2021 60th IEEE Conference on Decision and Control (CDC)
December 13-15, 2021. Austin, Texas

Globally Convergent Low Complexity Algorithms for Semidefinite Programming

Biel Roig-Solvas M. Sznaier

2021 60th IEEE Conference on Decision and Control (CDC) | 978-1-6654-3659-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/CDC45484.2021.9682942

***Abstract*— Semidefinite programs (SDP) are a staple of today's systems theory, with applications ranging from robust control to systems identification. However, current state-of-the art solution methods have poor scaling properties, and thus are limited to relatively moderate size problems. Recently, several approximations have been proposed where the original SDP is relaxed to a sequence of lower complexity problems (such as linear programs (LPs) or second order cone programs (SOCPs)). While successful in many cases, there is no guarantee that these relaxations converge to the global optimum of the original program. Indeed, examples exists where these relaxations "get stuck" at suboptimal solutions. To circumvent this difficulty in this paper we propose an algorithm to solve SDPs based on solving a sequence of LPs or SOCPs, guaranteed to converge in a finite number of steps to an ϵ-suboptimal solution of the original problem. We further provide a bound on the number of steps required, as a function of ϵ and the problem data.**

I. Introduction

Semidefinite Programs (SDPs) of the form:

$$\begin{aligned} X^*_{PSD} = \underset{X \succeq 0}{\text{argmin}} \quad & \text{Tr}\left(C^T X\right) \\ \text{s.t.} \quad & \text{Tr}\left(A_i^T X\right) = b_i \quad i = 1, \dots, M \end{aligned} \tag{1}$$

have become a staple of control theory, leading to efficient solutions to a wide range of problems [1], [2]. However, in spite of this success, SDP has found limited applicability in applications areas involving large date sets. To a large extent, this limitation stems from the poor scaling properties of conventional SDP solvers. For instance, computational complexity of interior point methods scales as mn^3, where n and m denote the number of variables and constraints [3].

Several approaches have been proposed to address these challenges by exploiting specific features of the problem, such as sparsity [4], [5], structure of the constraints [6], as well as first-order methods [7]. However, these methods do not apply to general scenarios. An alternative approach seeks to obtain lower complexity relaxations by replacing the semi-definite constraints with linear or second order cone constraints [8], [9], leading to an algorithm based that alternates between Cholesky decompositions and linear (LP) or second order cone (SOCP) programs. However, while successful in many scenarios, there is no guarantee that it will converge to the solution of the original SDP. Indeed, there are examples where the approach does not converge to the global minimum.

To address these difficulties, in this paper we propose a new algorithm, based upon a combination of Cholesky factorizations and LPs of SOCPs. Our main theoretical result shows that this algorithm is guaranteed to converge, in polynomial time, to an ϵ-optimal solution a generic SDP, thus answering a question left open in [8], [9]. We further provide a bound on the number of iterations needed in terms of ϵ and the problem data. To the best of our knowledge, this is the first globally convergent SDP algorithm based on LP and SOCP programs. These results are illustrated with some benchmark examples taken from the SDPLib data set that challenge commonly used packages such as CVX.

This work was partially supported by NSF grants CNS–1646121, ECCS–1808381 and CNS–2038493, AFOSR grant FA9550-19-1-0005, and ONR grant N00014-21-1-2431. The authors are with the ECE Department, Northeastern University, Boston, MA 02115. emails: {biel,msznaier}@ece.neu.edu

II. Background Results

A. Interior Point Methods for SDPs and the Central Path

First formulated by Karmakar [10], interior point methods (IPM) have become widely adopted due to their guaranteed polynomial runtime [11]. These methods handle conic constraints by adding to the cost a "barrier" that tends to infinity when approaching the boundary of the feasible set. To prevent numerical instability, IPMs solve a sequence of optimization problems in which the barrier is weighted by a factor $1/t$, where t is increased until ϵ-optimality is reached. In the case of SDPs, the most widely used barrier function is the negative log-determinant, which leads to problems of the form:

$$\begin{aligned} X_t = \underset{X}{\text{minimize}} \quad & \text{Tr}\left(C^T X\right) - \frac{1}{t}\log\left(|X|\right) \\ \text{s.t.} \quad & \text{Tr}\left(A_i^T X\right) = b_i \quad i = 1, \dots, M \end{aligned} \tag{2}$$

The curve defined by the optimizers X_t of (2) as a function of $t > 0$ is called the Central Path of the problem. As $t \to \infty$, X_t converges to the optimizer of (1), X^*_{PSD}. Moreover, due to duality theory, the elements of the central path satisfy the following inequality:

$$\text{Tr}\left(C^T X_t\right) \geq \text{Tr}\left(C^T X^*_{PSD}\right) \geq \text{Tr}\left(C^T X_t\right) - N/t \tag{3}$$

which provides an optimality bound at any point in the path.

B. DD and SDD relaxations of Semidefinite Programming

In [9], the authors proposed a relaxation for general SDPs based on replacing the positive semidefinite constraints by lower complexity ones involving diagonally-dominant and scaled diagonally-dominant matrices, defined below:

Definition 1: A symmetric matrix X with elements $X(i,j)$ is diagonally-dominant (DD) if

$$X(i,i) \;\geq\; \sum_{j \neq i} |X(i,j)| \quad \forall i$$

Definition 2: A symmetric matrix X is scaled diagonally-dominant (SDD) if there exist a positive diagonal matrix D and a DD matrix Y such that $X = DYD$.

