Using a Factored Dual in Augmented Lagrangian Methods for Semidefinite Programming

Marianna De Santis[a], Franz Rendl[b], Angelika Wiegele[b]

[a] *Dipartimento di Ingegneria Informatica Automatica e Gestionale, Sapienza Università di Roma, Via Ariosto, 25, 00185 Roma, Italy*
[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 propose to use the approximate maximum of the augmented Lagrangian to improve the convergence rate of alternating direction augmented Lagrangian (ADAL) frameworks. Numerical results are shown, giving some insights on 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 [1, 2, 3, 4, 5]. It is the purpose of this paper to elaborate on the alternating direction augmented Lagrangian (ADAL) algorithms proposed in [3, 4] 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], that looks at ADAL as a fixed point method. We give insights on how `DADAL` improves 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 $\mathcal{S}_n$ be the set of n-by-n symmetric matrices and $\mathcal{S}_n^+ \subset \mathcal{S}_n$ be the set of positive semidefinite matrices. Denoting by $\langle X, Y\rangle = \text{trace}(XY)$ the standard inner product in $\mathcal{S}_n$, we write the standard primal-dual pair of SDP problems as follows:

$$\begin{array}{ll} \min & \langle C, X\rangle \\ \text{s.t.} & \mathcal{A}X = b, \\ & X \in \mathcal{S}_n^+ \end{array} \tag{1}$$

and

$$\begin{array}{ll} \max & b^T y \\ \text{s.t.} & C - \mathcal{A}^\top y = Z \\ & Z \in \mathcal{S}_n^+, \end{array} \tag{2}$$

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

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

$$X \in \mathcal{S}_n^+, \quad \mathcal{A}X = b, \quad Z \in \mathcal{S}_n^+, \quad \mathcal{A}^\top y - Z = C, \quad ZX = 0. \tag{3}$$

We further assume that matrix A has full rank.

Let $v \in \mathbb{R}^n$ and $M \in \mathbb{R}^{m\times n}$. 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 (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 $(S)_-$.

Email addresses: `marianna.desantis@uniroma1.it` (Marianna De Santis), `franz.rendl@aau.at` (Franz Rendl), `angelika.wiegele@aau.at` (Angelika Wiegele)

2. Augmented Lagrangian Methods for SDP

Let $X \in \mathcal{S}_n$ be the Lagrange multiplier for the dual equation $Z - C + \mathcal{A}^\top y = 0$. In order to solve Problem (2) with the augmented Lagrangian method we introduce

$$L_\sigma(y, Z; X) := b^T y - \langle Z - C + \mathcal{A}^\top y, X\rangle - \frac{\sigma}{2}\|Z - C + \mathcal{A}^\top y\|^2.$$

To solve (2), we deal with

$$\begin{array}{ll} \max & L_\sigma(y, Z; X) \\ \text{s.t.} & y \in \mathbb{R}^m, \quad Z \in \mathcal{S}_n^+, \end{array} \tag{4}$$

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

Once Problem (4) is (approximately) solved, the multiplier X is updated by a first order rule:

$$X = X + \sigma(C - Z - \mathcal{A}^\top y) \tag{5}$$

and the process is iterated until convergence, i.e. until the optimality conditions (3) are satisfied within a certain tolerance (see Chapter 2 in [6] for further details).

Problem (4) 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:

$$y^{k+1} := \arg\max_{y\in\mathbb{R}^m} L_{\sigma^k}(y, Z^k; X^k), \tag{6}$$

$$Z^{k+1} := \arg\max_{Z\in\mathcal{S}_n^+} L_{\sigma^k}(y^{k+1}, Z; X^k), \tag{7}$$

$$X^{k+1} := X^k + \sigma^k(C - Z^{k+1} - \mathcal{A}^\top y^{k+1}). \tag{8}$$

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): y^{k+1} is the unique solution of

$$\nabla_y L_{\sigma^k}(y, Z^k; X^k) = b - \mathcal{A}(X^k + \sigma^k(Z^k - C + \mathcal{A}^\top y)) = 0,$$

that is

$$y^{k+1} = (\mathcal{A}\mathcal{A}^\top)^{-1}\Big(\frac{1}{\sigma^k}b - \mathcal{A}(\frac{1}{\sigma^k}X^k - C + Z^k)\Big).$$

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

$$\min_{Z\in\mathcal{S}_n^+} \|Z - W^k\|^2, \tag{9}$$

with $W^k = (\frac{X^k}{\sigma^k} - C + \mathcal{A}^\top y^{k+1})$, or, in other words, by projecting $W^k \in \mathcal{S}_n$ onto the (closed convex) cone $\mathcal{S}_n^-$ 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 [3, 4] can also be viewed as an alternating direction augmented Lagrangian method. In fact, the available implementation `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 [3]) only one iteration is performed.

