate solutions.

In the alternating direction framework proposed by Wen et al. [5], the augmented Lagrangian \( L_\sigma(y, Z; X) \) is maximized with respect to \( y \) and \( Z \) one after the other. More precisely, at every iteration \( k \), the new point \((X^{k+1}, y^{k+1}, Z^{k+1})\) is computed by the following steps:

\[
y^{k+1} := \arg \max_{y \in \mathbb{R}^m} L_\sigma(y, Z^k; X^k),
\]

\[
Z^{k+1} := \max_{Z \in S_+^n} L_\sigma(y^{k+1}, Z; X^k),
\]

\[
X^{k+1} := X^k + \sigma(Z^{k+1} - C + A^T y^{k+1}).
\]

These three steps are iterated until a stopping criterion is met. The update of \( y \) in (6) is derived from the first-order optimality condition of Problem (4):

\[
\nabla_y L_\sigma(y, Z^k; X^k) = b - A(X^k + \sigma Z^{k+1} - C + A^T y^k) = 0,
\]

that is

\[
y^{k+1} = (A^T A)^{-1} \left\{ \frac{1}{\sigma} b - A \left( \frac{1}{\sigma Z^{k+1}} X^k - C + Z^k \right) \right\}.
\]

Then, the maximization in (7) is conducted by considering the equivalent problem

\[
\min_{Z \in S_+^n} ||Z + W^k||^2,
\]

with \( W^k = \left( \frac{1}{\sigma} X^k - C + A^T y^{k+1} \right) \), or, in other words, by projecting \( W^k \) onto the (closed convex) cone \( S_+^n \) and taking its additive inverse (see Algorithm 1). Such a projection is computed via the spectral decomposition of the matrix \( W^k \).

The Boundary Point Method proposed in [4] can also be viewed as an alternating direction augmented Lagrangian method. In fact, as also noted in [4], the available implementation \texttt{mprw.m} (see https://www.math.aau.at/or/Software/) is an alternating direction augmented Lagrangian method, since in the inner loop (see Table 2 in [4]) only one iteration is performed.

We report in Algorithm 1 the scheme of \texttt{mprw.m}. The stopping criterion for \texttt{mprw.m} considers only the following primal and dual infeasibility errors:

\[
r_p = \frac{||A X - b||}{1 + ||b||},
\]

\[
r_d = \frac{||C - Z - A^T y||}{1 + ||C||};
\]

as the other optimality conditions (namely, \( X \in S_+^n, Z \in S_+^n, ZX = 0 \)) are satisfied up to machine accuracy throughout the algorithm. More precisely, the algorithm stops as soon as the quantity

\[
\delta = \max\{r_p, r_d\}
\]

is less than a fixed precision \( \varepsilon > 0 \).

Algorithm 1 Scheme of \texttt{mprw.m}

1. **Initialization**: Choose \( \sigma > 0, X \in S_+^n, \varepsilon > 0 \). Set \( Z = 0 \).
2. **Repeat until** \( \delta < \varepsilon \):
3. Compute \( y = (A A^T)^{-1} \left( \frac{1}{\sigma} b - A \left( \frac{1}{\sigma Z} X - C + Z \right) \right) \)
4. Compute \( Z = -(X/\sigma - C + A^T y)_+ \)
5. Compute \( \delta = \max\{r_p, r_d\} \)
6. **Update** \( \sigma \)

It is the main purpose of this paper to investigate enhancements to this algorithm. The key idea will be to replace in the subproblem (3) the constraint \( Z \in S_+^n \) by \( Z = VV^T \) and considering (4) as an unconstrained problem in \( y \) and \( V \).

3. Solving the subproblem (4)

By introducing a variable \( V \in \mathbb{R}^{n \times r} (1 \leq r \leq n) \), such that \( Z = VV^T \in S_+^n \), we reformulate Problem (4) as the following unconstrained maximization problem

\[
\max_{y \in \mathbb{R}^m, \ V \in \mathbb{R}^{n \times r}} \ L_\sigma(y, V; X)
\]

where

\[
L_\sigma(y, V; X) = b^T y - (VV^T - C + A^T y, X) - \frac{\sigma}{2} ||VV^T - C + A^T y||^2.
\]

Note that the number of columns \( r \) of matrix \( V \) represents the rank of the dual variable \( Z \).