From these definitions it follows that $DD_N \subset SDD_N \subset PSD_N$, where DD_N, SDD_N and PSD_N denote the cones of $N \times N$ DD, SDD and Positive Semi-Definite (PSD) matrices. Thus, relaxations of the SDP (1) can be obtained by simply replacing the constraint $X \succeq 0$ with the stronger ones $X \in DD_N$ or $X \in SDD_N$. The following results, adapted from [8], [12] provides an alternative characterization of DD and SDD matrices that was used in [8] to show that these relaxations indeed lead to LPs or SOCPs with lower complexity than the original SDP. Define the mapping $\Psi_{i,j}$ from 2×2 matrices to $N \times N$ matrices:

$$\Psi_{i,j}(\mathcal{M}) = \bar{\mathcal{M}} \quad \text{where} \begin{cases} \bar{\mathcal{M}}(\{i,j\},\{i,j\}) = \mathcal{M} \\ 0 \quad \text{otherwise.} \end{cases}$$

i.e. the $\{i,j\}$ sub-matrix of $\bar{\mathcal{M}}$ is $\mathcal{M}$, and the other entries of $\bar{\mathcal{M}}$ are 0. $\Psi_{i,j}(.)$ allows for characterizing DD and SDD matrices in terms of "exploded" 2×2 matrices as follows:

Lemma 1 ([8], [12]):

$$Y \in DD_N \iff Y = \sum_{i,j}^{N} \Psi_{i,j}(\mathcal{M}_{i,j}), \quad \mathcal{M}_{i,j} \in DD_2$$

Similarly,

$$Y \in SDD_N \iff Y = \sum_{i,j}^{N} \Psi_{i,j}(\mathcal{M}_{i,j}), \quad \mathcal{M}_{i,j} \succeq 0$$

Thus, enforcing the constraint $X \in DD_N(SDD)_N$ indeed reduces to a set of linear (second order cone) constraints.

C. Iterative Basis Update

While replacing the PSD constraint in (1) with the stronger one $X \in DD_N$ or $X \in SDD_N$ leads to a computationally cheaper optimization, the solution to these relaxed problems can be far from the true optimum. To alleviate this [8] proposed an iterative algorithm with improve performance, based on alternating between solving a sequence of DD/SDD problems and performing Cholesky factorizations. Briefly, the idea is to solve at step k a problem of the form

$$\begin{aligned} & X_k^*(U_{k-1}) = \underset{X,Y}{\operatorname{argmin}} \ \operatorname{Tr}\left(C^T X\right) \\ & \text{s.t. } \operatorname{Tr}\left(A_i^T X\right) = b_i \quad i = 1,\dots,M \\ & U_{k-1}^T Y U_{k-1} = X, \quad Y \in DD_N \ / \ SDD_N \end{aligned} \tag{4}$$

where U_{k-1} is a Cholesky factor of the previous solution, e.g. $X_{k-1} = U_{k-1}^T U_{k-1}$. Note that since $I \in DD_N(SDD_N)$, the previous iterate X_{k-1}^*, is always a feasible solution of (4). Hence the algorithm generates a sequence of solutions with non-increasing cost. This sequence, however, is not guaranteed to converge to the optimizer of the original SDP (1). Indeed, in numerical tests the algorithm tends to converge to strictly suboptimal values for all medium to large size problems ($N >> 10$). This is precisely what motivates the present paper. Our main result shows that indeed, it is possible to obtain an LP/SOCP based globally convergent algorithm.

III. Proposed Algorithm

In this section we present a globally convergent algorithm for solving the SDP (1) based on DD and SDD programs. The algorithm is split in two phases. The first phase, the *decrease phase*, consists of solving a sequence of DD/SDD programs, exactly as in [8]. As noted above, this sequence tends to stagnate on a suboptimal objective cost as the iterates approach the boundary of the PSD cone and their conditioning worsens. To prevent this, a second phase of the algorithm that consists of a series of steps designed to improve the iterates' conditioning starts after the decrease phase. We call these steps *centering* steps, as they guide the iterates towards the center path of the SDP by solving a sequence of analytic centerings on the DD/SDD set. These centering steps constitute the *centering phase* of the algorithm.

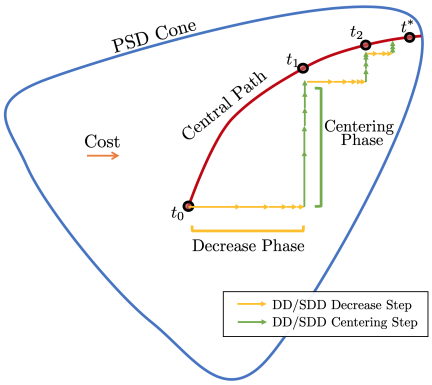

Fig. 1: The algorithm alternates between a *cost decreasing phase* and a *centering phase* that brings the iterate to a point ϵ_c-close to the central path. After a finite number of phases, the iterate is brought to a point ϵ_c-close to the central path with parameter $t_\kappa \geq t^*$, guaranteeing ϵ_g-convergence to the solution of the SDP.

