Preprints of the 20th World Congress
The International Federation of Automatic Control
Toulouse, France, July 9-14, 2017

Fast ADMM for homogeneous self-dual embedding of sparse SDPs ⋆

Yang Zheng * **Giovanni Fantuzzi** †
Antonis Papachristodoulou * **Paul Goulart** *
and Andrew Wynn †

* *Department of Engineering Science, University of Oxford, Parks Road, Oxford, OX1 3PJ, U.K. (e-mail: yang.zheng@eng.ox.ac.uk; paul.goulart@eng.ox.ac.uk; antonis@eng.ox.ac.uk).*
† *Department of Aeronautics, Imperial College London, South Kensington Campus, London, SW7 2AZ, U.K. (e-mail: gf910@ic.ac.uk; a.wynn@imperial.ac.uk).*

Abstract: We propose an efficient first-order method, based on the alternating direction method of multipliers (ADMM), to solve the homogeneous self-dual embedding problem for a primal-dual pair of semidefinite programs (SDPs) with chordal sparsity. Using a series of block eliminations, the per-iteration cost of our method is the same as applying a splitting method to the primal or dual alone. Moreover, our approach is more efficient than other first-order methods for generic sparse conic programs since we work with smaller semidefinite cones. In contrast to previous first-order methods that exploit chordal sparsity, our algorithm returns both primal and dual solutions when available, and a certificate of infeasibility otherwise. Our techniques are implemented in the open-source MATLAB solver CDCS. Numerical experiments on three sets of benchmark problems from the library SDPLIB show speed-ups compared to some common state-of-the-art software packages.

Keywords: Convex optimization, semidefinite programs, chordal sparsity, large-scale problems, first-order methods.

1. INTRODUCTION

Semidefinite programs (SDPs) are convex optimization problems commonly used in control theory, machine learning and signal processing. It is well known that although small and medium-sized SDPs can be efficiently solved in polynomial time using second-order interior-point methods (IPMs), these methods become less practical for large-scale SDPs due to memory and time constraints (Helmberg et al., 1996). As noted by Andersen et al. (2011), exploiting sparsity in SDPs has been one of the main approaches to improve the scalability of semidefinite programming, and it is still an active and challenging area of research.

In this paper, we present an efficient first-order algorithm to solve the homogeneous self-dual embedding formulation of large-scale SDPs characterized by *chordal sparsity*, meaning that the graph representing their aggregate sparsity pattern is chordal (or has a sparse *chordal extension*). Chordal graphs—undirected graphs with the property that every cycle of length greater than three has a chord—are very well studied objects in graph theory (Vandenberghe and Andersen, 2014). Their connection to SDPs relies on two fundamental theorems due to Grone et al. (1984) and Agler et al. (1988): provided that its sparsity pattern is chordal, a large positive semidefinite (PSD) cone can be equivalently replaced with a set of smaller PSD cones.

For this reason chordal sparsity is a key feature of SDPs, and recent years have seen increasing efforts to exploit it in order to increase the computational efficiency of SDP solvers. For instance, Fukuda et al. (2001) and Kim et al. (2011) proposed the *domain-space* and the *range-space* conversion techniques to reduce the computational burden of existing IPMs for sparse SDPs. These techniques, implemented in the MATLAB package SparseCoLO (Fujisawa et al., 2009), rely on the introduction of additional equality constraints to decouple the smaller PSD cones obtained from Grone's and Agler's theorems. However, the addition of equality constraints often offsets the benefit of working with smaller semidefinite cones.

One possible solution to this problem is to exploit the properties of chordal sparsity directly in IPMs (Fukuda et al., 2001; Andersen et al., 2010). Another promising direction is to solve decomposable SDPs via first-order methods. For instance, Sun et al. (2014) proposed a first-order splitting algorithm for conic optimization with partially separable structure. Kalbat and Lavaei (2015) applied the alternating direction method of multipliers (ADMM) to solve a special class of SDPs with fully decomposable constraints. Madani et al. (2015) developed a highly-parallelizable ADMM algorithm for sparse SDPs with applications to optimal power flow problems. More recently, the authors have combined ADMM and chordal

⋆ Y. Zheng and G. Fantuzzi contributed equally to this work. Y. Zheng is supported by the Clarendon Scholarship and the Jason Hu Scholarship.

decomposition to solve sparse SDPs in either primal or dual standard forms (Zheng et al., 2016), providing a conversion framework which is suitable for the application of first-order methods and parallels that of Fukuda et al. (2001) and Kim et al. (2011) for IPMs.

However, none of the aforementioned first-order methods can handle infeasible or unbounded problems. Solving the homogeneous self-dual embedding of the primal-dual pair of optimization problems (Ye et al., 1994) provides an elegant solution to this issue. The essence of this method is to search for a non-zero point in the non-empty intersection of a convex cone and an affine space. Using this point, one can then either recover an optimal solution of the original primal-dual pair of SDPs, or construct a certificate of primal or dual infeasibility. Homogeneous self-dual embeddings have been widely used in IPMs (Sturm, 1999; Ye, 2011); more recently, O'Donoghue et al. (2016a) have proposed an operator-splitting method for the homogeneous self-dual embedding of general conic programs that scales well with problem size, and the implementation can be found in the C package SCS (O'Donoghue et al., 2016b).