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

$$r_P = \frac{\|\mathcal{A}X - b\|}{1 + \|b\|},$$

$$r_D = \frac{\|C - Z - \mathcal{A}^\top y\|}{1 + \|C\|};$$

as the other optimality conditions (namely, $X \in \mathcal{S}_n^+, Z \in \mathcal{S}_n^+, ZX = 0$) 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 mprw.m`

1 **Initialization:** Choose $\sigma > 0$, $X \in \mathcal{S}_n^+$, $\varepsilon > 0$.
  Set $Z = 0$.
2 **Repeat until** $\delta < \varepsilon$:
3 **Compute** $y = (\mathcal{A}\mathcal{A}^\top)^{-1}\Big(\frac{1}{\sigma}b - \mathcal{A}(\frac{1}{\sigma}Y - C + Z)\Big)$
4 **Compute** $Z = -(X/\sigma - C + \mathcal{A}^\top y)_-$
  $X = \sigma(X/\sigma - C + \mathcal{A}^\top y)_+$
5 **Compute** $\delta = \max\{r_P, r_D\}$
6 **Update** σ

It is the main purpose of this paper to investigate enhancements to this algorithm. The key idea will be to replace in the subproblem (4) the constraint $Z \in \mathcal{S}_n^+$ by $Z = VV^\top$ and considering (4) as an unconstrained problem in y and V.

3. Solving the subproblem (4)

By introducing a variable $V \in \mathbb{R}^{n\times r}$ $(1 \le r \le n)$, such that $Z = VV^\top \in \mathcal{S}_n^+$, we reformulate Problem (4) as the following unconstrained maximization problem

$$\begin{array}{ll} \max & L_\sigma(y, V; X) \\ \text{s.t.} & y \in \mathbb{R}^m, \quad V \in \mathbb{R}^{n\times r}, \end{array} \tag{10}$$

where

$$L_\sigma(y, V; X) = b^T y - \langle VV^\top - C + \mathcal{A}^\top y, X\rangle - \frac{\sigma}{2}\|VV^\top - C + \mathcal{A}^\top y\|^2.$$

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 $(y^*, V^*) \in \mathbb{R}^m \times \mathbb{R}^{n\times r}$ be a stationary point for Problem* (10)*, then*

$$\begin{aligned} \nabla_y L_\sigma(y^*, V^*; X) &= b - \mathcal{A}(X + \sigma(V^*V^{*\top} - C + \mathcal{A}^\top y^*)) = 0, \\ \nabla_V L_\sigma(y^*, V^*; X) &= -2(X + \sigma(V^*V^{*\top} - C + \mathcal{A}^\top y^*))V^* = 0. \end{aligned} \tag{11}$$

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

Proposition 2. *Let $D_V \in \mathbb{R}^{n\times r}$. Let*

$$y(V + \alpha D_V) = y_0 + \alpha y_1 + \alpha^2 y_2, \tag{12}$$

with

$$\begin{aligned} y_0 &= (\mathcal{A}\mathcal{A}^\top)^{-1}\Big(\tfrac{1}{\sigma}b - \mathcal{A}(\tfrac{1}{\sigma}X - C + VV^\top)\Big),\\ y_1 &= (\mathcal{A}\mathcal{A}^\top)^{-1}\Big(-\mathcal{A}(D_V V^\top + VD_V^\top)\Big),\\ y_2 &= (\mathcal{A}\mathcal{A}^\top)^{-1}\Big(-\mathcal{A}(D_V D_V^\top)\Big). \end{aligned}$$

Then

$$\nabla_y L_\sigma(y(V + \alpha D_V), V + \alpha D_V; X) = 0,$$

for all $\alpha \in \mathbb{R}$.