Figure 1 shows a graphical description of the proposed algorithm. In the *decrease phase*, a sequence of problems of the form (4) are solved, decreasing the cost. After a fixed number s_d of decrease steps[1], the *centering phase* starts and a sequence of problems of the form (5) are solved :

$$\begin{aligned} & X_l(U_{l-1}) = \underset{X,Y}{\operatorname{argmin}} \ -\phi(Y) \ \text{ s.t.} \\ & \operatorname{Tr}\left(A_i^T X\right) = b_i \quad i = 1,\dots,M \\ & \operatorname{Tr}\left(C^T X\right) = \operatorname{Tr}\left(C^T X_{l-1}\right) \\ & U_{l-1}^T Y U_{l-1} = X, \quad Y \in DD_N \ / \ SDD_N \end{aligned} \tag{5}$$

Here the function $-\phi(Y)$ is the logarithmic barrier of the DD/SDD sets and U_{l-1} is the Cholesky factor of X_{l-1}. The sequence of centering steps converges to a point ϵ_c close to the central path of the SDP, i.e. the optimizer of the analytic centering defined on the PSD cone. At the end of this sequence the conditioning of the current iterate has improved and due to the properties of the SDP central path, the optimality gap is given by (3). At this point, a new *decrease phase* starts and the algorithm keeps iterating between *decrease* and *centering* phases, as outlined in Algorithm 1, until it converges. In the next section. we will prove that the algorithm converges to an ϵ-suboptimal solution of (1) in polynomial time and provide a bound on the number of iterations as a function of ϵ and the problem data.

[1]The theoretical guarantees developed in Section IV hold for any value of $s_d \geq 1$. Thus, in the sequel we assume $s_d = 1$ unless otherwise stated. The impact the choice s_d has on the algorithm performance is analyzed in the full version of the paper, available from the authors.

Algorithm 1: Globally Convergent DD/SDD Algorithm

Result: ϵ_g-optimal X^*
Initialize optimality gap as $g_{SDP} = \infty$; Initialize X_0;
$k = 0$;
while $g_{SDP} > \epsilon_g$ **do**
  % Start Decrease Phase
  for *Number of decrease steps* $s_d \geq 1$ **do**
    $k = k + 1$; Compute Cholesky Factors of X_{k-1};
    Solve for $X_k(U_{k-1})$ as in (4);
  end
  Update optimality gap g_{SDP};
  % Start Centering Phase
  Initialize centering gap $g_C = \infty$; $l = 0$, $X_{k,l} = X_k$;
  while $g_C > \epsilon_C$ **do**
    $l = l + 1$; Compute Cholesky Factors of $X_{k,l-1}$;
    Solve for $X_{k,l}(U_{k,l-1})$ as in (5);
    Update centering gap g_C;
  end
  Update optimality gap g_{SDP}; $X_k = X_{k,l}$;
end
$X^* = X_k$;

IV. Convergence Proofs

In this section we develop the proof of global convergence for Algorithm 1. This proof rests on the two following results. The first is the that the Centering Phase converges to an iterate ϵ_c-close to the central path in polynomial time. This will be established by finding a polynomial upper bound on the number of instances that problem (5) needs to be solved to achieve ϵ_c-optimality. The second result is a formal proof that Algorithm 1 reaches an iterate X^* with an optimality gap of ϵ_g with respect to the optimizer of (1) in at most a polynomial number of combined *decrease* and *centering* phases. In order to establish these results, we need the following assumptions:

A1: The data matrices C and A_i all satisfy that:
$$\begin{aligned} \text{Tr}\left(A_i^T A_j\right) = \text{Tr}\left(A_i^T C\right) = 0 \quad \forall 1 \leq i,j \leq M \\ ||C||_F = ||A_i||_F = 1 \quad \forall i = 1,\ldots,M \end{aligned}$$

A2: There exists at least one feasible X for (1) such that $X \succ 0$ (Slater's condition).

A3: The cost function evaluated at the optimizer of (1) satisfies $\text{Tr}(C^T X^*_{PSD}) > -\infty$.

Assumption A1 can be made to hold trivially for any SDP by orthogonalization and projection; A2 guarantees strong duality; and A3 guarantees that the optimal cost is finite.

A. Convergence of the Centering Phase

The goal of this proof is to show that a sequence of problems of the form (5) converge to the optimizer of the PSD analytic centering, which we define as:

$$\begin{aligned} X\left(U_{k-1}\right) = \underset{X,Y}{\text{argmin}} \; -h(Y), \quad \text{s.t.} \\ \text{Tr}\left(A_i^T X\right) = b_i \quad i = 1,\ldots,M \\ \text{Tr}\left(C^T X\right) = \text{Tr}\left(C^T X_{k-1}\right) \\ U_{k-1}^T Y U_{k-1} = X, \quad Y \succeq 0 \end{aligned} \tag{6}$$

where the objective function is defined as:

$$h(Y) = (N-1)\log\left(|Y|\right) - N(N-1)\log(N-1) \tag{7}$$

i.e. a scaled and shifted variant of the common logarithmic barrier for the PSD cone, the negative log-determinant. Next we define the barriers for the DD and SDD cones used in (5). Motivated by the decompositions in Lemma 1, we will consider the following logarithmic barriers:

$$\begin{aligned} \phi_{DD}(\mathcal{M}) = \frac{1}{2}\sum_{i,j>i}^{N} \log\left(\mathcal{M}_{i,j}(1,1)^2 - \mathcal{M}_{i,j}(1,2)^2\right) \\ + \log\left(\mathcal{M}_{i,j}(2,2)^2 - \mathcal{M}_{i,j}(1,2)^2\right) \\ \phi_{SDD}(\mathcal{M}) = \sum_{i,j>i}^{N} \log\left(\mathcal{M}_{i,j}(1,1)\,\mathcal{M}_{i,j}(2,2) - \mathcal{M}_{i,j}(1,2)^2\right) \end{aligned} \tag{8}$$

It can be shown that these barrier functions are self-concordant with respect to the entries of $\mathcal{M}$.