In this work, we show that the conversion techniques for primal and dual standard-form SDPs developed in Zheng et al. (2016) can be extended to the homogeneous self-dual embedding. Also, we extend the algorithm in O'Donoghue et al. (2016a) to take advantage of chordal sparsity. Our main contributions are:

(1) We formulate the homogeneous self-dual embedding of a primal-dual pair of SDPs whose conic constraints are decomposed using Grone's and Agler's theorems. This extends the conversion techniques for sparse SDPs developed in our previous work (Zheng et al., 2016). To the best of our knowledge, it is the first time that such a formulation has been presented.
(2) We extend the ADMM algorithm of O'Donoghue et al. (2016a) to take advantage of the special structure of our homogeneous self-dual formulation, thereby reducing its computational complexity. Our algorithm is more efficient than a direct application of the method of O'Donoghue et al. (2016a) to either the original primal-dual pair (i.e. before chordal sparsity is taken into account), or the decomposed problems: in the former case, the chordal decomposition reduces the cost of the conic projections; in the latter case, we speed up the affine projection step using a series of block-eliminations.
(3) We implement our techniques in the MATLAB solver CDCS (Cone Decomposition Conic Solver). This is the first open source first-order solver that exploits chordal decomposition and is able to handle infeasible problems. Numerical simulations on three sets of benchmark problems from the library SDPLIB (Borchers, 1999) demonstrate the efficiency of our self-dual algorithm compared to other commonly used software packages.

The rest of this paper is organized as follows. Section 2 reviews some background material. We present the homogeneous self-dual embedding of SDPs with chordal sparsity in Section 3. Section 4 discusses our ADMM algorithm in detail, and we report numerical experiments in Section 5. Finally, Section 6 offers concluding remarks.

2. PRELIMINARIES

2.1 Chordal graphs

Let $\mathcal{G}(\mathcal{V},\mathcal{E})$ be an undirected graph with nodes $\mathcal{V} = \{1,2,\ldots,n\}$ and edges $\mathcal{E} \subseteq \mathcal{V}\times\mathcal{V}$. A subset of nodes $\mathcal{C} \subseteq \mathcal{V}$ is called a *clique* if $(i,j) \in \mathcal{E}$ for any distinct nodes $i,j \in \mathcal{C}$. If $\mathcal{C}$ is not a subset of any other clique, then it is called a *maximal clique*. The number of nodes in $\mathcal{C}$ is denoted by $|\mathcal{C}|$, and $\mathcal{C}(i)$ indicates the i-th element of $\mathcal{C}$, sorted in the natural ordering. An undirected graph $\mathcal{G}$ is called *chordal* if every cycle of length greater than three has at least one chord (an edge connecting two nonconsecutive nodes in the cycle). Note that if $\mathcal{G}(\mathcal{V},\mathcal{E})$ is not chordal, it can be *chordal extended*, *i.e.*, we can construct a chordal graph $\mathcal{G}'(\mathcal{V},\mathcal{E}')$ by adding suitable edges to $\mathcal{E}$.

2.2 Sparse matrices defined by graphs

Let $\mathcal{G}(\mathcal{V},\mathcal{E})$ be an undirected graph, and assume that $(i,i) \in \mathcal{E}$ for any node $i \in \mathcal{V}$. A partial symmetric matrix is a symmetric matrix in which the entry X_{ij} is specified if and only if $(i,j) \in \mathcal{E}$. In this work, we use the following sets of symmetric matrices defined on $\mathcal{E}$:

$$\begin{aligned}
\mathbb{S}^n(\mathcal{E},?) &= \text{the space of } n\times n \text{ partial symmetric matrices with elements defined on } \mathcal{E},\\
\mathbb{S}^n_+(\mathcal{E},?) &= \{X \in \mathbb{S}^n(\mathcal{E},?) \mid \exists M \succeq 0, M_{ij} = X_{ij}, \forall (i,j) \in \mathcal{E}\},\\
\mathbb{S}^n(\mathcal{E},0) &= \{X \in \mathbb{S}^n \mid X_{ij} = 0, \text{if } (i,j) \notin \mathcal{E}\},\\
\mathbb{S}^n_+(\mathcal{E},0) &= \{X \in \mathbb{S}^n(\mathcal{E},0) \mid X \succeq 0\}.
\end{aligned}$$

Note that $\mathbb{S}^n_+(\mathcal{E},?)$ and $\mathbb{S}^n_+(\mathcal{E},0)$ are two types of sparse matrix cones, and that they are the dual of each other for any (that is, chordal or not) sparsity pattern $\mathcal{E}$ (Vandenberghe and Andersen, 2014).

