A Fast Distributed Algorithm for Decomposable Semidefinite Programs

Abdulrahman Kalbat and Javad Lavaei
Department of Electrical Engineering, Columbia University

***Abstract*—This paper aims to develop a fast, parallelizable algorithm for an arbitrary decomposable semidefinite program (SDP). To formulate a decomposable SDP, we consider a multi-agent canonical form represented by a graph, where each agent (node) is in charge of computing its corresponding positive semidefinite matrix subject to local equality and inequality constraints as well as overlapping (consistency) constraints with regards to the agent's neighbors. Based on the alternating direction method of multipliers, we design a numerical algorithm, which has a guaranteed convergence under very mild assumptions. Each iteration of this algorithm has a simple closed-form solution, consisting of matrix multiplications and eigenvalue decompositions performed by individual agents as well as information exchanges between neighboring agents. The cheap iterations of the proposed algorithm enable solving real-world large-scale conic optimization problems.**

I. Introduction

Alternating direction method of multipliers (ADMM) is a first-order optimization algorithm proposed in the mid-1970s by [1] and [2]. This method has attracted much attention recently since it can be used for large-scale optimization problems and also be implemented in parallel and distributed computational environments [3], [4]. Compared to second-order methods that are able to achieve a high accuracy via expensive iterations, ADMM relies on low-complex iterations and can achieve a modest accuracy in tens of iterations. Inspired by Nesterov's scheme for accelerating gradient methods [5], great effort has been devoted to accelerating ADMM and attaining a high accuracy in a reasonable number of iterations [6]. Since ADMM's performance is affected by the condition number of the problem's data, diagonal rescaling is proposed in [7] for a class of problems to improve the performance and achieve a linear rate of convergence. The $\mathcal{O}(\frac{1}{n})$ worst-case convergence rate of ADMM is proven in [8], [9] under the assumptions of closed convex sets and convex functions (not necessarily smooth). In [10], the $\mathcal{O}(\frac{1}{n})$ convergence rate is obtained for an asynchronous ADMM algorithm. The recent paper [11] represents ADMM as a dynamical system and then reduces the problem of proving the linear convergence of ADMM to verifying the stability of a dynamical system [11].

Semidefinite programs (SDP) are attractive due in part to three reasons. First, positive semidefinite constraints appear in many applications [12]. Second, SDPs can be used to study and approximate hard combinatorial optimization problems [13]. Third, this class of convex optimization problems includes linear, quadratic, quadratically-constrained quadratic, and second-order cone programs. It is known that small- to medium-sized SDP problems can be solved efficiently by interior point methods in polynomial time up to any arbitrary precision [14]. However, these methods are less practical for large-scale SDPs due to computation time and memory issues. However, it is possible to somewhat reduce the complexity by exploiting any possible structure in the problem such as sparsity.

The pressing need for solving real-world large-scale optimization problems calls for the development of efficient, scalable, and parallel algorithms. Because of the scalability of ADMM, the main objective of this work is to design a distributed ADMM-based parallel algorithm for solving an arbitrary sparse large-scale SDP with a guaranteed convergence, under very mild assumptions. We consider a canonical form of decomposable SDPs, which is characterized by a graph of agents (nodes) and edges. Each agent needs to find the optimal value of its associated positive semidefintie matrix subject to local equality and inequality constraints as well as overlapping constraints with its neighbors (more precisely, the matrices of two neighboring agents may be subject to consistency constraints). The objective function of the overall SDP is the summation of individual objectives of all agents. From the computation perspective, each agent is treated as a processing unit and each edge of the graph specifies what agents can communicate. We propose a distributed algorithm, whose iterations comprise local matrix multiplications and eigenvalue decompositions performed by individual agents as well as information exchanges between neighboring agents.

This paper is organized as follows. An overview of ADMM is provided in Section II. The distributed multi-agent SDP problem is formalized in Section III. An ADMM-based parallel algorithm is developed in Section IV, by first studying the 2-agent case and then investigating the general multi-agent case. Simulation results on randomly-generated large-scale SDPs with a few million variables are provided in Section V. Finally, some concluding remarks are drawn in Section VI.

Notations: $\mathbb{R}^n$ and $\mathbb{S}^n$ denote the sets of $n \times 1$ real vectors and $n \times n$ symmetric matrices, respectively. Lower case letters (e.g., x) represent vectors, and upper case letters (e.g., W) represent matrices. $\mathbf{tr}\{W\}$ denotes the trace of a matrix W and the notation $W \succeq 0$ means that W is symmetric and positive semidefinite. Given a matrix W, its (l, m) entry is denoted as $W(l, m)$. The symbols $(\cdot)^T$, $\|\cdot\|_2$ and $\|\cdot\|_F$ denote the transpose, ℓ_2-norm (for vectors) and Frobenius norm (for matrices) operators, respectively. The ordering operator $(a, b)_{\preceq}$ returns (a, b) if $a < b$ and returns (b, a) if $a > b$. The notation $|\mathcal{X}|$ represents the cardinality (or size) of the set $\mathcal{X}$. The finite

Email: ak3369@columbia.edu and lavaei@ee.columbia.edu.

This work was supported by the ONR YIP Award, NSF CAREER Award 1351279, and NSF EECS Award 1406865.

sequence of variables $x_1, \ldots, x_n$ is denoted by $\{x_i\}_{i=1}^{n}$. For an $m \times n$ matrix W, the notation $W(\mathcal{X}, \mathcal{Y})$ denotes the submatrix of W whose rows and columns are chosen form $\mathcal{X}$ and $\mathcal{Y}$, respectively, for given index sets $\mathcal{X} \subseteq \{1, \ldots, m\}$ and $\mathcal{Y} \subseteq \{1, \ldots, n\}$.

The notation $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ defines a graph $\mathcal{G}$ with the vertex (or node) set $\mathcal{V}$ and the edge set $\mathcal{E}$. The set of neighbors of vertex $i \in \mathcal{V}$ is denoted as $N(i)$. To orient the edges of $\mathcal{G}$, we define a new edge set $\mathcal{E}^+ = \{(i,j) \mid (i,j) \in \mathcal{E} \ \text{ and } \ i < j\}$.

II. Alternating Direction Method of Multipliers

Consider the optimization problem

$$\min_{x \in \mathbb{R}^n,\ y \in \mathbb{R}^m} \quad f(x) + g(y) \tag{1a}$$

$$\text{subject to} \quad Ax + By = c \tag{1b}$$

where $f(x)$ and $g(y)$ are convex functions, A, B are known matrices, and c is a given vector of appropriate dimension. The above optimization problem has a separable objective function and linear constraints. Before proceeding with the paper, three numerical methods for solving this problem will be reviewed.

The first method is *dual decomposition*, which uses the Lagrangian function

$$\begin{aligned} \mathcal{L}(x,y,\lambda) &= f(x) + g(y) + \lambda^T(Ax + By - c) \\ &= \underbrace{f(x) + \lambda^T Ax}_{h_1(x,\lambda)} + \underbrace{g(y) + \lambda^T By}_{h_2(y,\lambda)} - \lambda^T c \end{aligned} \tag{2}$$

where λ is the Lagrange multiplier corresponding to the constraint (1b). The above Lagrangian function can be separated into two functions $h_1(x,\lambda)$ and $h_2(y,\lambda)$. Inspired by this separation, the dual decomposition method is based on updating x, y and λ separately. This leads to the iterations

$$x^{t+1} := \underset{x}{\text{argmin}}\ h_1(x, \lambda^t) \tag{3a}$$

$$y^{t+1} := \underset{y}{\text{argmin}}\ h_2(y, \lambda^t) \tag{3b}$$

$$\lambda^{t+1} := \lambda^t + \alpha^t(Ax^{t+1} + By^{t+1} - c) \tag{3c}$$

for $t = 0, 1, 2, ...$, with an arbitrary initialization (x^0, y^0, λ^0), where α^t is a step size. Note that "argmin" denotes any minimizer of the corresponding function.