The first-order necessary optimality conditions for Problem (10) state the following:
Proposition 1. Let \( (y^*, V^*) \in \mathbb{R}^m \times \mathbb{R}^{nxr} \) be a stationary point for Problem \( P \), then

\[
\nabla_y L_{\sigma}(y^*, V^*; X) = b - A(X + \sigma(V^* V^T - C + A^T y^*)) = 0,
\]

\[
\nabla_V L_{\sigma}(y^*, V^*; X) = -2(X + \sigma(V^* V^T - C + A^T y^*))V^* = 0.
\]

(11)

From (11), we can easily see that we can keep the optimality conditions with respect to \( y \) satisfied, while moving \( V \) along any direction \( D_V \in \mathbb{R}^{nxr} \).

Proposition 2. Let \( D_V \in \mathbb{R}^{nxr} \). Let

\[
y(V + a D_V) = y_0 + a y_1 + a^2 y_2,
\]

with

\[
y_0 = (A A^T)^{-1} \left( \frac{1}{\sigma} b - A \left( \frac{1}{\sigma} X - C + V V^T \right) \right),
\]

\[
y_1 = (A A^T)^{-1} \left( -A D_V V^T + V D_V^T \right),
\]

\[
y_2 = (A A^T)^{-1} \left( -A D_V D_V^T \right).
\]

Then

\[
\nabla_y L_{\sigma}(y(V + a D_V), V + a D_V; X) = 0,
\]

for all \( a \in \mathbb{R} \).

Proof. Let \( a \in \mathbb{R} \). From (11), we have that \( \nabla_y L_{\sigma}(y, V; X) = 0 \) iff

\[
A A^T y = \left( \frac{1}{\sigma} b - A \left( \frac{1}{\sigma} X - C + V V^T \right) \right)
\]

Therefore, when \( V = V + a D_V \), we get

\[
(A A^T) y = \frac{1}{\sigma} b - A \left( \frac{1}{\sigma} X - C + (V + a D_V)(V + a D_V)^T \right)
\]

\[
= \frac{1}{\sigma} b - A \left( \frac{1}{\sigma} X - C + V V^T \right)
\]

\[
- a A (D_V V^T + V D_V^T) - a^2 A (D_V D_V^T).
\]

By multiplying both the l.h.s. and the r.h.s. with \( (A A^T)^{-1} \), we get the expression in (12) and the proposition is proven. \( \square \)

Thanks to Proposition 2, we can maximize \( L_{\sigma}(y, V; X) \) with respect to \( V \), keeping variable \( y \) updated according to (12) along the iterations. Thus we are in fact maximizing a polynomial of degree 4 in \( V \).

3.1. Direction Computation

In order to compute an ascent direction for the augmented Lagrangian we consider two possibilities. Either we use the gradient of \( L_{\sigma}(y, V; X) \) with respect to \( V \), or we use the gradient scaled with the inverse of the diagonal of the Hessian of \( L_{\sigma}(y, V; X) \). We recall the gradient of \( L_{\sigma} \) with respect to \( V \), see (11), as the \( n \times r \) matrix

\[
\nabla_V L_{\sigma}(y, V; X) = -2(M + \sigma V V^T)V.
\]

where \( M = X + \sigma(A^T y - C) \). In order to compute the generic \((s, t)\) entry on the main diagonal of the Hessian, we consider

\[
\lim_{t \to 0} \frac{1}{t} \left( f_{st}(V + t e_s e_t^T) - f_{st}(V) \right),
\]

where \( f_{st}(V) := e_s^T V V^T V e_t, s \in \{1, \ldots, n\}, t \in \{1, \ldots, r\} \). We get

\[
\frac{\partial^2 L_{\sigma}(y, V; X)}{\partial y_s \partial y_t} = -2m_{st} - 2\sigma \left( (e_s^T V e_t)^2 + \|V^T e_s\|^2 + \|Ve_t\|^2 \right).
\]

We define the \( n \times r \) matrix \( H \) by

\[
(H)_{st} = 2 \max(0, m_{st}) + 2\sigma ((e_s^T V e_t)^2 + \|V^T e_s\|^2 + \|Ve_t\|^2),
\]

and we propose to use the search direction \( D_V \), given by the gradient scaled by \( H \), thus

\[
(D_V)_{st} := \frac{(\nabla_V L_{\sigma})_{st}}{H_{st}}.
\]

We note that \( H \) is generically positive, as \( V \) should not contain columns equal to zero. In practice, we compute the scaled gradient direction in case \( \|\nabla_V L_{\sigma}(y, V; X)\| < 10^{-3} \).