Finally, let $\mathcal{C}$ be a maximal clique of the graph $\mathcal{G}$, and let $E_{\mathcal{C}} \in \mathbb{R}^{|\mathcal{C}|\times n}$ be the matrix with entries $(E_{\mathcal{C}})_{ij} = 1$ if $\mathcal{C}(i) = j$ and $(E_{\mathcal{C}})_{ij} = 0$ otherwise. Then, given a symmetric matrix $X \in \mathbb{S}^n$, the submatrix of X defined by the clique $\mathcal{C}$ can be represented as $E_{\mathcal{C}} X E_{\mathcal{C}}^T \in \mathbb{S}^{|\mathcal{C}|}$.

2.3 Chordal decomposition

The problems of deciding if $X \in \mathbb{S}^n_+(\mathcal{E},?)$ or $Z \in \mathbb{S}^n_+(\mathcal{E},0)$ can be posed as problems over several smaller (but coupled) convex cones according to the following theorems:

Theorem 1. (Grone et al., 1984). Let $\mathcal{G}(\mathcal{V},\mathcal{E})$ be a chordal graph with maximal cliques $\{\mathcal{C}_1,\mathcal{C}_2,\ldots,\mathcal{C}_p\}$. Then, $X \in \mathbb{S}^n_+(\mathcal{E},?)$ if and only if $X_k := E_{\mathcal{C}_k} X E_{\mathcal{C}_k}^T \in \mathbb{S}^{|\mathcal{C}_k|}_+$ for all $k = 1,\ldots,p$.

Theorem 2. (Agler et al., 1988). Let $\mathcal{G}(\mathcal{V},\mathcal{E})$ be a chordal graph with maximal cliques $\{\mathcal{C}_1,\mathcal{C}_2,\ldots,\mathcal{C}_p\}$. Then, $Z \in \mathbb{S}^n_+(\mathcal{E},0)$ if and only if there exist matrices $Z_k \in \mathbb{S}^{|\mathcal{C}_k|}_+$ for $k = 1,\ldots,p$ such that $Z = \sum_{k=1}^p E_{\mathcal{C}_k}^T Z_k E_{\mathcal{C}_k}$.

3. HOMOGENEOUS SELF-DUAL EMBEDDING OF SPARSE SDPS

Consider the standard *primal-dual pair* of SDPs, *i.e.*,

$$\begin{aligned}
\min_X \quad & \langle C, X\rangle\\
\text{subject to} \quad & \langle A_i, X\rangle = b_i, \quad i = 1,\ldots,m,\\
& X \in \mathbb{S}^n_+,
\end{aligned} \tag{1}$$

and

$$\begin{array}{ll} \max\limits_{y,Z} & \langle b, y\rangle \\ \text{subject to} & \sum\limits_{y=1}^{m} A_i y_i + Z = C, \\ & Z \in \mathbb{S}^n_+. \end{array} \tag{2}$$

The vector $b \in \mathbb{R}^m$ and the matrices C, A_1, …, $A_m \in \mathbb{S}^n$ are the problem data; X is the primal variable, and y, Z are the dual variables. We say that (1) and (2) have the *aggregate sparsity pattern* $\mathcal{G}(\mathcal{V}, \mathcal{E})$ if $C, A_1, \ldots, A_m \in \mathbb{S}^n(\mathcal{E}, 0)$. Throughout this work, we will assume that $\mathcal{G}$ is chordal (otherwise, it can be chordal extended), and that its maximal cliques $\mathcal{C}_1, \ldots, \mathcal{C}_p$ are small.

3.1 Sparse SDPs with Chordal Decomposition

Aggregate sparsity implies that the dual variable Z in (2) must have the sparsity pattern defined by $\mathcal{E}$, *i.e.*, $Z \in \mathbb{S}^n(\mathcal{E}, 0)$. Similarly, although the primal variable X in (1) is usually dense, the cost function and the equality constraints only depend on the entries X_{ij} in the sparsity pattern $\mathcal{E}$, while the remaining entries only guarantee that X is positive semidefinite. This means that it suffices to consider $X \in \mathbb{S}^n_+(\mathcal{E}, ?)$. Then, according to Theorems 1-2, we can rewrite (1) and (2), respectively, as

$$\begin{array}{ll} \min\limits_{X, X_1, \ldots, X_p} & \langle C, X\rangle \\ \text{subject to} & \langle A_i, X\rangle = b_i, \qquad i = 1, \ldots, m, \\ & X_k = E_{\mathcal{C}_k} X E_{\mathcal{C}_k}^T, \quad k = 1, \ldots, p, \\ & X_k \in \mathbb{S}^{|\mathcal{C}_k|}_+, \qquad k = 1, \ldots, p, \end{array} \tag{3}$$

and

$$\begin{array}{ll} \max\limits_{y, Z_1, \ldots, Z_p, V_1, \ldots, V_p} & \langle b, y\rangle \\ \text{subject to} & \sum\limits_{i=1}^{m} A_i y_i + \sum\limits_{k=1}^{p} E_{\mathcal{C}_k}^T V_k E_{\mathcal{C}_k} = C, \\ & Z_k = V_k, \qquad k = 1, \ldots, p, \\ & Z_k \in \mathbb{S}^{|\mathcal{C}_k|}_+, \quad k = 1, \ldots, p. \end{array} \tag{4}$$