Despite its decomposability, the dual decomposition method has robustness and convergence issues. The *method of multipliers* could be used to remedy these difficulties, which is based on the augmented lagrangian function

$$\begin{aligned} \mathcal{L}_\mu(x,y,\lambda) = f(x) + g(y) + \lambda^T(Ax + By - c) \\ + \frac{\mu}{2}\|Ax + By - c\|_2^2 \end{aligned} \tag{4}$$

where μ is a nonnegative constant. Notice that (4) is obtained by augmenting the Lagrangian function in (2) with a quadratic term in order to increase the smallest eigenvalue of the Hessian of the Lagrangian with respect to (x, y). However, this augmentation creates a coupling between x and y. The iterations corresponding to the method of multipliers are

$$(x^{t+1}, y^{t+1}) := \underset{(x,y)}{\text{argmin}}\ \mathcal{L}_\mu(x, y, \lambda^t) \tag{5a}$$

$$\lambda^{t+1} := \lambda^t + \mu(Ax^{t+1} + By^{t+1} - c) \tag{5b}$$

where $t = 0, 1, 2, ...$

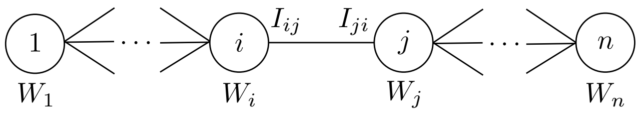

Fig. 1: A graph representation of the distributed multi-agent SDP.

In order to avoid solving a joint optimization with respect to x and y at every iteration, the *alternating direction method of multipliers* (ADMM) can be used. The main idea is to first update x by freezing y at its latest value, and then update y based on the most recent value of x. This leads to the 2-block ADMM problem with the iterations [4]:

$$\text{Block 1:} \quad x^{t+1} := \underset{x}{\text{argmin}}\ \mathcal{L}_\mu(x, y^t, \lambda^t) \tag{6a}$$

$$\text{Block 2:} \quad y^{t+1} := \underset{y}{\text{argmin}}\ \mathcal{L}_\mu(x^{t+1}, y, \lambda^t) \tag{6b}$$

$$\text{Dual:} \quad \lambda^{t+1} := \lambda^t + \mu(Ax^{t+1} + By^{t+1} - c) \tag{6c}$$

ADMM offers a distributed computation property, a high degree of robustness, and a guaranteed convergence under very mild assumptions. In the reminder of this paper, we will use this first-order method to solve large-scale decomposable SDP problems.

III. Problem Formulation

Consider a simple, connected, and undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with the node set $\mathcal{V} := \{1, \ldots, n\}$ and the edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, as shown in Figure 1. In a physical context, each node could represent an agent (or a machine or a processor or a thread) and each edge represents a communication link between the agents. In the context of this paper, each agent is in charge of computing a positive semidefinite matrix variable W_i, and each edge $(i,j) \in \mathcal{E}$ specifies an overlap between the matrix variables W_i and W_j of agents i and j. More precisely, each edge (i,j) is accompanied by two arbitrary integer-valued index sets I_{ij} and I_{ji} to capture the overlap between W_i and W_j through the equation $W_i(I_{ij}, I_{ij}) = W_j(I_{ji}, I_{ji})$. Figure 2 illustrates this specification through an example with three overlapping matrices, where every two neighboring submatrices with an identical color must take the same value at optimality. Another way of thinking about this setting is that Figure 1 represents the sparsity graph of an arbitrary sparse large-scale SDP with a single global matrix variable W, which is then reformulated in terms of certain matrices of W, named $W_1, ..., W_n$, using the Chordal extension and matrix completion theorems [15]. The objective of this paper is to solve the decomposable SDP problem (interchangeably referred to as distributed multi-agent SDP) given below.

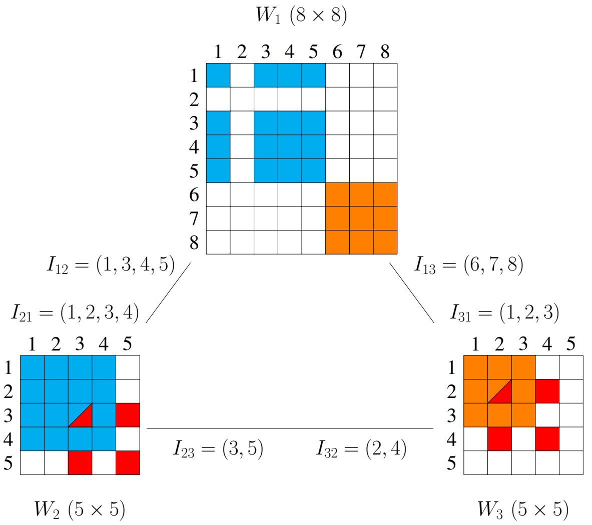

Fig. 2: An illustration of the definitions of I_{ij} and I_{ji} for three overlapping submatrices W_1, W_2 and W_3

Decomposable SDP:

$$\min \sum_{i \in \mathcal{V}} \mathbf{tr}(A_i W_i) \tag{7a}$$

$$\text{subject to :}$$

$$\mathbf{tr}(B_j^{(i)} W_i) = c_j^{(i)} \qquad \forall\ j = 1, \ldots, p_i \quad \text{and} \quad i \in \mathcal{V} \tag{7b}$$

$$\mathbf{tr}(D_l^{(i)} W_i) \leq d_l^{(i)} \qquad \forall\ l = 1, \ldots, q_i \quad \text{and} \quad i \in \mathcal{V} \tag{7c}$$

$$W_i \succeq 0 \qquad \forall\ i \in \mathcal{V} \tag{7d}$$

$$W_i(I_{ij}, I_{ij}) = W_j(I_{ji}, I_{ji}) \qquad \forall\ (i,j) \in \mathcal{E}^+ \tag{7e}$$

with the variables $W_i \in \mathbb{S}^{n_i}$ for $i = 1, ..., n$, where

- the superscript in $(\cdot)^{(i)}$ is not a power but means that the expression corresponds to agent $i \in \mathcal{V}$.
- n_i denotes the size of the submatrix W_i, and p_i and q_i show the numbers of equality and inequality constraints for agent i, respectively.
- $c_j^{(i)}$ and $d_l^{(i)}$ denote the j^{th} and l^{th} elements of the vectors $c_i \in \mathbb{R}^{p_i}$ and $d_i \in \mathbb{R}^{q_i}$ for agent i, as defined below:
$$c_i \triangleq [c_1^{(i)}, \ldots, c_{p_i}^{(i)}]^T, \qquad d_i \triangleq [d_1^{(i)}, \ldots, d_{q_i}^{(i)}]^T$$
- the matrices A_i, $B_j^{(i)}$, and $D_l^{(i)}$ are known and correspond to agent $i \in \mathcal{V}$.