3.2. Exact Linesearch

Given a search direction \( D_V \), we note that \( L_{\sigma}(y(V + a D_V), V + a D_V; X) \) is a polynomial of degree 4 in \( a \), so that we can interpolate five different points

\[
(a_i, L_{\sigma}(y(V + a_i D_V), V + a_i D_V; X)), \quad i = 1, \ldots, 5,
\]

to get its analytical expression. This also means that the maximum of \( L_{\sigma}(y(V + a D_V), V + a D_V; X) \) can be detected analytically (using the Cardano formula). In practice, we evaluate the 4-degree polynomial \( L_{\sigma}(y(V + a D_V), V + a D_V; X) \) in few thousands of points in \((0,10)\) and take the best \( a \).

Algorithm 3 is a scheme of the generic iteration we perform to maximize \( L_{\sigma}(y, V; X) \), \( X \) being fixed.

Algorithm 2 Solving (4) approximately

1. Input: \( \sigma > 0, y \in \mathbb{R}^m, V \in \mathbb{R}^{nxr} \)
2. Repeat until \( \|\nabla_V L_{\sigma}\| < \epsilon_{\text{inner}} \)
3. Compute the search direction \( D_V \in \mathbb{R}^{nxr} \) using (13)
4. Compute optimal steps size \( \alpha \)
5. Update \( y = y(V + a D_V) \) according to (12) and \( V = V + a D_V \).

4. DADAL: a dual step for improving alternating direction augmented Lagrangian methods for SDP

Our idea is to insert the approximated solution of Problem 4 obtained from Algorithm 2 as a simultaneous update of \( y \) and \( V \) to be performed before the projection step in Algorithm 1. We detail below the scheme of the ADAL method where this “dual step” is inserted. We refer to Algorithm 5 as DADAL.
Algorithm 3 DADAL

1. **Initialization:** Choose $\sigma > 0$, $r > 0$, $\varepsilon > 0$, $X \in S_n$, $V \in \mathbb{R}^{n \times n}$, $Z = VV^T$, $y \in \mathbb{R}^n$.
2. **Repeat until** $\delta < \epsilon$:
   3. **Update** $(y, V)$ by Algorithm 2.
   4. **Compute** $Z = -(X/(\sigma + C + A^Ty))$.
   5. **Update** $r \leftarrow \text{rank}(Z)$ and $V$ so that $VV^T = Z$.
   6. **Compute** $\delta = \max(r_P, r_D)$.
   7. **Update** $\sigma$.

4.1. Convergence Analysis

The convergence analysis of DADAL may follow from the analysis in [5], that looks at ADAL as a fixed point method. One iteration of the ADAL method can be seen as the result of the combination of two operators, namely $(Z^k, X^k) = \mathcal{P}(W(Z^k, X^k))$, where $\mathcal{P}$ denotes the projection performed at Step 4 in Algorithm 1 and $W(Z^k, X^k) = X^k/(\sigma + C + A^Ty)(Z^k, X^k)$, being $y(Z^k, X^k) = (A(A^T)^{-1}b)/(\sigma + A(X^k/\sigma + r - C + Z^k))$. It can be shown that $\mathcal{P}$ and $W$ are non-expansive operators (see [5] for further details). Hence, the key step in the proof of Theorem 2 in [5] states the following

$$
\|((Z^{k+1}, X^{k+1})/\sigma) - (Z^k, X^k)/\sigma\| = \|\mathcal{P}(W(Z^k, X^k)) - \mathcal{P}(W(Z^k, X^k))\|
\leq \|W(Z^k, X^k) - W(Z^k, X^k)\|
\leq \|((Z^k, X^k)/\sigma) - (Z^k, X^k)/\sigma\|.

(14)

In DADAL, before the projection step, the dual variables are updated by performing one maximization step for the augmented Lagrangian, so that we get $(\hat{y}^k, \hat{v}^k)$ from $(y^k, v^k)$ and, in particular, the dual matrix is given as $\hat{Z}^k = \hat{y}^k \hat{v}^{k^T}$.

**Proposition 3.** Let $\hat{Z}, \hat{Z}, X, X^*, Z^* \in S_n$ and let $Z \neq Z^*$. Let $\|\hat{Z} - Z^*\|^2 \leq \rho Z - Z^*, \|Z - Z^*\|^2$, with $0 < \rho \leq 1$. Then $\hat{\rho}$ exists, $\hat{\rho} \leq 1$, $\hat{\rho} \geq \rho$ such that $\|(\hat{Z}, X) - (Z^*, X^*)\|^2 \leq \hat{\rho} \|(Z, X) - (Z^*, X^*)\|^2$.