To ease the exposition, let $\mathrm{vec} : \mathbb{S}^n \to \mathbb{R}^{n^2}$ be the usual operator mapping a matrix to the stack of its column, and define the vectorized data $c := \mathrm{vec}(C)$, $A := [\mathrm{vec}(A_0) \ \ldots \ \mathrm{vec}(A_m)]^T$, the vectorized variables $x := \mathrm{vec}(X)$, $x_k := \mathrm{vec}(X_k)$, $z_k := \mathrm{vec}(Z_k)$, $v_k := \mathrm{vec}(V_k)$ for all $k = 1, \ldots, p$, and the matrices

$$H_k := E_{\mathcal{C}_k} \otimes E_{\mathcal{C}_k}, \quad k = 1, \ldots, p, \tag{5}$$

such that $x_k = \mathrm{vec}(X_k) = \mathrm{vec}(E_k X E_k^T) = H_k x$. Note that H_1, …, H_p are "entry-selector" matrices of 1's and 0's, whose rows are orthonormal and such that $H_k^T H_k$ is diagonal. These matrices project x onto the subvectors x_1, …, x_p, respectively.

If we denote the constraints $X_k \in \mathbb{S}^{|\mathcal{C}_k|}_+$ by $x_k \in \mathcal{S}_k$, we can rewrite (3) and (4) as

$$\begin{array}{ll} \min\limits_{x, x_1, \ldots, x_p} & \langle c, x\rangle \\ \text{subject to} & Ax = b, \\ & x_k = H_k x, \quad k = 1, \ldots, p, \\ & x_k \in \mathcal{S}_k, \qquad k = 1, \ldots, p, \end{array} \tag{6}$$

and

$$\begin{array}{ll} \max\limits_{y, z_1, \ldots, z_p, v_1, \ldots, v_p} & \langle b, y\rangle \\ \text{subject to} & A^T y + \sum\limits_{k=1}^{p} H_k^T v_k = c, \\ & z_k - v_k = 0, \quad k = 1, \ldots, p, \\ & z_k \in \mathcal{S}_k, \qquad k = 1, \ldots, p. \end{array} \tag{7}$$

3.2 Homogeneous Self-Dual Embedding

For notational simplicity, let $\mathcal{S} := \mathcal{S}_1 \times \cdots \times \mathcal{S}_p$ and define

$$s := \begin{bmatrix} x_1 \\ \vdots \\ x_p \end{bmatrix}, \quad z := \begin{bmatrix} z_1 \\ \vdots \\ z_p \end{bmatrix}, \quad v := \begin{bmatrix} v_1 \\ \vdots \\ v_p \end{bmatrix}, \quad H := \begin{bmatrix} H_1 \\ \vdots \\ H_p \end{bmatrix}.$$

When strong duality holds for (6) and (7), the following KKT conditions are necessary and sufficient for optimality of the tuple (x, s, y, v, z):

- (x, s) is primal feasible; adding optimal slack variables $r = 0$ and $w = 0$, this amounts to
$$\begin{array}{rr} Ax - r = b, & r = 0, \\ s + w = Hx, & w = 0, \quad s \in \mathcal{S}. \end{array} \tag{8}$$
- (y, v, z) is dual feasible; adding an optimal slack variable $h = 0$, this amounts to
$$\begin{array}{rr} A^T y + H^T v + h = c, & h = 0, \\ z - v = 0, & z \in \mathcal{S}. \end{array} \tag{9}$$
- The duality gap is zero, *i.e.*
$$c^T x - b^T y = 0. \tag{10}$$

The idea behind the homogeneous self-dual embedding (Ye et al., 1994) is to introduce two non-negative and complementary variables τ and κ in addition to the variables zero variables r, w and h introduced above, and embed the KKT conditions (8), (9) and (10) into the linear system

$$\begin{bmatrix} h \\ z \\ r \\ w \\ \kappa \end{bmatrix} = \begin{bmatrix} 0 & 0 & -A^T & -H^T & c \\ 0 & 0 & 0 & I & 0 \\ A & 0 & 0 & 0 & -b \\ H & -I & 0 & 0 & 0 \\ -c^T & 0 & b^T & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ s \\ y \\ v \\ \tau \end{bmatrix}. \tag{11}$$

Any solution of this embedding can be used to recover an optimal solution for (6)-(7), or provide a certificate of primal or dual infeasibility (see O'Donoghue et al., 2016a).

Letting $n_d = \sum_{k=1}^{p} |\mathcal{C}_k|^2$, defining $\mathcal{K} := \mathbb{R}^{n^2} \times \mathcal{S} \times \mathbb{R}^m \times \mathbb{R}^{n_d} \times \mathbb{R}_+$, and writing

$$u := \begin{bmatrix} x \\ s \\ y \\ v \\ \tau \end{bmatrix}, \quad v := \begin{bmatrix} h \\ z \\ r \\ w \\ \kappa \end{bmatrix}, \quad Q := \begin{bmatrix} 0 & 0 & -A^T & -H^T & c \\ 0 & 0 & 0 & I & 0 \\ A & 0 & 0 & 0 & -b \\ H & -I & 0 & 0 & 0 \\ -c^T & 0 & b^T & 0 & 0 \end{bmatrix}$$

to further ease the notation, the decomposed primal-dual pair of SDPs (6)-(7) can be recast as the feasibility problem

$$\begin{array}{ll} \text{find} & (u, v) \\ \text{subject to} & v = Qu, \\ & (u, v) \in \mathcal{K} \times \mathcal{K}^*, \end{array} \tag{12}$$

where $\mathcal{K}^* = \{0\}^{n^2} \times \mathcal{S}^* \times \{0\}^m \times \{0\}^{n_d} \times \mathbb{R}_+$ is the dual of the cone $\mathcal{K}$ (here, $\mathcal{S}^*$ is the dual of $\mathcal{S}$, and $\{0\}^p$ is the p-dimensional zero vector).