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

$$\mathcal{A}\mathcal{A}^\top y = \Big(\frac{1}{\sigma}b - \mathcal{A}(\frac{1}{\sigma}X - C + VV^\top)\Big).$$

Therefore, when $V = V + \alpha D_V$, we get

$$\begin{aligned} (\mathcal{A}\mathcal{A}^\top)y &= \frac{1}{\sigma}b - \mathcal{A}\Big(\frac{1}{\sigma}X - C + (V + \alpha D_V)(V + \alpha D_V)^\top\Big)\\ &= \frac{1}{\sigma}b - \mathcal{A}\Big(\frac{1}{\sigma}X - C + VV^\top\Big)\\ &\quad - \alpha\mathcal{A}(D_V V^\top + VD_V^\top) - \alpha^2\mathcal{A}(D_V D_V^\top). \end{aligned}$$

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

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 possibilties. 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_σ with respect to V, see (11), as the $n \times r$ matrix

$$\nabla_V L_\sigma(y, V; X) = -2(M + \sigma VV^\top)V,$$

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

$$\lim_{t\to 0}\ \frac{1}{t}\ \Big(f_{st}(V + te_s e_t^\top) - f_{st}(V)\Big),$$

where $f_{st}(V) := e_s^\top VV^\top Ve_t, \quad s \in \{1, \dots, n\},\ t \in \{1, \dots, r\}$.

We get

$$\frac{\partial^2 L_\sigma(y, V; X)}{\partial v_{s,t}\partial v_{s,t}} = -2m_{ss} - 2\sigma\Big((e_s^\top Ve_t)^2 + \|V^\top e_s\|^2 + \|Ve_t\|^2\Big).$$

We define the $n \times r$ matrix H by

$$(H)_{s,t} = 2\max\{0, m_{ss}\} + 2\sigma\left((e_s^\top Ve_t)^2 + \|V^\top e_s\|^2 + \|Ve_t\|^2\right),$$

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

$$(D_V)_{s,t} := \frac{(\nabla_V L_\sigma)_{s,t}}{H_{s,t}}. \tag{13}$$

We note that H is generically positive, as V should not contain columns all 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 $L_\sigma(y(V+\alpha D_V), V+\alpha D_V; X)$ is a polynomial of degree 4 in α, so that we can interpolate five different points

$$(\alpha_i,\ L_\sigma(y(V + \alpha_i D_V), V + \alpha_i D_V; X)\,), \quad i = 1, \dots, 5,$$

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

Algorithm 2 is a scheme of the generic iteration we perform to maximize $L_\sigma(y, V; X)$, X being fixed.

Algorithm 2 `Solving (4) approximately`

Input: $\sigma > 0$, $y \in \mathbb{R}^m$, $V \in \mathbb{R}^{n\times r}$
Repeat until $\|\nabla_V L_\sigma\| < \epsilon_{inner}$
  Compute the search direction $D_V \in \mathbb{R}^{n\times r}$ using (13)
  Compute optimal stepsize α
  Update $y = y(V + \alpha D_V)$ according to (12)
    and $V = V + \alpha D_V$.

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 3 as `DADAL`.

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 $(Z^{k+1}, X^{k+1}) = \mathcal{P}(W(Z^k, X^k))$, where $\mathcal{P}$ denotes the projection performed at Step 4 in Algorithm 1 and $W(Z^k, X^k) = X^k/\sigma - C + \mathcal{A}^\top y(Z^k, X^k)$, being $y(Z^k, X^k) = (\mathcal{A}\mathcal{A}^\top)^{-1}\Big(b/\sigma^k - \mathcal{A}(X^k/\sigma^k - C + Z^k)\Big)$. It can be shown that $\mathcal{P}$ and W are non-expansive operators (see [5] for

Algorithm 3 DADAL

Initialization: Choose $\sigma > 0$, $r > 0$, $\varepsilon > 0$,
$X \in \mathbb{S}^+_n$, $V \in \mathbb{R}^{n\times r}$, $Z = VV^\top$, $y \in \mathbb{R}^m$.
Repeat until $\delta < \varepsilon$:
  Update (y, V) by Algorithm 2
  Compute $Z = -(X/\sigma - C + \mathcal{A}^\top y)_-$
    $X = \sigma(X/\sigma - C + \mathcal{A}^\top y)_+$
  Update $r \leftarrow$ rank(Z) and V so that $VV^\top = Z$
  Compute $\delta = \max\{r_P, r_D\}$
  Update σ

further details). Hence, the key step in the proof of Theorem 2 in [5] states the following

$$\begin{aligned}\|(Z^{k+1}, X^{k+1}/\sigma) - (Z^*, X^*/\sigma)\| &= \|\mathcal{P}(W(Z^k, X^k)) - \mathcal{P}(W(Z^*, X^*))\| \\ &\le \|W(Z^k, X^k) - W(Z^*, X^*)\| \\ &\le \|(Z^k, X^k/\sigma) - (Z^*, X^*/\sigma)\|.\end{aligned} \tag{14}$$

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 $\hat{Z}^k = \hat{V}^k \hat{V}^{k\top}$.