**Proof.**

$$
\|(\hat{Z}, X) - (Z^*, X^*)\|^2 = \|\hat{Z} - Z^*\|^2 + \|X - X^*\|^2
\leq \hat{\rho}\|Z - Z^*\|^2 + \|X - X^*\|^2 = \hat{\rho}\|(Z, X) - (Z^*, X^*)\|^2,
$$

where $\hat{\rho} = \rho + \varepsilon$, with

$$
\varepsilon = (1 - \rho) c(X, Z),
$$

and

$$
c(X, Z) := \frac{\|X - X^*\|^2}{\|Z, X) - (Z^*, X^*)\|^2}.
$$

Note that, if $\rho < 1$, we have that $\hat{\rho} < 1$.

**Theorem 1.** Let $(\hat{y}^k, \hat{v}^k)$ be the dual variables obtained from $(y^k, v^k)$ by performing one iteration of Algorithm 2. Let the direction $D_V$ in Algorithm 2 be chosen such that

$$
\|\hat{Z}^k - Z^*\| \leq \delta\|\hat{Z}^k - Z^*\|.
$$

(15)

Then, the series of inequalities (14) can be extended as:

$$
\|(\hat{Z}^k, X^k)/\sigma) - (Z^*, X^*)/\sigma\| \leq \hat{\delta}\|(\hat{Z}^k, X^k)/\sigma) - (Z^*, X^*)/\sigma\|.
$$

The rest of the proof follows the same arguments as those in Theorem 2 in [5].

**Remark 1.** When dealing with unconstrained optimization problems, it is well known that the sequence $(\hat{x}^k)$ produced by Newton’s method converges superlinearly to a stationary point $x^*$ if the starting point $x^0$ is sufficiently close to $x^*$ (see e.g. Prop 1.4.1 in [7]). Therefore, condition (15) is satisfied when, e.g., $D_V$ is chosen as the Newton direction and our starting $V$ in Algorithm 2 is in a neighborhood of the optimal solution. Assumption (15) is in fact motivating our direction computation: as soon as we are close enough to an optimal solution, we try to mimic the Newton direction by scaling the gradient with the inverse of the diagonal of the Hessian (13).

4.2. Choice and Update of the Penalty Parameter

Let $(y^0, Z^0, X^0)$ be our starting solution. In defining a starting penalty parameter $\sigma^0$, i.e. a starting value for scaling the violation of the dual equality constraints in the augmented Lagrangian, we might want to take into account the dual infeasibility error $r_D$ (defined in Section 2) at the starting solution. Since the penalty parameter enters in the update of the primal solution (8) as well, we can tune $\sigma^0$ so that the starting primal and dual infeasibility errors are balanced.

Our proposal is to use the following as starting penalty parameter:

$$
\sigma^0 = \frac{r_D}{\|aX^0 - b\|/\|C - Z^0 - A^Ty^0\|}.
$$

(16)

It has been noticed that the update of the penalty parameter $\sigma$ is crucial for the computational performances of ADAL methods.
for SDPs \cite{5,6,7}. Therefore, in order to improve the numerical performance of DADAL, strategies to dynamically adjust the value of $\sigma$ may be considered: In the implementation of DADAL used in our numerical experience (see Section 5), we adopt the following strategy.

We monitor the primal and dual relative errors, and note that the stopping condition of the augmented Lagrangian method is given by $r_p \leq \varepsilon$, provided that the inner problem is solved with reasonable accuracy. In order to improve the numerical performance, we use the following intuition. If $r_p$ is much smaller than $r_p$ (we test for $100r_p < r_p$), this indicates that $\sigma$ is too big in the subproblem. On the other hand, if $r_p$ is much larger than $r_p$ (we test for $2r_p > r_p$), then $\sigma$ should be increased to facilitate overall progress. If either of these two conditions occurs consecutively for several iterations, we change $\sigma$ dividing or multiplying it by 1.3. Similar heuristics have been suggested also in \cite{5} and \cite{4}.