4. ADMM FOR THE HOMOGENEOUS SELF-DUAL EMBEDDING

4.1 Basic algorithm

Problem (12) is in the same form considered by O'Donoghue et al. (2016a), so we can directly apply their ADMM algorithm. The k-th iteration of the algorithm consists of the following three steps (O'Donoghue et al., 2016a), where $\Pi_{\mathcal{K}}$ denotes projection on the cone $\mathcal{K}$:

$$\hat{u}^{k+1} = (I+Q)^{-1}(u^k + v^k), \tag{13a}$$
$$u^{k+1} = \Pi_{\mathcal{K}}(\hat{u}^{k+1} - v^k), \tag{13b}$$
$$v^{k+1} = v^k - \hat{u}^{k+1} + u^{k+1}. \tag{13c}$$

Note that (13b) is inexpensive, since $\mathcal{K}$ is the cartesian product of simple cones (zero, free and non-negative cones) and small PSD cones, and can be efficiently carried out in parallel. The third step is also computationally inexpensive and parallelizable. On the contrary, although the preferred factorization of $I+Q$ (or its inverse) can be cached before starting the iterations, a direct implementation of (13a) can be computationally intensive since Q is a very large matrix. However, Q is highly structured and sparse; in the next sections, we show how its special structure can be exploited to devise a series of block-eliminations that speed up the affine projection in (13a).

4.2 Solving the linear system

The affine projection step (13a) requires the solution of a linear system in the form

$$\begin{bmatrix} I & \hat{A}^T & \hat{c} \\ -\hat{A} & I & \hat{b} \\ -\hat{c}^T & -\hat{b}^T & 1 \end{bmatrix} \begin{bmatrix} \hat{u}_1 \\ \hat{u}_2 \\ \hat{u}_3 \end{bmatrix} = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}, \tag{14}$$

where

$$\hat{A} = \begin{bmatrix} -A & 0 \\ -H & I \end{bmatrix}, \qquad \hat{c} = \begin{bmatrix} c \\ 0 \end{bmatrix}, \qquad \hat{b} = \begin{bmatrix} -b \\ 0 \end{bmatrix}.$$

Note that u_3 and ω_3 are scalars. Letting

$$M := \begin{bmatrix} I & \hat{A}^T \\ -\hat{A} & I \end{bmatrix}, \qquad \zeta := \begin{bmatrix} \hat{c} \\ \hat{b} \end{bmatrix},$$

and carrying out block elimination on (14) we obtain

$$(M + \zeta\zeta^T) \begin{bmatrix} \hat{u}_1 \\ \hat{u}_2 \end{bmatrix} = \begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix} - \omega_3 \zeta. \tag{15a}$$
$$\hat{u}_3 = \omega_3 + \hat{c}^T \hat{u}_1 + \hat{b}^T \hat{u}_2. \tag{15b}$$

Moreover, the matrix inversion lemma (Boyd and Vandenberghe, 2004, Appendix C.4.3) allows us to write the solution of (15a) as

$$\begin{bmatrix} \hat{u}_1 \\ \hat{u}_2 \end{bmatrix} = \left[M^{-1} - \frac{(M^{-1}\zeta)\zeta^T M^{-1}}{1 + \zeta^T (M^{-1}\zeta)} \right] \left(\begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix} - \omega_3 \zeta \right). \tag{16}$$

Note that the vector $M^{-1}\zeta$ and the scalar $1 + \zeta^T(M^{-1}\zeta)$ only depend on the problem data, and can be cached before starting the ADMM iterations. Consequently, updating $\hat{u}_1$, $\hat{u}_2$ and $\hat{u}_3$ at each iteration requires:

(1) the solution of the "inner" linear system to compute

$$M^{-1} \left(\begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix} - \omega_3 \zeta \right).$$

(2) a series of inexpensive vector inner products and scalar-vector operations in (15b) and (16).