The formulation in (7) has three main ingredients:

- **Local objective function:** each agent $i \in \mathcal{V}$ has its own local objective function $\mathbf{tr}(A_i W_i)$ with respect to the local matrix variable W_i. The summation of all local objective functions denotes the global objective function in (7a).
- **Local constraints:** each agent $i \in \mathcal{V}$ has local equality and inequality constraints (7b) and (7c), respectively, as well as a local positive semidefiniteness constraint (7d).
- **Overlapping constraints:** constraint (7e) states that certain entries of W_i and W_j are identical.

The objective is to design a distributed algorithm for solving (7), by allowing each agent $i \in \mathcal{V}$ to collaborate with its neighbors $N(i)$ to find an optimal value for its positive semidefinite submatrix W_i while meeting its own constraints as well as all overlapping constraints. This is accomplished by local computations performed by individual agents and local communication between neighboring agents for information exchange.

There are two scenarios in which (7) could be used. In the first scenario, it is assumed that the SDP problem of interest is associated with a multi-agent system and matches the formulation in (7) exactly. In the second scenario, we consider an arbitrary sparse SDP problem in the centralized standard form, i.e., an SDP with a single positive semidefinite matrix W, and then convert it into a distributed SDP with multiple but smaller positive semidefinite matrices W_i to match the formulation in (7) (note that a dense SDP problem can be put in the form of (7) with $n = 1$). The conversion from a standard SDP to a distributed SDP is possible using the idea of chordal decomposition of positive semidefinite cones in [16], which exploits the fact that a matrix W has a positive semidefinite completion if and only if certain submatrices of W, denoted as $W_1, ..., W_n$, are positive semidefinite [17].

In this work, we propose an iterative algorithm for solving the decomposable SDP problem (7) using the first-order ADMM method. We show that each iteration of this algorithm has a simple closed-form solution, which consists of matrix multiplication and eigenvalue decomposition over matrices of size n_i for agent $i \in \mathcal{V}$.

Our work improves upon some recent papers in this area. [18] is a special case of our work with $n = 1$, which does not offer any parallelizable algorithm for sparse SDPs and may not be applicable to large-scale sparse SDP problems. [16] uses the clique-tree conversion method to decompose sparse SDPs with chordal sparsity pattern into smaller sized SDPs, which can then be solved by interior point methods but this approach is limited by the large number of consistency constraints for the overlapping parts. Recently, [19] solves the decomposed SDP created by [16] using a first-order splitting method, but it requires solving a quadratic program at every iteration, which again imposes some limitations on the scalability of the proposed algorithm. In contrast, the algorithm to be proposed here is parallelizable with low computations at every iteration, without requiring any initial feasible point unlike interior point methods.

IV. Distributed Algorithm for Decomposable Semidefinite Programs

In this section, we design an ADMM-based algorithm to solve (7). For the convenience of the reader, we first consider the case where there are only two overlapping matrices W_1 and W_2. Later on, we derive the iterations for the general case with an arbitrary graph $\mathcal{G}$.

A. Two-Agent Case

Assume that there are two overlapping matrices W_1 and W_2 embedded in a global SDP matrix variable W as shown in Figure 3, where "*" submatrices of W are redundant (meaning that there is no explicit constraint on the entries of these parts). The SDP problem for this case can be put in the canonical form (7), by setting $\mathcal{V} = \{1, 2\}$, $\mathcal{E}^+ = \{(1, 2)\}$ and $|\mathcal{V}| = 2$:

$$\min_{\substack{W_1 \in \mathbb{S}^{n_1} \\ W_2 \in \mathbb{S}^{n_2}}} \quad \mathbf{tr}(A_1 W_1) + \mathbf{tr}(A_2 W_2) \tag{8a}$$

$$\text{s.t.} \quad \mathbf{tr}(B_j^{(1)} W_1) = c_j^{(1)} \qquad \forall\, j = 1, \ldots, p_1 \tag{8b}$$

$$\mathbf{tr}(B_j^{(2)} W_2) = c_j^{(2)} \qquad \forall\, j = 1, \ldots, p_2 \tag{8c}$$

$$\mathbf{tr}(D_l^{(1)} W_1) \leq d_l^{(1)} \qquad \forall\, l = 1, \ldots, q_1 \tag{8d}$$

$$\mathbf{tr}(D_l^{(2)} W_2) \leq d_l^{(2)} \qquad \forall\, l = 1, \ldots, q_2 \tag{8e}$$

$$W_1, W_2 \succeq 0 \tag{8f}$$

$$W_1(I_{12}, I_{12}) = W_2(I_{21}, I_{21}) \tag{8g}$$

where the data matrices A_1, $B_j^{(1)}$,$D_l^{(1)} \in \mathbb{S}^{n_1}$, the matrix variable $W_1 \in \mathbb{S}^{n_1}$ and the vectors $c_1 \in \mathbb{R}^{p_1}$ and $d_1 \in \mathbb{R}^{q_1}$ correspond to agent 1, whereas the data matrices A_2, $B_j^{(2)}$,$D_l^{(2)} \in \mathbb{S}^{n_2}$, the matrix variable $W_2 \in \mathbb{S}^{n_2}$ and the vectors $c_2 \in \mathbb{R}^{p_2}$ and $d_2 \in \mathbb{R}^{q_2}$ correspond to agent 2. Constraint (8g) states that the (I_{12}, I_{12}) submatrix of W_1 overlaps with the (I_{21}, I_{21}) submatrix of W_2. With no loss of generality, assume that the overlapping part occurs at the lower right corner of W_1 and the upper left corner of W_2, as illustrated in Figure 3. The dual of the 2-agent SDP problem in (8) can be expressed as

$$\min \left(c_1^T z_1 + d_1^T v_1\right) + \left(c_2^T z_2 + d_2^T v_2\right) \tag{9a}$$

$$\text{subject to :}$$

$$-\sum_{j=1}^{p_1} z_j^{(1)} B_j^{(1)} - \sum_{l=1}^{q_1} v_l^{(1)} D_l^{(1)} + R_1 - \begin{bmatrix} 0 & 0 \\ 0 & H_{1,2} \end{bmatrix} = A_1 \tag{9b}$$

$$-\sum_{j=1}^{p_2} z_j^{(2)} B_j^{(2)} - \sum_{l=1}^{q_2} v_l^{(2)} D_l^{(2)} + R_2 + \begin{bmatrix} H_{2,1} & 0 \\ 0 & 0 \end{bmatrix} = A_2 \tag{9c}$$

$$H_{1,2} = H_{2,1} \tag{9d}$$

$$v_1, v_2 \geq 0 \tag{9e}$$

$$R_1, R_2 \succeq 0 \tag{9f}$$

with the variables $z_1, z_2, v_1, v_2, R_1, R_2, H_{1,2}, H_{2,1}$, where $z_1 \in \mathbb{R}^{p_1}$, $z_2 \in \mathbb{R}^{p_2}$, $v_1 \in \mathbb{R}^{q_1}$ and $v_2 \in \mathbb{R}^{q_2}$ are the Lagrange multipliers corresponding to the equality and inequality constraints in (8b)-(8e), respectively, and the dual matrix variables $R_1 \in \mathbb{S}^{n_1}$ and $R_2 \in \mathbb{S}^{n_2}$ are the Lagrange multiplier corresponding to the constraint (8f). The dual matrix variable $H_{1,2}$ is the Lagrange multiplier corresponding to the submatrix $W_1(I_{12}, I_{12})$ of W_1, whereas $H_{2,1}$ is the Lagrange multiplier corresponding to the submatrix $W_2(I_{21}, I_{21})$ of W_2. Since the overlapping entries between W_1 and W_2 are equal, as reflected in constraint (8g), the corresponding Lagrange multipliers should be equal as well, leading to constraint (9d).