5. Numerical Results

In this section we report our numerical experience: we compare the performance of DADAL and \texttt{mprw.m} on randomly generated instances, on instances from the SDP problem underlying the Lovász theta number of a graph and on linear ordering problem instances. Both \texttt{mprw.m} and DADAL are implemented in MATLAB R2014b and are available at \url{https://www.math.aau.at/or/Software/}. In our implementation of Algorithm 3, we use the choice and update strategy for the penalty parameter $\sigma$ described in Section 4.2 and we perform two iterations of Algorithm 2 in order to update $(y, V)$ in Step 3. We set the accuracy level $\varepsilon = 10^{-5}$.

We also report the run time of the interior point method for SDP using the solver \texttt{MOSEK} \cite{8}. The experiments were carried out on an Intel Core i7 processor running at 3.1 GHz under Linux.

5.1. Comparison on randomly generated instances

The random instances considered in the first experiment are some of those used in \cite{4} (see Table 1). For these instances, the Cholesky factor of $AA^T$ is computed once and then used along the iterations in order to update the dual variable $y$.

As can be seen in Table 2, interior point methods are not able to solve instances with more than 30000 constraints due to memory limitations. For what concerns DADAL, the number of iterations decreases. However, we observe that a decrease in the number of iterations does not always correspond to an improvement in the computational time: This suggests that the update of the dual variables performed at Step 3 in DADAL may be too expensive. According to the MATLAB profiling of our code, this is due to the need of solving three linear systems for the update of $y$ (see Proposition 2).

5.2. Computation of the Lovász theta number

Given a graph $G$, let $V(G)$ and $E(G)$ be its set of vertices and its set of edges, respectively. The Lovász theta number $\theta(G)$ of
\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|}
\hline
Problem & \(n\) & \(m\) & \(\theta(G)\) \\
\hline
\(\theta\)-62 & 300 & 13389 & 29.6413 \\
\(\theta\)-82 & 400 & 23871 & 34.3669 \\
\(\theta\)-102 & 500 & 37466 & 38.3906 \\
\(\theta\)-103 & 500 & 62515 & 22.5286 \\
\(\theta\)-104 & 500 & 87244 & 33.1363 \\
\(\theta\)-123 & 600 & 90019 & 24.6687 \\
\(\theta\)-162 & 800 & 127599 & 37.0097 \\
\(\theta\)-1000 & 1000 & 249750 & 31.8053 \\
\(\theta\)-1500 & 1500 & 562125 & 38.8665 \\
\(\theta\)-2000 & 2000 & 999500 & 44.8558 \\
\hline
\end{tabular}
\caption{Instances from the Kim-Chuan Toh collection \cite{toh1997interior}.}
\end{table}

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline
Problem & \hline
\hline
\(\theta\)-62 & 205.1 & 790 & 9.4 & 173 & 6.8 \\
\(\theta\)-82 & 1174.5 & 821 & 18.2 & 143 & 11.1 \\
\(\theta\)-102 & - & 852 & 32.2 & 163 & 20.7 \\
\(\theta\)-103 & - & 848 & 33.8 & 192 & 25.1 \\
\(\theta\)-104 & - & 873 & 34.0 & 290 & 32.9 \\
\(\theta\)-123 & - & 874 & 55.0 & 191 & 38.6 \\
\(\theta\)-162 & - & 907 & 108.9 & 165 & 61.2 \\
\(\theta\)-1000 & - & 941 & 212.9 & 177 & 117.8 \\
\(\theta\)-1500 & - & 997 & 803.6 & 131 & 266.6 \\
\(\theta\)-2000 & - & 1034 & 1948.5 & 111 & 478.8 \\
\hline
\end{tabular}
\caption{Comparison on \(\theta\)-number instances (from the Kim-Chuan Toh collection).}
\end{table}

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline
Problem & MSEK time(s) & mprw.m iter & mprw.m time(s) & DADAL iter & DADAL time(s) \\
\hline
\(\theta\)-62 & 205.1 & 790 & 9.4 & 173 & 6.8 \\
\(\theta\)-82 & 1174.5 & 821 & 18.2 & 143 & 11.1 \\
\(\theta\)-102 & - & 852 & 32.2 & 163 & 20.7 \\
\(\theta\)-103 & - & 848 & 33.8 & 192 & 25.1 \\
\(\theta\)-104 & - & 873 & 34.0 & 290 & 32.9 \\
\(\theta\)-123 & - & 874 & 55.0 & 191 & 38.6 \\
\(\theta\)-162 & - & 907 & 108.9 & 165 & 61.2 \\
\(\theta\)-1000 & - & 941 & 212.9 & 177 & 117.8 \\
\(\theta\)-1500 & - & 997 & 803.6 & 131 & 266.6 \\
\(\theta\)-2000 & - & 1034 & 1948.5 & 111 & 478.8 \\
\hline
\end{tabular}
\caption{Comparison between mprw.m and DADAL on \(\theta\)-number instances - best parameter tuning.}
\end{table}