Lemma 2: The following properties hold:
If $Y = \Psi(\mathcal{M}) \in DD_N$, then $-\phi_{DD}(\mathcal{M}) \geq -\phi_{SDD}(\mathcal{M})$.
If $Y = \Psi(\mathcal{M}) \in SDD_N$, then $-\phi_{SDD}(\mathcal{M}) \geq -h(Y)$.

Proof: Omitted for space reasons, can be found in the ArXiv version of the paper. ■

Corollary 1: If $Y = \Psi(\mathcal{M}) \in DD_N$, then $-\phi_{DD}(\mathcal{M}) \geq -\phi_{SDD}(\mathcal{M}) \geq -h(Y)$. Moreover, if $\mathcal{M}_{i,j} = \frac{1}{N-1} I_{2\times 2}$ for all $i < j$, then $Y = \Psi(\mathcal{M}) = I$ and $-\phi_{DD}(\mathcal{M}) = -\phi_{SDD}(\mathcal{M}) = -h(Y) = N(N-1)\log(N-1)$

The proof of convergence rests on the self-concordance of $-\phi(\mathcal{M})$ and the inequalities in the Lemma above. It uses the following properties of self-concordant functions [13]. Consider the problem of minimizing a convex self-concordant function $f(x)$ using a Newton method [13]. Denote by x the current iterate, x_+ the iterate after taking a Newton step from x, x^* global minimizer of $f(x)$ and by $\lambda(x)$ the Newton decrement of $f(x)$ evaluated at x. Finally take the line-search constants $\alpha \in (0, 0.5)$ and $\beta \in (0,1)$ and the variable $\eta = \frac{1-2\alpha}{4}$. Then we have the following [13]:

$$f(x_+) \leq f(x) - \alpha\beta \frac{\lambda^2(x)}{1+\lambda(x)} \tag{9}$$

$$\lambda(x) \leq \eta \quad \Longrightarrow \quad f(x_+) \leq f(x) - \alpha\lambda^2(x) \tag{10}$$

$$\lambda(x) \leq 0.68 \quad \Longrightarrow \quad f(x) \geq f(x^*) \geq f(x) - \lambda^2(x) \tag{11}$$

Next, we will apply these properties to the specific scenario arising when solving problem (5) using a Newton method, with the variable Y parametrized as $Y = \Psi(\mathcal{M})$. Denote by $\mathcal{M}_0$ the set where all elements of the set are $\mathcal{M}_{i,j} = \frac{1}{N-1} I_{2\times 2}$, i.e. $Y_0 = \Psi(\mathcal{M}_0) = I$, which by construction is always a feasible solution of (5). Evaluating the Newton decrement for[2] $-\phi(\mathcal{M})|_{\mathcal{M}=\mathcal{M}_0}$ as $\lambda_\phi(\mathcal{M}_0)$, then (9) leads to the following lemma:

Lemma 3: Take U_{l-1} to be the Cholesky factors of X_{l-1}. Then if the Newton decrement of problem (5) satisfies $\lambda_\phi(\mathcal{M}_0) > \frac{\eta}{\sqrt{N-1}}$, its optimizer $X_l(U_{l-1})$ satisfies:

$$-(N-1)\log\left(|X_l|\right) \leq -(N-1)\log\left(|X_{l-1}|\right) - \xi$$

where ξ is a positive constant of the form:

$$\xi = \frac{\alpha\beta}{\sqrt{N-1}} \frac{\eta^2}{\sqrt{N-1}+\eta}$$

[2]As the proof applies to both DD and SDD cases, we drop the subscripts for notation clarity.

Similarly, (10) and (11) lead to:

Lemma 4: If the Newton decrement of (5) satisfies $\lambda_\phi(\mathcal{M}_0) \leq \frac{\eta}{\sqrt{N-1}}$, and the centering optimality gap between X_{l-1} and the optimizer X_* of (6), g_{l-1} is given by:

$$g_{l-1} = (N-1)\ (\log(|X_*|) - \log(|X_{l-1}|))$$

then g_l is upper-bounded by:

$$\begin{aligned}&\frac{N-1-\alpha}{N-1}\min(\eta^2, g_{l-1}) \geq g_l\\ &= (N-1)\ (\log(|X_*|) - \log(|X_l|))\end{aligned}$$

Lemmas 3 and 4 provide the foundation for the proof of polynomial complexity of the Centering Phase:

Theorem 1: The Centering Phase described in Algorithm 1 converges to ϵ_C-optimality to the optimizer of (6) as:

$$\epsilon_C \geq (N-1)\ (\log(|X_*|) - \log(|X_L|))$$

in at most L iterations, where L is given by:

$$\begin{aligned}L = &\lceil \frac{(N-1)\ (\log(|X_*|) - \log(|X_0|)) - \epsilon_C}{\xi} \rceil\\ &+ \lceil \frac{\log(\epsilon_C) - \log(\eta^2)}{\log(N-1-\alpha) - \log(N-1)} \rceil\end{aligned}$$

and X_0 is the starting point of the Centering Phase.