4.3 Solving the "inner" linear system

Recalling the definition of M, computing (16) requires the solution of a linear system of the form

$$\begin{bmatrix} I & \hat{A}^T \\ -\hat{A} & I \end{bmatrix} \begin{pmatrix} \hat{u}_1 \\ \hat{u}_2 \end{pmatrix} = \begin{pmatrix} \hat{\omega}_1 \\ \hat{\omega}_2 \end{pmatrix}. \tag{17}$$

Block elimination leads to

$$\hat{u}_1 = \hat{\omega}_1 - \hat{A}^T \hat{u}_2, \tag{18a}$$
$$(I + \hat{A}\hat{A}^T)\hat{u}_2 = \hat{A}\hat{\omega}_1 + \hat{\omega}_2. \tag{18b}$$

Recalling the definition of $\hat{A}$ and recognizing that $D := H^T H = \sum_{k=1}^{p} H_k^T H_k$ is a diagonal matrix, we also have

$$I + \hat{A}\hat{A}^T = \begin{bmatrix} I + D + A^T A & -H^T \\ -H & 2I \end{bmatrix}. \tag{19}$$

Given the special structure of this matrix, block elimination can be used again to solve (18b). Simple algebraic manipulations show that the only matrix to be factorized (or inverted) before starting the ADMM iterations is

$$I + \frac{1}{2} D + A^T A. \tag{20}$$

In fact, this matrix is of the "diagonal plus low rank" form, so the matrix inversion lemma can be used to reduce the size of the matrix to invert even further.

4.4 Summary of computational gains

The algorithm outlined in the previous sections is clearly more efficient than a direct application of the ADMM algorithm of O'Donoghue et al. (2016a) to the decomposed primal-dual pair of SDPs (6)-(7). In fact, the cost of the conic projection (13b) will be the same for both algorithms, but the sequence of block eliminations and applications of the matrix inversion lemma we have described reduces the cost of the affine projection step. In the algorithm, we only need to factor an $m \times m$ matrix, instead of the matrix Q of size $(n^2 + 2n_d + m + 1) \times (n^2 + 2n_d + m + 1)$.

Furthermore, it can be checked that when we exploit the special structure of the matrix $I + Q$, the overall computational cost of (13a) coincides (at least to leading order) with the cost of the affine projection step when the algorithm of O'Donoghue et al. (2016a) is applied to the original primal-dual pair (1)-(2), before chordal decomposition. This means that our algorithm should also outperform the algorithm of O'Donoghue et al. (2016a) applied to the original primal-dual pair of SDPs (1)-(2): the cost of the affine projection is the same, but the conic projection in our algorithm is more efficient since we work with smaller positive semidefinite cones.

5. NUMERICAL SIMULATIONS

We have implemented our techniques in CDCS (Cone Decomposition Conic Solver) (Zheng et al., 2017). The code is available at

`https://github.com/oxfordcontrol/CDCS`.

In addition to the homogeneous self-dual embedding algorithm, CDCS also includes the primal and dual methods of Zheng et al. (2016). Currently, CDCS was tested on three sets of benchmark problems in SDPLIB (Borchers, 1999): *(1)* Four small and medium-sized SDPs (two

Table 1. Details of the SDPLIB problems considered in this work.

	Small and medium-size ($n \leq 100$)				Large-scale and sparse ($n \geq 800$)				Infeasible	
	theta1	theta2	qap5	qap9	maxG11	maxG32	qpG11	qpG51	infp1	infd1
Original cone size, n	50	100	26	82	800	2000	1600	2000	30	30
Affine constraints, m	104	498	136	748	800	2000	800	1000	10	10
Number of cliques, p	1	1	1	1	598	1499	1405	1675	1	1
Maximum clique size	50	100	26	82	24	60	24	304	30	30
Minimum clique size	50	100	26	82	5	5	1	1	30	30

Table 2. Results for some small and medium-sized SDPs in SDPLIB

		SeDuMi	SparseCoLO+ SeDuMi	SCS	CDCS (primal)	CDCS (dual)	Self-dual
theta1	Total time (s)	0.262	0.279	0.145	0.751	0.707	0.534
	Pre- time (s)	0	0.005	0.011	0.013	0.010	0.012
	Iterations	14	14	240	317	320	230
	Objective	2.300×10^1	2.300×10^1	2.300×10^1	2.299×10^1	2.299×10^1	2.303×10^1
theta2	Total time (s)	1.45	1.55	0.92	1.45	1.30	0.60
	Pre- time (s)	0	0.014	0.018	0.046	0.036	0.031
	Iterations	15	15	500	287	277	110
	Objective	3.288×10^1	3.288×10^1	3.288×10^1	3.288×10^1	3.288×10^1	3.287×10^1
qap5	Total time (s)	0.365	0.386	0.412	0.879	0.748	1.465
	Pre- time (s)	0	0.006	0.026	0.011	0.009	0.009
	Iterations	12	12	320	334	332	783
	Objective	-4.360×10^2	-4.360×10^2	-4.359×10^2	-4.360×10^2	-4.364×10^2	-4.362×10^2
qap9	Total time (s)	6.291	6.751	3.261	7.520	7.397	1.173
	Pre- time (s)	0	0.012	0.010	0.064	0.036	0.032
	Iterations	25	25	2000	2000	2000	261
	Objective	-1.410×10^3	-1.410×10^3	-1.409×10^3	-1.407×10^3	-1.409×10^3	-1.410×10^3