\subsection{Comparison on linear ordering problem instances}

Ordering problems associate to each ordering (or permutation) of a set of \(n\) objects \(N = \{1, \ldots, n\}\) a profit and the goal is to find an ordering of maximum profit. In the case of the linear ordering problem (LOP), this profit is determined by those pairs \((u, v) \in N \times N\), where \(u\) comes before \(v\) in the ordering. The simplest formulation of LOP problems is a binary linear programming problem. Several semidefinite relaxation have been proposed to compute bounds on this challenging combinatorial problem \cite{bennell2009new}, the basic one obtained from the matrix lifting approach has the following formulation:

\[
\max \langle C, Z \rangle \\
\text{s.t. } X_{ij} = 0, \quad \forall \, i, j \in E(G), \\
\text{trace} X = 1, \quad X \in S_n^+,
\]

where \(J\) is the matrix of all ones.

In Table 3 we report the dimension and the optimal value of the theta instances considered, obtained from some random graphs of the Kim-Chuan Toh collection \cite{toh1997interior}. As in the case of randomly generated instances, \texttt{MOSEK} can only solve instances with \(m < 25000\).

In the case of theta instances \(AA^T\) is a diagonal matrix, so that the update of the dual variable \(y\) turned out to be less expensive with respect to the case of random instances. We can see in Table 4 the benefits of using DADAL: the improvements both in terms of number of iterations and in terms of computational time are evident. We want to underline that for specific instances different tuning of the parameters in \texttt{DADAL} can further improve the running time. In Table 5 we report the results we obtained with DADAL keeping \(\sigma\) fixed to \(\sigma^0\) along the iterations and performing only one iteration of Algorithm 2 in order to update \((y, V)\) in Step 3: this turned out to be a better choice for these instances and resulted in even a better performance compared to mprw.m.

5.3. Comparison on linear ordering problem instances

We investigate the idea of factorizing the dual variable \(Z\) when solving SDPs in standard form within augmented Lagrangian approaches. Our proposal is to use a first order update of the dual variables in order to improve the convergence rate of ADAL methods. We add this improvement step to the implementation mprw.m and we conclude that the approach proposed looks particularly promising for solving structured SDPs (which is the case for many applications).

From our computational experience we notice that the spectral decomposition needed to perform the projection is not necessarily the computational bottleneck anymore. In fact, the ma-
trix multiplications needed in order to update y (see Proposition 2) can be the most expensive operations (e.g. in the case of randomly generated instances).

We also tried to use the factorization of $Z$ in a “pure” augmented Lagrangian algorithm. However, this turned out to be not competitive with respect to ADAL methods. This maybe due to the fact that positive semidefiniteness of the primal matrix and complementarity conditions are not satisfied by construction (as is the case in ADAL methods) and this slows down the convergence. We want to remark that dealing with the update of the penalty parameter in ADAL methods turned out to be a critical issue. Here we propose a unified strategy that leads to a satisfactory performance on the instances tested. However, for specific instances, different parameter tuning may further improve the performance as demonstrated for the case of computing the $\theta$-number.

Finally, we want to comment about the extension of DADAL to deal with SDPs in general form. Of course, any inequality constraint may be transformed into an equality constraint via the introduction of a non-negative slack variable, so that DADAL can be applied to solve any SDP. However, avoiding the transformation to SDPs in standard form is in general preferable, in order to preserve favorable constraint structures such as sparsity and orthogonality.

In [5], an alternating direction augmented Lagrangian method to deal with SDPs that includes, in particular, positivity constraints on the elements of the matrix $X$ is presented and tested, even if no convergence analysis is given. In fact, when considering multi-blocks alternating direction augmented Lagrangian methods, theoretical convergence is an issue [11]. The investigation on how to properly insert the proposed factorization technique within converging augmented Lagrangian schemes for SDPs in general form will be the topic of future work.

**Acknowledgement**

The authors would like to thank the anonymous referees for their careful reading of the paper and for their constructive comments, which were greatly appreciated. The third author acknowledges support by the Austrian Science Fund (FWF): I 3199-N31.