Proof: At each iteration the centering optimality gap is reduced, either by a fixed amount ξ if $\lambda_\phi(\mathcal{M}_0) > \frac{\eta}{\sqrt{N-1}}$ (Lemma 3), or by a multiplicative factor $\frac{N-1-\alpha}{N-1}$ if $\lambda_\phi(\mathcal{M}_0) \leq \frac{\eta}{\sqrt{N-1}}$ (Lemma 4). Bringing the centering gap below ϵ_C requires at most L_1 iterations for the fixed decrease, with $L_1 = \lceil ((N-1)\ (\log(|X_*|) - \log(|X_0|)) - \epsilon_C)/\xi \rceil$, or L_2 iterations for the relative decrease, where $L_2 \lceil \left(\log(\epsilon_C) - \log(\eta^2)\right) / \left(\log(N-1-\alpha) - \log(N-1)\right) \rceil$, leading to a total running time of at most $L_1 + L_2$ iterations:

$$\begin{aligned}L = L_1 + L_2 = &\lceil \frac{(N-1)\ (\log(|X_*|) - \log(|X_0|)) - \epsilon_C}{\xi} \rceil\\ &+ \lceil \frac{\log(\epsilon_C) - \log(\eta^2)}{\log(N-1-\alpha) - \log(N-1)} \rceil\end{aligned}$$

■

B. Overall Convergence of the Proposed Algorithm

In this section we provide a sketch of the proof of finite time convergence of the algorithm to an ϵ-suboptimal solution. Due to space constraints, some of the proofs are omitted, but they can be found in the ArXiv version of the paper. The main idea of the proof is to show that alternating between decreasing and centering phases leads to a sequence of solutions X_k which are identical to the ones obtained using an interior point algorithm to solve (2) for a specific sequence t_k that satisfies $t_k > \chi t_{k-1}$, where $\chi > 1$ is a constant that depends on the problem data. It follows that a desired value t^* (corresponding to a given optimality gap) can be found in at most $\kappa = \lceil \frac{\log(N/\epsilon^*) - \log(t_0)}{\log(\chi)} \rceil$ iterations. For simplicity, we assume first that $\epsilon_C = 0$, i.e. exact convergence of the *centering* phase, and then, at the end of this section, indicate how to extend these results to the case of $\epsilon_C > 0$.

The proof is divided into the following steps.

1) **Step 1:** Given X_k, the output after the k^{th} combined decrease and centering steps, find a value $\hat{t}_k$ such that $t_k \geq \hat{t}_k > t_{k-1}$ and the solution $\hat{X}_k$ of (2) with $t = \hat{t}_k$ satisfies $\text{Tr}\left(C^T \hat{X}_k\right) \geq \text{Tr}\left(C^T X_k\right)$, that is, the proposed algorithm generates a solution with a lower or equal cost that the one that would have been generated by an Interior Point method with $t = \hat{t}_k$.
2) **Step 2:** Use the result above, combined with the strictly decreasing property of the solution to (2) to establish that $t_k > \chi t_{k-1}$.

We start the proof of **step 1** by recasting the PSD centering problem (6) in its most simple form:

$$\begin{aligned}X_\zeta(c) = \underset{X \succeq 0}{\text{argmin}} \quad &-\log(|X|)\\ \text{s.t.} \quad &\text{Tr}\left(A_i^T X\right) = b_i \quad i = 1, \ldots, M\\ &\text{Tr}\left(C^T X\right) = c\end{aligned} \tag{12}$$

Note the similarity between (12) and (2), where the former is parametrized by the affine subspace $\text{Tr}\left(C^T X\right) = c$ and the latter by the trade-off weight $1/t$ in the cost function. Next we show that the optimizer of (12) is also an optimizer of (2) for a particular value of t:

Lemma 5: Given $X_\zeta(c)$ the optimizer of problem (12), the optimizer X_t of (2) is equivalent to $X_\zeta(c)$ when $t = -\tau$, where τ is the dual variable of (12) associated to the linear constraint $\text{Tr}\left(C^T X\right) = c$ evaluated at $X = X_\zeta(c)$.

The dual of problem (2) is:

$$\begin{aligned}Z_t, y_t = \underset{Z \succeq 0, y}{\text{argmin}} \quad &-b^T y\ - \frac{1}{t}\log(|Z|)\\ \text{s.t.} \quad &Z = C - \sum_{i=1}^{M} y_i A_i\end{aligned} \tag{13}$$

From duality, it follows that $X_t(t) = \frac{1}{t} Z_t^{-1}(t)$ and $y_t = \gamma_{i,t}$ [13], where $\gamma_{i,t}$ are the dual variables of the linear constraints of (2). Furthermore, it can be shown (see the ArXiv version) that, for any value of t in the range $t_0 \leq t \leq t^*$, the norm of the dual variables $\gamma_{i,t}$ is bounded above by $\sum_i \gamma_{i,t}^2 \leq \Theta$, where Θ is a finite positive constant that depends only on the problem data. The KKT conditions of (2) yield:

$$X_{k-1}^{-1} = X_{t_{k-1}}^{-1} = t_{k-1}\left(C - \sum \gamma_i A_i\right)$$

Hence, orthogonality of the data matrices C, A_i implies:

$$||X_{t_{k-1}}^{-1}||_F^2 = t_{k-1}^2\left(1 + \sum \gamma_i^2\right) \tag{14}$$

Next we introduce a feasible solution $\hat{Y}$ to problem (4). To do so, we make use of the following Lemma:

Lemma 6: If $||Y - I||_F^2 \leq 1$, then Y is scaled diagonally-dominant. Furthermore, if $||Y - I||_F^2 \leq \frac{2}{N+1}$, Y is diagonally-dominant.