Proposition 3. *Let $\hat{Z}, Z, X, X^*, Z^* \in \mathcal{S}_n$ and let $Z \neq Z^*$. Let*

$$\|\hat{Z} - Z^*\|^2 \le \rho \|Z - Z^*\|^2,$$

with $\rho \le 1$. Then $\bar{\rho}$ exists, $\bar{\rho} \le 1$, $\bar{\rho} \ge \rho$ such that

$$\|(\hat{Z}, X) - (Z^*, X^*)\|^2 \le \bar{\rho} \|(Z, X) - (Z^*, X^*)\|^2$$

Proof.

$$\begin{aligned}\|(\hat{Z}, X) - (Z^*, X^*)\|^2 &= \|\hat{Z} - Z^*\|^2 + \|X - X^*\|^2 \\ &\le \rho \|Z - Z^*\|^2 + \|X - X^*\|^2 \\ &= \bar{\rho} \|(Z, X) - (Z^*, X^*)\|^2,\end{aligned}$$

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

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

and

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

Since $0 \le c(X, Z) < 1$, we have $0 \le \varepsilon \le (1 - \rho)$, so that $\bar{\rho} \le 1$ and $\bar{\rho} \ge \rho$. □

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

$$\|\hat{Z}^k - Z^*\| \le \delta \|Z^k - Z^*\|, \tag{15}$$

with $\delta \le 1$. Then the sequence $\{X^k, y^k, Z^k\}$ generated by Algorithm 3 converges to a solution $\{X^*, y^*, Z^*\}$.

Proof. Since condition (15) holds, we can apply Proposition 3, so that $\bar{\delta}$ exists, $\delta \le \bar{\delta} \le 1$, such that

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

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

$$\begin{aligned}\|(Z^{k+1}, X^{k+1}/\sigma) - (Z^*, X^*/\sigma)\| &= \|\mathcal{P}(W(\hat{Z}^k, X^k)) - \mathcal{P}(W(Z^*, X^*))\| \\ &\le \|W(\hat{Z}^k, X^k) - W(Z^*, X^*)\| \\ &\le \|(\hat{Z}^k, X^k/\sigma) - (Z^*, X^*/\sigma)\| \\ &\le \bar{\delta} \|(Z^k, X^k/\sigma) - (Z^*, X^*/\sigma)\|.\end{aligned}$$

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

Note that $\delta < 1$ in (15) implies $\bar{\delta} < 1$ and the additional step of maximizing the augmented Lagrangian is strictly improving the convergence rate of ADAL methods.

Remark 1. *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 σ^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 σ^0 so that the starting primal and dual infeasibility errors are balanced.

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

$$\sigma^0 = \frac{r_P}{r_D} \frac{\|\mathcal{A}X^0 - b\|}{\|C - Z^0 - \mathcal{A}^\top y^0\|} \tag{16}$$

It has been noticed that the update of the penalty parameter σ is crucial for the computational performances of ADAL methods for SDPs [3, 4, 5]. Therefore, in order to improve the numerical perfomance of DADAL, strategies to dynamically adjust the value of σ 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 $r_D \le \varepsilon$, provided that the inner problem is solved with reasonable accuracy. In order to improve the numerical performance, we use the following intuition. If r_D is much smaller than r_P (we test for $100 r_D < r_p$), this indicates that σ is too big in the subproblem. On the other hand, if r_D is much larger than r_p (we test for $2 r_D > r_P$), then σ should be increased to facilitate overall progress. If either of these two conditions occurs consecutively for several iterations, we change σ dividing or multiplying it by 1.3. Similar heuristics have been suggested also in [5] and [4].