Table 3. Results for some large-scale sparse SDPs

		SeDuMi	SparseCoLO+ SeDuMi	SCS	CDCS (primal)	CDCS (dual)	Self-dual
maxG11	Total time (s)	92.0	9.83	160.5	126.6	114.1	23.9
	Pre- time (s)	0	2.39	0.07	3.33	4.28	2.45
	Iterations	13	15	1860	1317	1306	279
	Objective	6.292×10^2	6.292×10^2	6.292×10^2	6.292×10^2	6.292×10^2	6.295×10^2
maxG32	Total time (s)	1.385×10^3	577.4	2.487×10^3	520.0	273.8	87.4
	Pre- time (s)	0	7.63	0.589	53.9	55.6	30.5
	Iterations	14	15	2000	1796	943	272
	Objective	1.568×10^3	1.568×10^3	1.568×10^3	1.568×10^3	1.568×10^3	1.568×10^3
qpG11	Total time (s)	675.3	27.3	1.115×10^3	273.6	92.5	32.1
	Pre- time (s)	0	11.2	0.57	6.26	6.26	3.85
	Iterations	14	15	2000	1355	656	304
	Objective	2.449×10^3	2.449×10^3	2.449×10^3	2.449×10^3	2.449×10^3	2.450×10^3
qpG51	Total time (s)	1.984×10^3	–	2.290×10^3	1.627×10^3	1.635×10^3	538.1
	Pre- time (s)	0	–	0.90	10.82	12.77	7.89
	Iterations	22	–	2000	2000	2000	716
	Objective	1.182×10^3	–	1.288×10^3	1.183×10^3	1.186×10^3	1.181×10^3

Lovász ϑ number problems, theta1 and theta2, and two quadratic assignment problems, qap5 and qap9); *(2)* Four large-scale sparse SDPs (two max-cut problems, maxG11 and maxG32, and two SDP relaxations of box-constrained quadratic programming problems, qpG11 and qpG51); *(3)* Two infeasible SDPs (infp1 and infd1).

Table 1 reports the problem dimensions and some chordal decomposition details. The performance of our self-dual method is compared to that of the interior-point solver SeDuMi (Sturm, 1999), of the first-order solver SCS (O'Donoghue et al., 2016b), and of the primal and dual methods in CDCS (Zheng et al., 2016). SparseCoLO (Fujisawa et al., 2009) was used as a preprocessor for SeDuMi. The solution returned by SeDuMi is of high accuracy, so we can assess the quality of the solution computed by CDCS. Instead, SCS is a high-performance first-order solver for general conic programs, so we can assess the advantages of chordal decomposition. Note that SeDuMi should not be compared to the other solvers on CPU time, because the latter only aim to achieve moderate accuracy. In all cases we set the termination tolerance for CDCS and SCS to $\epsilon_{\text{tol}} = 10^{-4}$ with a maximum of 2000 iterations, while we used SeDuMi's default options. All experiments were carried out on a computer with a 2.8 GHz Intel(R) Core(TM) i7 CPU and 8GB of RAM.

Our numerical results are summarized in Tables 2–5. In all feasible cases, the objective value returned by our self-dual algorithm was within 0.6% of the (accurate) value returned by SeDuMi. The small and medium-sized dense SDPs were solved in comparable CPU time by all solvers (Table 2). For the four large-scale sparse SDPs, our self-dual method was faster than either SeDuMi or SCS (Table 3). As expected, problems with smaller maximum clique size, such as maxG11, maxG32, and qpG11, were solved more efficiently (less than 100 s using our self-dual algorithm). Note that the conversion techniques in SparseCoLO can give speedups in some cases, but the failure to solve the problem qpG51—due to memory overflow caused by the

Table 4. Results for two infeasible SDPs

		SeDuMi	SparseCoLO +SeDuMi	SCS	Self-dual
infp1	Total time (s)	0.063	0.083	0.062	0.18
	Pre- time (s)	0	0.010	0.016	0.010
	Iterations	2	2	20	104
	Status	Infeasible	Infeasible	Infeasible	Infeasible
infd1	Total time (s)	0.125	0.140	0.050	0.144
	Pre- time (s)	0	0.009	0.013	0.009
	Iterations	4	4	40	90
	Status	Infeasible	Infeasible	Infeasible	Infeasible

Table 5. CPU time per iteration (s)

	SCS	CDCS (primal)	CDCS (dual)	Self-dual
theta1	6×10^{-4}	2.3×10^{-3}	2.2×10^{-3}	2.3×10^{-3}
theta2	1.8×10^{-3}	5.1×10^{-3}	4.7×10^{-3}	5.5×10^{-3}
qap5	1.2×10^{-3}	2.6×10^{-3}	2.2×10^{-3}	1.9×10^{-3}
qap9	1.5×10^{-3}	3.6×10^{-3}	3.7×10^{-3}	4.2×10^{-3}
maxG11	0.086	0.094	0.084	0.077
maxG32	1.243	0.260	0.231	0.209
qpG11	0.557	0.198	0.132	0.093
qpG51	1.144	0.808	0.811	0.741

large number of consensus constraints in the converted problem—highlights the drawbacks.