If we apply ADMM to (9), it becomes impossible to split the variables into two blocks of variables associated with agents 1 and 2. The reason is that the augmented Lagrangian function of (9) creates a coupling between $H_{1,2}$ and $H_{2,1}$, which then requires updating $H_{1,2}$ and $H_{2,1}$ jointly. This issue can be resolved by introducing a new auxiliary variable $H^{(1,2)}$ in order to decompose the constraint $H_{1,2} = H_{2,1}$ into two constraints $H_{1,2} = H^{(1,2)}$ and $H_{2,1} = H^{(1,2)}$. Similarly, to make the update of v_1 and v_2 easier, we do not impose positivity constraints directly on v_1 and v_2 as in (9e). Instead, we impose the positivity on two new vectors $u_1, u_2 \geq 0$ and then add the additional constraints $v_1 = u_1$ and $v_2 = u_2$. By applying the previous modifications, (9) could be rewritten in the decomposable form

$$\min \quad \sum_{i=1}^{2} \left(c_i^T z_i + d_i^T v_i + I_+(R_i) + I_+(v_i)\right) \tag{10a}$$

$$\text{subject to :}$$

$$-\sum_{j=1}^{p_1} z_j^{(1)} B_j^{(1)} - \sum_{l=1}^{q_1} v_l^{(1)} D_l^{(1)} + R_1 - \begin{bmatrix} 0 & 0 \\ 0 & H_{1,2} \end{bmatrix} = A_1 \tag{10b}$$

$$-\sum_{j=1}^{p_2} z_j^{(2)} B_j^{(2)} - \sum_{l=1}^{q_2} v_l^{(2)} D_l^{(2)} + R_2 + \begin{bmatrix} H_{2,1} & 0 \\ 0 & 0 \end{bmatrix} = A_2 \tag{10c}$$

$$H_{1,2} = H^{(1,2)} \tag{10d}$$

$$H_{2,1} = H^{(1,2)} \tag{10e}$$

$$v_1 = u_1 \tag{10f}$$

$$v_2 = u_2 \tag{10g}$$

with the variables $z_1, z_2, v_1, u_1, v_2, u_2, R_1, R_2, H_{1,2}, H_{2,1}$, $H^{(1,2)}$, where $I_+(R_i)$ is equal to 0 if $R_i \succeq 0$ and is $+\infty$ otherwise, and $I_+(v_i)$ is equal to 0 if $v_i \geq 0$ and is $+\infty$ otherwise. To streamline the presentation, define

$$B_i^{\text{sum}} = \sum_{j=1}^{p_i} z_j^{(i)} B_j^{(i)}, \quad D_i^{\text{sum}} = \sum_{l=1}^{q_i} v_l^{(i)} D_l^{(i)}, \quad i = 1, 2 \tag{11}$$

and

$$H_{1,2}^{\text{full}} = \begin{bmatrix} 0 & 0 \\ 0 & H_{1,2} \end{bmatrix}, \quad H_{2,1}^{\text{full}} = \begin{bmatrix} -H_{2,1} & 0 \\ 0 & 0 \end{bmatrix} \tag{12}$$

Note that B_i^{sum}, D_i^{sum}, $H_{1,2}^{\text{full}}$ and $H_{2,1}^{\text{full}}$ are functions of the variables z_i, v_i, $H_{1,2}$ and $H_{2,1}$, respectively, but the arguments are dropped for notational simplicity. The augmented Lagrangian function for (10) can be obtained as

$$\begin{aligned}
&\mathcal{L}_\mu\left(\mathcal{F}, \mathcal{M}\right) = \\
&\sum_{i=1}^{2} \left(c_i^T z_i + d_i^T v_i + I_+(R_i) + I_+(v_i)\right) \\
&+ \frac{\mu}{2} \left\| -B_1^{\text{sum}} - D_1^{\text{sum}} + R_1 - H_{1,2}^{\text{full}} - A_1 + \frac{G_1}{\mu} \right\|_F^2 \\
&+ \frac{\mu}{2} \left\| -B_2^{\text{sum}} - D_2^{\text{sum}} + R_2 - H_{2,1}^{\text{full}} - A_2 + \frac{G_2}{\mu} \right\|_F^2 \\
&+ \frac{\mu}{2} \left\| H_{1,2} - H^{(1,2)} + \frac{G_{1,2}}{\mu} \right\|_F^2 \\
&+ \frac{\mu}{2} \left\| H_{2,1} - H^{(1,2)} + \frac{G_{2,1}}{\mu} \right\|_F^2 \\
&+ \frac{\mu}{2} \left\| v_1 - u_1 + \frac{\lambda_1}{\mu} \right\|_2^2 + \frac{\mu}{2} \left\| v_2 - u_2 + \frac{\lambda_2}{\mu} \right\|_2^2
\end{aligned} \tag{13}$$

where $\mathcal{F} = \left(z_1, z_2, v_1, v_2, u_1, u_2, R_1, R_2, H_{1,2}, H_{2,1}, H^{(1,2)}\right)$ is the set of optimization variables and $\mathcal{M} =$

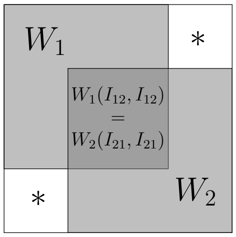

Fig. 3: Positive semidefinite matrix W (two blocks)

$(G_1, G_2, G_{1,2}, G_{2,1}, \lambda_1, \lambda_2)$ is the set of Lagrange multipliers whose elements correspond to constraints (10b) - (10g), respectively. Note that the augmented Lagrangian in (13) is obtained using the identity

$$\begin{aligned}\mathbf{tr}\left[X^T(A-B)\right] + \frac{\mu}{2}\left\|A-B\right\|_F^2 = \\ = \frac{\mu}{2}\left\|A - B + \frac{X}{\mu}\right\|_F^2 + \text{constant}\end{aligned} \tag{14}$$