Consider now a positive constant Φ that satisfies $\Phi \leq \{2/(N+1), 1\}$ for the DD and SDD case, respectively. Then defining $Q = U_{k-1}^{-T} C U_{k-1}^{-1}$, a feasible solution for (4) is given by $\hat{Y} = I - \sqrt{\Phi}\frac{Q}{||Q||_F}$. By construction this solution satisfies $||\hat{Y} - I||_F^2 = \Phi$, ensuring by Lemma 6 that $\hat{Y}$ is contained in the appropriate DD or SDD cone. Moreover, defining $\hat{X}$ as $\hat{X} = U_{k-1}^T \hat{Y} U_{k-1}$, the linear constraints evaluated at $\hat{X}$

satisfy:

$$\begin{aligned}&\mathrm{Tr}\left(A_i^T\hat{X}\right)=\mathrm{Tr}\left(A_i^TU_{k-1}^T\hat{Y}U_{k-1}\right)\\&=\mathrm{Tr}\left(A_i^TX_{k-1}\right)-\frac{\sqrt{\Phi}}{||Q||_F}\mathrm{Tr}\left(A_i^TU_{k-1}^TQU_{k-1}\right)\\&=\mathrm{Tr}\left(A_i^TX_{k-1}\right)-\frac{\sqrt{\Phi}}{||Q||_F}\mathrm{Tr}\left(A_i^TC\right)=\mathrm{Tr}\left(A_i^TX_{k-1}\right)=b_i\end{aligned}$$

due to the orthogonality between A_i and C (Assumption A1). Thus $\hat{Y}$ is a feasible solution of (4) with associated cost:

$$\begin{aligned}\hat{\mathcal{C}}_k&=\mathrm{Tr}\left(C^TU_{k-1}^T\hat{Y}U_{k-1}\right)\\&=\mathrm{Tr}\left(C^TX_{k-1}\right)-\frac{\sqrt{\Phi}}{||Q||_F}\mathrm{Tr}\left(C^TC\right)\\&=\mathrm{Tr}\left(C^TX_{k-1}\right)-\frac{\sqrt{\Phi}}{||Q||_F}\end{aligned}\tag{15}$$

where $||C||_F^2=1$ from Assumption A1. Using the properties of the Frobenius norm, the norm of Q can be bounded as $||Q||_F=||U_{k-1}^{-T}CU_{k-1}^{-1}||_F\leq||C||_F||X_{k-1}^{-1}||_F$. The matrix X_{k-1} is the optimizer of the Centering Phase at stage $k-1$ and thus by Lemma 5 satisfies $X_{k-1}=X_{t_{k-1}}$. Combining this with (14) yields: yields:

$$\begin{aligned}&\mathrm{Tr}\left(C^TX_k\right)\leq\mathrm{Tr}\left(C^T\hat{X}\right)=\mathrm{Tr}\left(C^TX_{k-1}\right)-\frac{\sqrt{\Phi}}{||Q||_F}\\&\leq\mathrm{Tr}\left(C^TX_{k-1}\right)-\frac{\sqrt{\Phi}}{||C||_F||X_{k-1}^{-1}||_F}\\&\leq\mathrm{Tr}\left(C^TX_{k-1}\right)-\frac{\sqrt{\Phi}}{t_{k-1}\sqrt{1+\Theta}}<\mathrm{Tr}\left(C^TX_{k-1}\right)\end{aligned}\tag{16}$$

Equation (16) provides conservative a bound on the cost decrease after a *decrease* step is taken.

Step 2: From the strictly decreasing property of the objective of (1) with respect to t, if follows that the values of t such that the corresponding solution to (2) are X_k and X_{k-1} satisfy $t_k>t_{k-1}$. However, in order to establish finite time convergence we need to prove that $t_k\geq\chi t_{k-1}$, with $\chi>1$. This requires, given X_k, finding the corresponding value t_k. Since this is a non-trivial problem, we will find a function $g(.)$ such that $g(t_k)\leq\mathrm{Tr}\left(C^TX_k\right)$ and use it as a proxy, to find some $\tilde{t_k}\leq t_k$. The desired results will be established by showing that $t_k\geq\tilde{t_k}\geq\chi t_{k-1}$.

Lemma 7: Given t_o, let X_{t_o} denote the solution to (2) corresponding to $t=t_o$. Then the function $g_0(t)=\mathrm{Tr}\left(C^TX_{t_o}\right)-N/t_0+N/t$ is a lower bound of $\mathrm{Tr}\left(C^TX_t\right)$ for any $t\geq t_0$.