As shown in Table 4, our self-dual algorithm successfully detects infeasible problems, while our previous first-order methods (CDCS-primal and CDCS-dual) do not have this ability. Finally, Table 5 lists the average CPU time per iteration for the first-order algorithms. When comparing the results, it should be kept in mind that our codes are written in MATLAB, while SCS is implemented in C. Nevertheless, we still see that our self-dual algorithm is faster than SCS for the large-scale sparse SDPs (maxG11, maxG32, qpG11 and qpG51), which is expected since the conic projection step is more efficient with smaller PSD cones.

6. CONCLUSION

In this paper, we formulated the homogeneous self-dual embedding of a primal-dual pair of sparse SDPs whose conic constraints are decomposed using chordal decomposition techniques, thereby extending the conversion methods developed in previous work by the authors (Zheng et al., 2016). We also showed how the special structure of our homogeneous self-dual formulation can be exploited to develop an efficient ADMM algorithm, which we implemented in the conic solver CDCS. Our numerical simulations on some benchmark problems from the library SDPLIB show that our self-dual algorithm can give speedups compared to interior-point solvers such as SeDuMi—even when chordal sparsity is exploited using SparseCoLO—and also compared to the state-of-the-art first-order solver SCS.

REFERENCES

Agler, J., Helton, W., McCullough, S., and Rodman, L. (1988). Positive semidefinite matrices with a given sparsity pattern. *Linear Alg. Its Appl.*, 107, 101–149.

Andersen, M., Dahl, J., Liu, Z., and Vandenberghe, L. (2011). Interior-point methods for large-scale cone programming. *Optim. for machine learning*, 55–83.

Andersen, M.S., Dahl, J., and Vandenberghe, L. (2010). Implementation of nonsymmetric interior-point methods for linear optimization over sparse matrix cones. *Math. Program. Computation*, 2, 167–201.

Borchers, B. (1999). SDPLIB 1.2, a library of semidefinite programming test problems. *Optim. Methods Softw.*, 11(1-4), 683–690.

Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press.

Fujisawa, K., Kim, S., Kojima, M., Okamoto, Y., and Yamashita, M. (2009). Users manual for SparseCoLO: Conversion methods for sparse conic-form linear optimization problems. Technical report, Research Report B-453, Tokyo Institute of Technology, Japan.

Fukuda, M., Kojima, M., Murota, K., and Nakata, K. (2001). Exploiting sparsity in semidefinite programming via matrix completion I: General framework. *SIAM J. Optim.*, 11(3), 647–674.

Grone, R., Johnson, C.R., Sá, E.M., and Wolkowicz, H. (1984). Positive definite completions of partial hermitian matrices. *Linear Alg. Its Appl.*, 58, 109–124.

Helmberg, C., Rendl, F., Vanderbei, R.J., and Wolkowicz, H. (1996). An interior-point method for semidefinite programming. *SIAM J. Optim.*, 6(2), 342–361.

Kalbat, A. and Lavaei, J. (2015). A fast distributed algorithm for decomposable semidefinite programs. In *IEEE 54th Conf. Decision Control (CDC)*, 1742–1749.

Kim, S., Kojima, M., Mevissen, M., and Yamashita, M. (2011). Exploiting sparsity in linear and nonlinear matrix inequalities via positive semidefinite matrix completion. *Math. Program.*, 129(1), 33–68.

Madani, R., Kalbat, A., and Lavaei, J. (2015). ADMM for sparse semidefinite programming with applications to optimal power flow problem. In *IEEE 54th Conf. Decision Control (CDC)*, 5932–5939.

O'Donoghue, B., Chu, E., Parikh, N., and Boyd, S. (2016a). Conic optimization via operator splitting and homogeneous self-dual embedding. *J. Optimiz. Theory App.*, 169(3), 1042–1068.

O'Donoghue, B., Chu, E., Parikh, N., and Boyd, S. (2016b). SCS: Splitting conic solver, version 1.2.6. `https://github.com/cvxgrp/scs`.

Sturm, J.F. (1999). Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optim. Methods Softw.*, 11(1-4), 625–653.

Sun, Y., Andersen, M.S., and Vandenberghe, L. (2014). Decomposition in conic optimization with partially separable structure. *SIAM J. Optim.*, 24(2), 873–897.

Vandenberghe, L. and Andersen, M.S. (2014). Chordal graphs and semidefinite optimization. *Found. Trends Optim.*, 1(4), 241–433.

Ye, Y. (2011). *Interior point algorithms: theory and analysis*, volume 44. John Wiley & Sons.

Ye, Y., Todd, M.J., and Mizuno, S. (1994). An $\mathcal{O}\sqrt{n}l$-iteration homogeneous and self-dual linear programming algorithm. *Math. Oper. Res.*, 19(1), 53–67.

Zheng, Y., Fantuzzi, G., Papachristodoulou, A., Goulart, P., and Wynn, A. (2016). Fast ADMM for semidefinite programs with chordal sparsity. *arXiv:1609.06068v2 [math-OC]*.

Zheng, Y., Fantuzzi, G., Papachristodoulou, A., Goulart, P., and Wynn, A. (2017). CDCS: Cone decomposition conic solver, version 1.0. `https://github.com/oxfordcontrol/CDCS`.