In order to proceed, we need to split the set of optimization variables $\mathcal{F}$ into two blocks of variables. To this end, define $\mathcal{X} = \left\{u_1, u_2, R_1, R_2, H^{(1,2)}\right\}$ and $\mathcal{Y} = \{z_1, z_2, v_1, v_2, H_{1,2}, H_{2,1}\}$. Using the method delineated in Section II, the two-block ADMM iterations can be obtained as

$$\text{(Block 1)}\quad \mathcal{X}^{t+1} = \underset{\mathcal{X}}{\text{argmin}}\ \mathcal{L}_\mu\left(\mathcal{X}, \mathcal{Y}^t, \mathcal{M}^t\right) \tag{15a}$$

$$\text{(Block 2)}\quad \mathcal{Y}^{t+1} = \underset{\mathcal{Y}}{\text{argmin}}\ \mathcal{L}_\mu\left(\mathcal{X}^{t+1}, \mathcal{Y}, \mathcal{M}^t\right) \tag{15b}$$

$$G_1^{t+1} = G_1^t + \mu\left(-B_1^{\overset{t+1}{\text{sum}}} - D_1^{\overset{t+1}{\text{sum}}} + R_1^{t+1} - H_{1,2}^{\overset{t+1}{\text{full}}} - A_1\right) \tag{15c}$$

$$G_2^{t+1} = G_2^t + \mu\left(-B_2^{\overset{t+1}{\text{sum}}} - D_2^{\overset{t+1}{\text{sum}}} + R_2^{t+1} - H_{2,1}^{\overset{t+1}{\text{full}}} - A_2\right) \tag{15d}$$

$$G_{1,2}^{t+1} = G_{1,2}^t + \mu\left(H_{1,2}^{t+1} - H^{\overset{t+1}{(1,2)}}\right) \tag{15e}$$

$$G_{2,1}^{t+1} = G_{2,1}^t + \mu\left(H_{2,1}^{t+1} - H^{\overset{t+1}{(1,2)}}\right) \tag{15f}$$

$$\lambda_1^{t+1} = \lambda_1^t + \mu\left(v_1^{t+1} - u_1^{t+1}\right) \tag{15g}$$

$$\lambda_2^{t+1} = \lambda_2^t + \mu\left(v_2^{t+1} - u_2^{t+1}\right) \tag{15h}$$

for $t = 0, 1, 2, ...$ The above updates are derived based on the fact that ADMM aims to find a saddle point of the augmented lagrangian function by alternatively performing one pass of Gauss Seidel over $\mathcal{X}$ and $\mathcal{Y}$ and then updating the Lagrange multipliers $\mathcal{M}$ through Gradient ascent.

It is straightforward to show that the optimization over $\mathcal{X}$ in Block 1 is fully decomposable and amounts to 5 separate optimization subproblems with respect to the individual variables $u_1, u_2, R_1, R_2, H^{(1,2)}$. In addition, the optimization over $\mathcal{Y}$ in Block 2 is equivalent to 2 separate optimization subproblems with the variables $(z_1, v_1, H_{1,2})$ and $(z_2, v_2, H_{2,1})$, respectively. Interestingly, all these subproblems have closed-form solutions. The corresponding iterations that need to be taken by agents 1 and 2 are provided in (16) and (17) (given in the next page). Note that these agents need to perform local computation in every iteration according to (16) and (17) and then exchange the updated values of the pairs $(H_{1,2}, G_{1,2})$ and $(H_{2,1}, G_{2,1})$ with one another.

To elaborate on (16) and (17), the positive semidefinite matrices R_1 and R_2 are updated through the operator $(\cdot)_+$, where X_+ is defined as the projection of an arbitrary symmetric matrix X onto the set of positive semidefinite matrices by replacing its negative eigenvalues with 0 in the eigenvalue decomposition [18]. The positive vectors u_1 and u_2 are also updated through the operator $(x)_+$, which replaces any negative entry in an arbitrary vector x with 0 while keeping the nonnegative entries. Using the first-order optimality condition $\nabla_{H^{(1,2)}}\mathcal{L}_\mu(\cdot) = 0$, one could easily find the closed-form solution for $H^{(1,2)}$ as shown in (16c) and (17c). By combining the conditions $\nabla_{z_1}\mathcal{L}_\mu(\cdot) = 0$, $\nabla_{v_1}\mathcal{L}_\mu(\cdot) = 0$ and $\nabla_{H_{1,2}}\mathcal{L}_\mu(\cdot) = 0$, the updates of $(z_1, v_1, H_{1,2})$ and $(z_2, v_2, H_{2,1})$ reduce to a (not necessarily unique) linear mapping, denoted as Lin$(\cdot)$ in (16d) and (17d) (due to non-uniqueness, we may have multiple solutions, and any of them can be used in the updates). The Lagrange multipliers in $\mathcal{M}$ are updated through Gradient ascent, as specified in (16e)-(16g) for agent 1 and in (17e)-(17g) for agent 2.

B. Multi-Agent Case

In this part, we will study the general distributed multi-agent SDP (7). The dual of this problem, after considering all modifications used to convert (9) to (10), can be expressed in the decomposable form

$$\min \quad \sum_{i\in\mathcal{V}}\left(c_i^T z_i + d_i^T v_i + I_+(R_i) + I_+(v_i)\right) \tag{18a}$$

$$\text{subject to :}$$

$$-B_i^{\text{sum}} - D_i^{\text{sum}} + R_i - \sum_{k\in N(i)} H_{i,k}^{\text{full}} = A_i \quad \forall\ i \in \mathcal{V} \tag{18b}$$

$$H_{i,j} = H^{(i,j)} \qquad \forall\ (i,j) \in \mathcal{E}^+ \tag{18c}$$

$$H_{j,i} = H^{(i,j)} \qquad \forall\ (i,j) \in \mathcal{E}^+ \tag{18d}$$

$$v_i = u_i \qquad \forall\ i \in \mathcal{V} \tag{18e}$$

with the variables $z_i, v_i, u_i, R_i, H_{i,j}, H_{j,i}, H^{(i,j)}$ for every $i \in \mathcal{V}$ and $(i,j) \in \mathcal{E}^+$, where $B_i^{\text{sum}} = \sum_{j=1}^{p_i} z_j^{(i)} B_j^{(i)}$, $D_i^{\text{sum}} = \sum_{l=1}^{q_i} v_l^{(i)} D_l^{(i)}$ and $H_i^{\text{sum}} = \sum_{k\in N(i)} H_{i,k}^{\text{full}}$. Note that $z_i \in \mathbb{R}^{p_i}$ and $v_i \in \mathbb{R}^{q_i}$ are the Lagrange multipliers corresponding to the equality and inequality constraints in (7b) and (7c), respectively, and that $R_i \in \mathbb{S}^{n_i}$ is the Lagrange multiplier corresponding to the constraint (7d). Each element $h_{i,k}^{\text{full}}(a,b)$ of $H_{i,k}^{\text{full}}$ is either zero or equal to the Lagrange multiplier corresponding to an overlapping element $W_i(a,b)$ between W_i and W_k. For a better understanding of the difference between $H_{i,j}^{\text{full}}$, $H_{i,j}$ and H_i^{sum}, an example is given in Figure 4 for the case where agent 1 is overlapping with agents 2 and 3. The ADMM iterations for the general case can be derived similarly to the 2-agent case, which yields the local computation (19) for each agent $i \in \mathcal{V}$. Consider the parameters defined in (20) for every $i \in \mathcal{V}$, $(i,j) \in \mathcal{E}^+$, and time $t \in \{1, 2, 3, ...\}$. Define V^t as