Problem	n	m	p	seed
P1	300	20000	3	3002030
P2	300	25000	3	3002530
P3	300	10000	4	3001040
P4	400	30000	3	4003030
P5	400	40000	3	4004030
P6	400	15000	4	4001540
P7	500	30000	3	5003030
P8	500	40000	3	5004030
P9	500	50000	3	5005030
P10	500	20000	4	5002040
P11	600	40000	3	6004030
P12	600	50000	3	6005030
P13	600	60000	3	6006030
P14	600	20000	4	6002040
P15	700	50000	3	7005030
P16	700	70000	3	7007030
P17	700	90000	3	7009030
P18	800	70000	3	8007030
P19	800	100000	3	80010030
P20	800	110000	3	80011030

Table 1: Randomly generated instances.

	MOSEK	mprw.m		DADAL	
Problem	time(s)	iter	time(s)	iter	time(s)
P1	633.1	340	**10.1**	**68**	10.2
P2	2440.4	427	33.1	**76**	**31.2**
P3	129.1	260	**11.3**	**146**	32.6
P4	7514.6	306	**13.3**	**101**	18.9
P5	-	376	**51.4**	**73**	56.3
P6	375.7	255	**19.5**	**187**	72.9
P7	7388.3	268	**11.5**	**151**	23.1
P8	-	289	**15.2**	**130**	30.7
P9	-	319	**35.3**	**111**	74.4
P10	886.0	251	**26.9**	**222**	119.8
P11	-	266	**17.3**	**177**	41.9
P12	-	275	**18.6**	**148**	42.1
P13	-	293	**28.4**	**132**	68.7
P14	1156.3	**249**	**20.3**	270	116.2
P15	-	262	**22.4**	**207**	83.7
P16	-	278	**30.1**	**151**	79.2
P17	-	303	**74.2**	**128**	181.5
P18	-	264	**34.3**	**207**	111.2
P19	-	285	**53.3**	**157**	126.4
P20	-	296	**80.9**	**133**	168.6

Table 2: Comparison on randomly generated instances.

5. Numerical Results

In this section we report our numerical experience: we compare the performance of DADAL and 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 mprw.m and DADAL are implemented in MATLAB R2014b and are available at https://www.math.aau.at/or/Software/. In our implementation of Algorithm 3, we use the choice and update strategy for the penalty parameter σ 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 MOSEK [7]. 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 [4] (see Table 1). For these instances, the Cholesky factor of $\mathcal{A}\mathcal{A}^\top$ 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 $\vartheta(G)$ of G is defined as the optimal value of the following SDP problem:

$$\begin{aligned} &\max \langle J, X\rangle \\ &\text{s.t. } X_{ij} = 0, \quad \forall\, ij \in E(G), \\ &\quad \text{traceX} = 1, \quad X \in \mathcal{S}^+_n, \end{aligned}$$

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 [8]. As in the case of randomly generated instances, MOSEK can only solve instances with $m < 25000$.

In the case of theta instances $\mathcal{A}\mathcal{A}^\top$ is a diagonal matrix, so that the update of the dual variable y turned 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 DADAL can further improve the running time. In Table 5, we report the results we obtained with DADAL keeping σ fixed to σ^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 perfomance compared to mprw.m.

5.3. 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

Problem	n	m	$\vartheta(G)$
ϑ-62	300	13389	29.6413
ϑ-82	400	23871	34.3669
ϑ-102	500	37466	38.3906
ϑ-103	500	62515	22.5286
ϑ-104	500	87244	13.3363
ϑ-123	600	90019	24.6687
ϑ-162	800	127599	37.0097
ϑ-1000	1000	249750	31.8053
ϑ-1500	1500	562125	38.8665
ϑ-2000	2000	999500	44.8558

Table 3: Instances from the Kim-Chuan Toh collection [8].

	MOSEK	mprw.m		DADAL	
Problem	time(s)	iter	time(s)	iter	time(s)
ϑ-62	205.1	790	9.4	**173**	**6.8**
ϑ-82	1174.5	821	18.2	**143**	**11.1**
ϑ-102	-	852	32.2	**163**	**20.7**
ϑ-103	-	848	33.8	**192**	**25.1**
ϑ-104	-	873	34.0	**290**	**32.9**
ϑ-123	-	874	55.0	**191**	**38.6**
ϑ-162	-	907	108.9	**165**	**61.2**
ϑ-1000	-	941	212.9	**177**	**117.8**
ϑ-1500	-	997	803.6	**131**	**266.6**
ϑ-2000	-	1034	1948.5	**111**	**478.8**