Using the result above, we can now establish the main result of the paper:

Theorem 2: Algorithm 1 converges to ϵ_g-optimality in at most κ iterations of Decrease and Centering Phases, where κ is given by:

$$\kappa=\lceil\frac{\log\left(N/\epsilon^*\right)-\log\left(t_0\right)}{\log\left(\chi\right)}\rceil\qquad\chi=\frac{N\sqrt{1+\Theta}}{N\sqrt{1+\Theta}-\sqrt{\Phi}}>1\tag{17}$$

Proof: Let $\hat{\mathcal{C}}_k$, $\hat{t}_k$ denote the cost associated with the feasible solution $\hat{Y}_k$ and the corresponding value of the parameter. We can use Lemma 7 to find a value $\tilde{t}_k$ such that $g_{k-1}(\tilde{t}_k)=\hat{\mathcal{C}}_k$, which implies that $\hat{t}_k>\tilde{t}_k>t_{k-1}$. This $\tilde{t}_k$ is given by:

$$\begin{aligned}g_{k-1}(\tilde{t}_k)=\hat{\mathcal{C}}_k\implies&\mathrm{Tr}\left(C^TX_{k-1}\right)-\frac{N}{t_{k-1}}+\frac{N}{\tilde{t}_k}=\hat{\mathcal{C}}_k\\&\leq\mathrm{Tr}\left(C^TX_{k-1}\right)-\frac{\sqrt{\Phi}}{t_{k-1}\sqrt{1+\Theta}}\\&\implies\tilde{t}_k\geq t_{k-1}\frac{N\sqrt{1+\Theta}}{N\sqrt{1+\Theta}-\sqrt{\Phi}}\equiv t_{k-1}\,\chi\end{aligned}\tag{18}$$

where $\chi>1$ depends only on the problem data. Since $g_{k-1}(\tilde{t}_k)$ is a lower bound of $\mathrm{Tr}\left(C^TX_{\tilde{t}_k}\right)$, from monotonicity it follows that $\hat{t}_k\geq\tilde{t}_k\geq\chi t_{k-1}$ Next, note that X_k, the optimal solution to (4) satisfies $\mathrm{Tr}\left(C^TX_{k-1}\right)\leq\hat{\mathcal{C}}_k$ and hence its associated parameter t_k satisfies $t_k\geq\hat{t}_k$. Thus, it follows that at the end of the decrease phase, the value of t corresponding to the optimal solution X_k satisfies $t_k\geq\chi t_{k-1}$. Thus, the number of iterations needed to reach $t^*=N/\epsilon^*$ is given by $t^*\leq\chi^\kappa\,t_0$ with κ given by (17). ■

So far we have considered, for simplicity, the case where the *centering* phase returns exactly to the center path, that is, X_k exactly solves (6). Below, we briefly show that the convergence results hold, even if the *centering* phase provides an ϵ_c suboptimal solution X_L to (6), provided that $\epsilon_c<\bar{\epsilon}_c$, where the constant $\bar{\epsilon}_c$ depends only on the problem data. The intuition behind the proof, illustrated in Figure 2, is that if X_L and X_k are close enough, then the solution $\hat{X}$ can be shown to satisfy the norm constraint $||U_L^{-T}\,\hat{X}\,U_L^{-1}-I||_F^2\leq\Phi'$, where U_L are the Cholesky factors of X_L and Φ' is a constant such that $\Phi<\Phi'\leq\{2/(N+1),1\}$. Thus $\hat{X}$ used in the proof of step 1 above is within the feasible set of the decrease step (4) taken from X_L, rather than X_k (green circle in Fig 2). Proceeding along these lines, the following results can be established (see the ArXiv version of the paper for a proof)

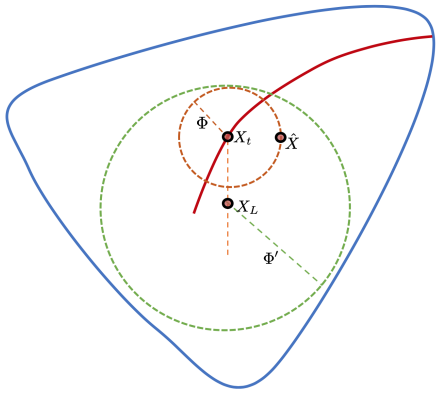

Fig. 2: Outline of the proof of convergence when the *centering phase* stops at a ϵ_C-optimal point X_L. Circles in orange and green signify volumes that lie within a decrease step of X_t and X_L, respectively. If X_t and X_L are close enough, measured by ϵ_C, then the orange volume is contained within the green one and $\hat{X}$ lies within a decrease step from X_L, extending the guarantees of Theorem 2 to the suboptimal $\epsilon>0$ case.

Theorem 3: Assume X_L is an ϵ_C-optimal solution to the *centering* phase. Assume further that $\Phi'=\{2/(N+1),1\}$,

for the DD and SDD case respectively, and that $\Phi < \Phi'$. Then if:

$$\epsilon_c^* = -\log\left(1 - \left(\frac{\sqrt{\Phi'} - \sqrt{\Phi}}{\sqrt{N} + \sqrt{\Phi'}}\right)^3\right) \tag{19}$$

the cost function after a decrease step taken from X_L is at least as low as the cost function evaluated at $\hat{X}$.