Iterations for Agent 1

$$R_1^{t+1} = \left(B_1^{\overset{t}{\text{sum}}} + D_1^{\overset{t}{\text{sum}}} + H_{1,2}^{\overset{t}{\text{full}}} + A_1 - \frac{G_1^t}{\mu} \right)_+ \tag{16a}$$

$$u_1^{t+1} = \left(v_1^t + \frac{\lambda_1^t}{\mu} \right)_+ \tag{16b}$$

$$\overset{t+1}{H^{(1,2)}} = \frac{1}{2} \left(H_{1,2}^t + H_{2,1}^t + \frac{G_{1,2}^t}{\mu} + \frac{G_{2,1}^t}{\mu} \right) \tag{16c}$$

$$(z_1, v_1, H_{1,2})^{t+1} = \text{Lin}\left(u_1^{t+1}, R_1^{t+1}, \overset{t+1}{H^{(1,2)}}, G_1^t, G_{1,2}^t, \lambda_1^t \right) \tag{16d}$$

$$G_1^{t+1} = G_1^t + \mu \left(-B_1^{\overset{t+1}{\text{sum}}} - D_1^{\overset{t+1}{\text{sum}}} + R_1^{t+1} - H_{1,2}^{\overset{t+1}{\text{full}}} - A_1 \right) \tag{16e}$$

$$G_{1,2}^{t+1} = G_{1,2}^t + \mu \left(H_{1,2}^{t+1} - \overset{t+1}{H^{(1,2)}} \right) \tag{16f}$$

$$\lambda_1^{t+1} = \lambda_1^t + \mu \left(v_1^{t+1} - u_1^{t+1} \right) \tag{16g}$$

Iterations for Agent 2

$$R_2^{t+1} = \left(B_2^{\overset{t}{\text{sum}}} + D_2^{\overset{t}{\text{sum}}} + H_{2,1}^{\overset{t}{\text{full}}} + A_2 - \frac{G_2^t}{\mu} \right)_+ \tag{17a}$$

$$u_2^{t+1} = \left(v_2^t + \frac{\lambda_2^t}{\mu} \right)_+ \tag{17b}$$

$$\overset{t+1}{H^{(1,2)}} = \frac{1}{2} \left(H_{1,2}^t + H_{2,1}^t + \frac{G_{1,2}^t}{\mu} + \frac{G_{2,1}^t}{\mu} \right) \tag{17c}$$

$$(z_2, v_2, H_{2,1})^{t+1} = \text{Lin}\left(u_2^{t+1}, R_2^{t+1}, \overset{t+1}{H^{(1,2)}}, G_2^t, G_{2,1}^t, \lambda_2^t \right) \tag{17d}$$

$$G_2^{t+1} = G_2^t + \mu \left(-B_2^{\overset{t+1}{\text{sum}}} - D_2^{\overset{t+1}{\text{sum}}} + R_2^{t+1} - H_{2,1}^{\overset{t+1}{\text{full}}} - A_2 \right) \tag{17e}$$

$$G_{2,1}^{t+1} = G_{2,1}^t + \mu \left(H_{2,1}^{t+1} - \overset{t+1}{H^{(1,2)}} \right) \tag{17f}$$

$$\lambda_2^{t+1} = \lambda_2^t + \mu \left(v_2^{t+1} - u_2^{t+1} \right) \tag{17g}$$

Iterations for Agent $i \in \mathcal{V}$

$$R_i^{t+1} = \left(B_i^{\overset{t}{\text{sum}}} + D_i^{\overset{t}{\text{sum}}} + H_i^{\overset{t}{\text{sum}}} + A_i - \frac{G_i^t}{\mu} \right)_+ \tag{19a}$$

$$u_i^{t+1} = \left(v_i^t + \frac{\lambda_i^t}{\mu} \right)_+ \tag{19b}$$

$$\overset{t+1}{H^{(i,k)\preceq}} = \frac{1}{2} \left(H_{i,k}^t + H_{k,i}^t + \frac{G_{i,k}^t}{\mu} + \frac{G_{k,i}^t}{\mu} \right) \quad \forall k \in N(i) \tag{19c}$$

$$\begin{aligned} \left(z_i^{t+1}, v_i^{t+1}, \left\{ H_{i,k}^{t+1} \right\}_{k \in N(i)} \right) = \text{Lin}\Big(& u_i^{t+1}, R_i^{t+1}, \\ & \left\{ \overset{t+1}{H^{(i,k)\preceq}} \right\}_{k \in N(i)}, G_i^t, \left\{ G_{i,k}^t \right\}_{k \in N(i)}, \lambda_i^t \Big) \end{aligned} \tag{19d}$$

$$G_i^{t+1} = G_i^t + \mu \left(-B_i^{\overset{t+1}{\text{sum}}} - D_i^{\overset{t+1}{\text{sum}}} + R_i^{t+1} - H_i^{\overset{t+1}{\text{sum}}} - A_i \right) \tag{19e}$$

$$G_{i,k}^{t+1} = G_{i,k}^t + \mu \left(H_{i,k}^{t+1} - H^{\overset{t+1}{(i,k)\preceq}} \right) \quad \forall k \in N(i) \tag{19f}$$

$$\lambda_i^{t+1} = \lambda_i^t + \mu \left(v_i^{t+1} - u_i^{t+1} \right) \tag{19g}$$

$$\left(\Delta_{p1}^t \right)_i = \left\| B_i^{\overset{t}{\text{sum}}} + D_i^{\overset{t}{\text{sum}}} + H_i^{\overset{t}{\text{sum}}} + A_i - R_i^t \right\|_F^2 \tag{20a}$$

$$\left(\Delta_{p2}^t \right)_{i,j} = \left\| H_{i,j}^t - \overset{t}{H^{(i,j)}} \right\|_F^2 \tag{20b}$$

$$\left(\Delta_{p3}^t \right)_{i,j} = \left\| H_{j,i}^t - \overset{t}{H^{(i,j)}} \right\|_F^2 \tag{20c}$$

$$\left(\Delta_{p4}^t \right)_i = \| v_i^t - u_i^t \|_2^2 \tag{20d}$$

$$\left(\Delta_{d1}^t \right)_i = \left\| R_i^t - R_i^{t-1} \right\|_F^2 \tag{20e}$$

$$\left(\Delta_{d2}^t \right)_i = \left\| u_i^t - u_i^{t-1} \right\|_2^2 \tag{20f}$$

$$\left(\Delta_{d3}^t \right)_{i,j} = 2 \left\| \overset{t}{H^{(i,j)}} - \overset{t-1}{H^{(i,j)}} \right\|_F^2 \tag{20g}$$

$$\begin{aligned} V^t = & \sum_{i \in \mathcal{V}} \left(\left(\Delta_{p1}^t \right)_i + \left(\Delta_{p4}^t \right)_i + \left(\Delta_{d1}^t \right)_i + \left(\Delta_{d2}^t \right)_i \right) \\ & + \sum_{i,j \in \mathcal{E}^+} \left(\left(\Delta_{p2}^t \right)_{i,j} + \left(\Delta_{p3}^t \right)_{i,j} + \left(\Delta_{d3}^t \right)_{i,j} \right) \end{aligned} \tag{21}$$

Note that $(\Delta_{p1}, \Delta_{p2}, \Delta_{p3}, \Delta_{p4})$, $(\Delta_{d1}, \Delta_{d1}, \Delta_{d1})$, and V are the primal residues, dual residues and aggregate residue for the decomposed problem (18). It should be noticed that the dual residues are only considered for the variables in the block $\mathcal{X} = \{u_i, R_i, H^{(i,j)}\}$. Since $H^{(i,j)}$ appears twice in (18), the norm in the residue Δ_{d3} is multiplied by 2. The main result of this paper will be stated below.