Table 4: Comparison on ϑ-number instances (from the Kim-Chuan Toh collection).

	mprw.m		DADAL	
Problem	iter	time(s)	iter	time(s)
ϑ-62	790	9.4	**132**	**4.9**
ϑ-82	821	18.2	**124**	**6.8**
ϑ-102	852	32.2	**123**	**11.3**
ϑ-103	848	33.8	**125**	**11.7**
ϑ-104	873	34.0	**166**	**15.2**
ϑ-123	874	55.0	**121**	**17.1**
ϑ-162	907	108.9	**103**	**27.4**
ϑ-1000	941	212.9	**105**	**50.6**
ϑ-1500	997	803.6	**94**	**136.8**
ϑ-2000	1034	1948.5	**89**	**283.9**

Table 5: Comparison between mprw.m and DADAL on theta number instances - best parameter tuning.

			MOSEK	mprw.m		DADAL	
$\lvert N\rvert$	n	m	time(s)	iter	time(s)	iter	time(s)
10	46	166	0.4	411	**0.2**	**117**	0.8
20	191	1331	3.6	441	**2.3**	**150**	2.6
30	436	4496	34.5	483	**12.1**	**170**	15.5
40	781	10661	294.1	542	**60.7**	**232**	79.8
50	1226	20826	1339.8	604	237.7	**215**	**232.1**
60	1771	35991	-	636	759.8	**252**	**748.3**
70	2416	57156	-	707	**2049.1**	**273**	2122.7
80	3161	85321	-	745	4788.6	**282**	**4395.9**
90	4006	121486	-	773	9589.3	**300**	**8923.2**
100	4951	166651	-	821	18820.7	**323**	**17944.1**

Table 6: Comparison on linear ordering problems.

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 [9], the basic one obtained from the matrix lifting approach has the following formulation:

$$\begin{aligned} &\max \langle C, Z\rangle \\ &\text{s.t. } Z \in \mathcal{S}_n^+,\ \operatorname{diag}(Z) = e, \\ &\quad y_{ij,jk} - y_{ij,ik} - y_{ik,jk} = -1, \quad \forall i < j < k, \end{aligned}$$

where $Z = \begin{bmatrix} 1 & y^\top \\ y & Y \end{bmatrix}$ is of order $n = \binom{N}{2} + 1$. We have considered LOP instances where the dimension of the set N ranges from 10 to 100 and the matrix C is randomly generated.

Again, for these instances, we have that $\mathcal{A}\mathcal{A}^\top$ is a diagonal matrix and using DADAL, especially when dealing with large scale instances, leads to an improvement both in terms of number of iterations and in terms of computational time (see Table 6).

6. Conclusions

We investigate the idea of factorizing the dual variable Z when solving SDPs 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 matrix 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. Finally, 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 ϑ-number.

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.

References

[1] S. Burer, R. D. C. Monteiro, A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization, Math. Program. 95 (2, Ser. B) (2003) 329–357.

[2] S. Burer, R. D. C. Monteiro, Local minima and convergence in low-rank semidefinite programming, Math. Program. 103 (3, Ser. A) (2005) 427–444.

[3] J. Povh, F. Rendl, A. Wiegele, A Boundary Point Method to solve Semidefinite Programs, Computing 78 (2006) 277–286.

[4] J. Malick, J. Povh, F. Rendl, A. Wiegele, Regularization methods for semidefinite programming, SIAM J. Optim. 20 (1) (2009) 336–356.

[5] Z. Wen, D. Goldfarb, W. Yin, Alternating direction augmented lagrangian methods for semidefinite programming, Math. Progr. Comp. 2 (3) (2010) 203–230.

[6] D. P. Bertsekas, Constrained optimization and Lagrange multiplier methods, Computer Science and Applied Mathematics, Academic Press Inc. [Harcourt Brace Jovanovich Publishers], New York, 1982.

[7] M. ApS, The MOSEK optimization toolbox for MATLAB manual. Version 8.0.0.81. (2017).
URL `http://docs.mosek.com/8.0/toolbox/index.html`

[8] K.-C. Toh, Solving large scale semidefinite programs via an iterative solver on the augmented systems, SIAM J. Optim. 14 (3) (2003) 670–698 (electronic).

[9] P. Hungerländer, F. Rendl, Semidefinite relaxations of ordering problems, Mathematical Programming 140 (1) (2013) 77–97.