C. Numerical Tests on SDPLib problems and random SDPs

In this section we test the numerical performance of the proposed approach on some standard benchmark problems from the SDPLib dataset [14] and on randomly generated SDPs. From the SDPLib library, we use the theta number instances of the SDPLib *Theta-1* and *Theta-2* and MaxCut problems *mcp100*, *mcp125-1*, *mcp125-2*, *mcp250-1* and *mcp250-2*, which are also SDP relaxations of the well-known NP-hard problem of finding the maximum cut in a graph. In all cases we used the SDD cone and set the number of decrease and centering steps $s_d = 5$. The results of running the algorithm implemented in Matlab R2016a on a MacBook Pro system with a 3 GHz dual core processor and 16 GB of RAM are shown in Table I. As shown there, the algorithm solved all instances within $\epsilon_g = 10^{-3}$ of optimal using comparatively modest computational resources.

Problem	N	M	opt. c	time (s)
theta1	50	104	-23	81.69
theta2	100	498	-32.879	2464.62
mcp100	100	100	-226.157	755.53
mcp124-1	124	124	-141.990	1508.26
mcp124-2	124	124	-269.880	1525.94
mcp250-1	250	250	-317.264	21259.07
mcp250-2	250	250	-531.930	22074.42
RandomSDP-1	50	50	-43.023	120.5
RandomSDP-2	50	100	-30.277	129.44
RandomSDP-3	100	50	-0.648	441.86
RandomSDP-4	100	100	0.968	608.83
RandomSDP-5	250	250	4265.157	15104.15

TABLE I: Execution time on SDPLib test problems and randomly generated SDPs. In all case the algorithm converged to the optimal values c^* within a 10^{-3} tolerance.

V. Conclusions.

While semi-definite programs are one of the cornerstones of modern control and identification, their use has been limited to relatively small problems. This can be traced to the poor scaling properties of existing SDP solvers. Interior point methods scale at least as $\mathcal{O}(mn^3)$ and first order ADDM methods have a complexity of at least $\mathcal{O}(n^3)$ per iteration, typically requiring substantial more iterations than IP methods. This poor scaling has prevented the use of SDPs in other fields such as machine learning, typically involving large data sets. To mitigate this computational complexity [8], [9] proposed to solve relaxations where the diagonally-dominant (DD) and scaled-diagonally dominant (SDD) cones are used as proxies for the semi-definite cone. However, these methods cannot guarantee convergence to the optimal value of the original SDP. Indeed, numerical evidence show that these relaxations tend to converge to strictly suboptimal values for all medium to large size problems ($N >> 10$). To circumvent this difficulty, in this paper we presented an algorithm, based on the diagonally-dominant (DD) and scaled-diagonally dominant (SDD) SDP relaxations developed in [8], [9] to solve SDPs to global optimality. The proposed approach relies on sequentially decreasing the cost of the iterates and improving its conditioning until an ϵ_g-optimal iterate is reached. The method is guaranteed to converge to that ϵ_g-optimal solution after a polynomially bounded number of iterations, resulting to the best of our knowledge in the first globally convergent SDP algorithm based on DD and SDD conic problems. The proposed algorithm is also shown to converge in numerical experiments using SDP problems from the SDPLib dataset [14] and randomly generated SDPs. Besides the practical implications of this algorithm, our work closes an open question in the field of efficient algorithms for large-scale SDPs: whether or not it is possible to solve an arbitrary SDP in polynomial time by solving a sequence of lower complexity Linear Programs or Second Order Cone Programs.

References

[1] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*, volume 15 of *Studies in Applied Mathematics*. SIAM, Philadelphia, PA, June 1994.

[2] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM review*, 38(1):49–95, 1996.

[3] F. Alizadeh, J. P. Haeberly, and M. L. Overton. Primal-dual interior-point methods for semidefinite programming: Convergence rates, stability and numerical results. *SIAM J. on Optimization*, 8(3):746–768, August 1998.

[4] K. Fujisawa, M. Kojima, and K. Nakata. Exploiting sparsity in primal-dual interior-point methods for semidefinite programming. *Mathematical Programming*, 79(1-3):235–253, 1997.

[5] L. Vandenberghe and M. S. Andersen. Chordal graphs and semidefinite optimization. *Foundations and Trends in Optimization*, 1(4):241–433, 2015.

[6] N. Boumal. A riemannian low-rank method for optimization over semidefinite matrices with block-diagonal constraints. *arXiv preprint arXiv:1506.00575*, 2015.

[7] Y. Zheng, G. Fantuzzi, A. Papachristodoulou, P. Goulart, and A. Wynn. Fast ADMM for semidefinite programs with chordal sparsity. In *2017 American Control Conference (ACC)*, pages 3335–3340. IEEE, 2017.

[8] A. A. Ahmadi and G. Hall. Sum of squares basis pursuit with linear and second order cone programming. *Algebraic and Geometric Methods in Discrete Mathematics, Contemp. Math*, 685:27–53, 2015.

[9] A. A. Ahmadi and A. Majumdar. DSOS and SDSOS optimization: more tractable alternatives to sum of squares and semidefinite optimization. *SIAM J. on Applied Algebra and Geometry*, 3(2):193–230, 2019.

[10] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 302–311, 1984.

[11] Y. Nesterov and A. Nemirovskii. *Interior-point polynomial algorithms in convex programming*. SIAM, 1994.

[12] E. G. Boman, D. Chen, O. Parekh, and S. Toledo. On factor width and symmetric h-matrices. *Linear algebra and its applications*, 405:239–248, 2005.

[13] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[14] B. Borchers. Sdplib 1.2, a library of semidefinite programming test problems. *Optimization Methods and Software*, 11(1-4):683–690, 1999.