Theorem 1. *Assume that Slater's conditions hold for the decomposable SDP problem (7). Consider the iterative algorithm given in* (19). *The following statements hold:*

- *The aggregate residue V^t attenuates to 0 in a non-increasing way as t goes to $+\infty$.*
- *For every $i \in \mathcal{V}$, the limit of $(G_1^t, G_2^t, ..., G_n^t)$ at $t = +\infty$ is an optimal solution for $(W_1, W_2, ..., W_n)$.*

Proof. After realizing that (19) is obtained from a two-block ADMM procedure, the theorem follows from [20] that studies the convergence of a standard ADMM problem. The details are omitted for brevity. □

Since the proposed algorithm is iterative with an asymptotic convergence, we need a finite-time stopping rule. Based on [21], we terminate the algorithm as soon as $\max\{\text{P}_1, \text{P}_2, \text{D}_1, \text{D}_2, \text{D}_3, \text{D}_4, \text{Gap}\}$ becomes smaller than a pre-

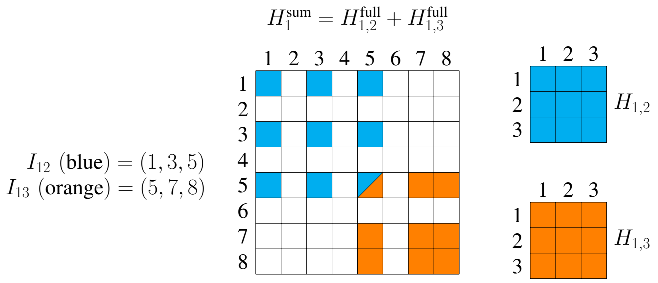
$H_1^{\text{sum}} = H_{1,2}^{\text{full}} + H_{1,3}^{\text{full}}$
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8
I_{12} (blue) $= (1,3,5)$
I_{13} (orange) $= (5,7,8)$
1 2 3
1 2 3
$H_{1,2}$
1 2 3
1 2 3
$H_{1,3}$

Fig. 4: An illustration of the difference between $H_{i,j}^{\text{full}}$, $H_{i,j}$ and H_i^{sum}. Agent 1 is overlapping with agents 2 and agent 3 at the entries specified by I_{12} and I_{23}. The white squares in the left matrix $H_{1,2}^{\text{full}} + H_{1,3}^{\text{full}}$ represent those entries with value 0, and the color squares carry Lagrange multipliers.

specified tolerance, where

$$(\mathrm{P}_1)_i = \frac{\left\|\overline{B}_i^T\overline{W}_i - c_i\right\|_2 + \left\|\max\left(\overline{D}_i^T\overline{W}_i - d_i, \mathbf{0}\right)\right\|_2}{1+\|c_i\|_2} \tag{22a}$$

$$(\mathrm{P}_2)_{i,j} = \frac{\|W_i(I_{ij},I_{ij}) - W_j(I_{ji},I_{ji})\|_F}{1+\|W_i(I_{ij},I_{ij})\|_F + \|W_j(I_{ji},I_{ji})\|_F} \tag{22b}$$

$$(\mathrm{D}_1)_i = \frac{\|-B_i^{\text{sum}} - D_i^{\text{sum}} + R_i - H_i^{\text{sum}} - A_i\|_F}{1+\|A_i\|_1} \tag{22c}$$

$$(\mathrm{D}_2)_{i,j} = \frac{\left\|H_{i,j} - H^{(i,j)}\right\|_F}{1+\|H_{i,j}\|_F + \left\|H^{(i,j)}\right\|_F} \tag{22d}$$

$$(\mathrm{D}_3)_{i,j} = \frac{\left\|H_{j,i} - H^{(i,j)}\right\|_F}{1+\|H_{j,i}\|_F + \left\|H^{(i,j)}\right\|_F} \tag{22e}$$

$$(\mathrm{D}_4)_i = \frac{\|v_i - u_i\|_2}{1+\|v_i\|_2 + \|u_i\|_2} \tag{22f}$$

$$\text{Gap} = \frac{\left|\sum_{i\in\mathcal{V}}\left(c_i^T z_i + d_i^T v_i - \mathbf{tr}\left(A_i W_i\right)\right)\right|}{1+\left|\sum_{i\in\mathcal{V}}\left(c_i^T z_i + d_i^T v_i\right)\right| + \left|\sum_{i\in\mathcal{V}}\mathbf{tr}\left(A_i W_i\right)\right|} \tag{22g}$$

for every $i \in \mathcal{V}$ and $(i,j) \in \mathcal{E}^+$, where

- the letters P and D refer to the primal and dual infeasibilities, respectively.
- $\overline{W}_i$ is the vectorized version of W_i obtained by stacking the columns of W_i one under another to create a column vector.
- $\overline{B}_i$ and $\overline{D}_i$ are matrices whose columns are the vectorized versions of $B_j^{(i)}$ and $D_l^{(i)}$ for $j = 1,\ldots,p_i$ and $l = 1,\ldots,q_i$, respectively.

The stopping criteria in (22) are based on the primal and dual infeasibilities as well as the duality gap.

V. Simulations Results

The objective of this section is to elucidate the results of this work on randomly generated large-scale structured SDP problems. A prototype of the algorithm was implemented in MATLAB and all of the simulations below were run on a laptop with an Intel Core i7 quad-core 2.5 GHz CPU and 8 GB RAM.

For every $i \in \mathcal{V}$, we generate a random instance of the problem as follows:

- Each matrix A_i is chosen as $\Omega + \Omega^T + n_i I$, where the entries of Ω are uniformly chosen from the integer set $\{1,2,3,4,5\}$. This creates reasonably well-conditioned matrices A_i.
- Each matrix B_j(or D_l) is chosen as $\Omega + \Omega^T$, where Ω is generated as before.
- Each matrix variable W_i is assumed to be 40 by 40.
- The matrices $W_1, ..., W_n$ are assumed to overlap with each other in a banded structure, associated with a path graph $\mathcal{G}$ with the edges $(1,2), (2,3), ..., (n-1,n)$. One can regard W_i's as submatrices of a full-scale matrix variable W in the form of Figure 3 but with n overlapping blocks, where 25% of the entries of every two neighboring matrices W_i and W_{i+1} (leading to a 10×10 submatrix) overlaps.

In order to demonstrate the proposed algorithm on large-scale SDPs, three different values will be considered for the total number of overlapping blocks (or agents): 1000, 2000 and 4000. To give the reader a sense of how large the simulated SDPs are, the total number of entries of W_i's in the decomposed SDP problem (N_{Decomp}) and the total number of entries of W in the corresponding full-SDP problem (N_{Full}) are listed below:

- 1000 agents: $N_{\text{Full}} = 0.9$ billion, $N_{\text{Decomp}} = 1.6$ million
- 2000 agents: $N_{\text{Full}} = 3.6$ billion, $N_{\text{Decomp}} = 3.2$ million
- 4000 agents: $N_{\text{Full}} = 14.4$ billion, $N_{\text{Decomp}} = 6.4$ million

The simulation results are provided in Table I with the following entries: P_{obj} and D_{obj} are the primal and dual objective values, "iter" denotes the number of iterations needed to achieve a desired tolerance, t_{CPU} and t_{iter} are the total CPU time (in seconds) and the time per iteration (in seconds per iteration), and "Optimality" (in percentage) is calculated as:

$$\text{Optimality Degree } (\%) = 100 - \frac{P_{\text{obj}} - D_{\text{obj}}}{P_{\text{obj}}} \times 100$$

As shown in Table I, the simulations were run for three cases:

- $p_i = 5$ and $q_i = 0$: each agent has 5 equality constraints and no inequality constraints.
- $p_i = 0$ and $q_i = 5$: each agent has no equality constraints and 5 inequality constraints.
- $p_i = 5$ and $q_i = 5$: each agent has 5 equality constraints and 5 inequality constraints.

All solutions reported in Table I are based on the tolerance of 10^{-3} and an optimality degree of at least 99.9%. The aggregative residue V^t is plotted in Figure 5 for the 4000-agent case with $p_i = q_i = 5$, which is a monotonically decreasing function. Note that the time per iteration is between 1.66 and 18.03 in a MATLAB implementation, which can be reduced significantly in C++. Efficient and computationally cheap preconditioning methods could dramatically reduce the number of iterations, but this is outside the scope of this paper.

VI. Conclusion

This paper develops a fast, parallelizable algorithm for an arbitrary decomposable semidefinite program (SDP). To formulate a decomposable SDP, we consider a multi-agent canonical form represented by a graph, where each agent (node) is

Cases		1000	2000	4000
$p_i = 5$ $q_i = 0$	P_{obj}	4.897e+5	9.864e+5	1.9664e+6
	D_{obj}	4.896e+5	9.863e+5	1.9661e+6
	iter	263	257	278
	t_{CPU} (sec)	437.56	853.75	4180.10
	t_{iter} (sec per iter)	1.66	3.32	15.04
	Optimality	99.98%	99.98%	99.98%
$p_i = 0$ $q_i = 5$	P_{obj}	8.2363e+5	1.64756e+6	3.27674e+6
	D_{obj}	8.2362e+5	1.64755e+6	3.27674e+6
	iter	1081	1127	1299
	t_{CPU} (sec)	1828.19	4353.49	19699.84
	t_{iter} (sec per iter)	1.69	3.86	15.16
	Optimality	99.998%	99.9994%	99.9996%
$p_i = 5$ $q_i = 5$	P_{obj}	5.313e+5	1.0673e+6	2.1284e+6
	D_{obj}	5.312e+5	1.0672e+6	2.1282e+6
	iter	291	409	473
	t_{CPU} (sec)	547.62	1416.83	8530.38
	t_{iter} (sec per iter)	1.88	3.46	18.03
	Optimality	99.98%	99.991%	99.993%

TABLE I: Simulation results for three cases with 1000, 2000 and 4000 agents.

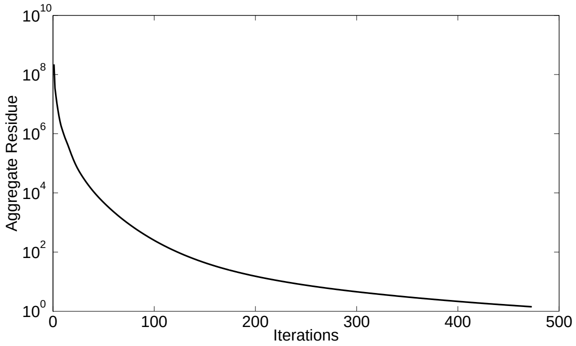

Fig. 5: Aggregate residue for the case of 4000 agents with $p_i = q_i = 5$.

in charge of computing its corresponding positive semidefinite matrix. The main goal of each agent is to ensure that its matrix is optimal with respect to some measure and satisfies local equality and inequality constraints. In addition, the matrices of two neighboring agents may be subject to overlapping constraints. The objective function of the optimization is the sum of all objectives of individual agents. The motivation behind this formulation is that an arbitrary sparse SDP problem can be converted to a decomposable SDP by means of the Chordal extension and matrix completion theorems. Using the alternating direction method of multipliers, we develop a distributed algorithm to solve the underlying SDP problem. At every iteration, each agent performs simple computations (matrix multiplication and eigenvalue decomposition) without having to solve any optimization subproblem, and then communicates some information to its neighbors. By deriving a Lyapunov-type non-increasing function, it is shown that the proposed algorithm converges as long as Slater's conditions hold. Simulations results on large-scale SDP problems with a few million variables are offered to elucidate the efficacy of this work.

References

[1] D. Gabay and B. Mercier, "A dual algorithm for the solution of nonlinear variational problems via finite element approximation," *Computers and Mathematics with Applications*, vol. 2, no. 1, pp. 17 – 40, 1976.

[2] R. Glowinski and A. Marroco, "Sur l'approximation, par lments finis d'ordre un, et la rsolution, par pnalisation-dualit d'une classe de problmes de dirichlet non linaires," *ESAIM: Mathematical Modelling and Numerical Analysis - Modlisation Mathmatique et Analyse Numrique*, vol. 9, no. R2, pp. 41–76, 1975. [Online]. Available: http://eudml.org/doc/193269

[3] J. Eckstein and W. Yao, "Augmented lagrangian and alternating direction methods for convex optimization: A tutorial and some illustrative computational results," *RUTCOR Research Reports*, vol. 32, 2012.

[4] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[5] Y. Nesterov, "A method of solving a convex programming problem with convergence rate $O(1/k^2)$," *Soviet Mathematics Doklady*, vol. 27, no. 2, pp. 372–376, 1983.

[6] T. Goldstein, B. O'Donoghue, S. Setzer, and R. Baraniuk, "Fast alternating direction optimization methods," *SIAM Journal on Imaging Sciences*, vol. 7, no. 3, pp. 1588–1623, 2014.

[7] P. Giselsson and S. Boyd, "Diagonal scaling in Douglas–Rachford splitting and ADMM," *53rd IEEE Conference on Decision and Control*, 2014.

[8] B. He and X. Yuan, "On the $O(1/n)$ convergence rate of the Douglas–Rachford alternating direction method," *SIAM Journal on Numerical Analysis*, vol. 50, no. 2, pp. 700–709, 2012.

[9] R. D. C. Monteiro and B. F. Svaiter, "Iteration-complexity of block-decomposition algorithms and the alternating direction method of multipliers," *SIAM Journal on Optimization*, vol. 23, no. 1, pp. 475–507, 2013.

[10] E. Wei and A. Ozdaglar, "On the $O(1/k)$ Convergence of Asynchronous Distributed Alternating Direction Method of Multipliers," *ArXiv e-prints*, Jul. 2013.

[11] R. Nishihara, L. Lessard, B. Recht, A. Packard, and M. I. Jordan, "A general analysis of the convergence of ADMM," *arXiv preprint arXiv:1502.02009*, 2015.

[12] J. Lavaei and S. H. Low, "Zero duality gap in optimal power flow problem," *IEEE Transactions on Power Systems*, vol. 27, no. 1, pp. 92–107, 2012.

[13] M. X. Goemans and D. P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *Journal of the ACM (JACM)*, vol. 42, no. 6, pp. 1115–1145, 1995.

[14] L. Vandenberghe and S. Boyd, "Semidefinite programming," *SIAM Review*, vol. 38, pp. 49–95, 1994.

[15] R. Madani, G. Fazelnia, S. Sojoudi, and J. Lavaei, "Low-rank solutions of matrix inequalities with applications to polynomial optimization and matrix completion problems," *IEEE Conference on Decision and Control*, 2014.

[16] M. Fukuda, M. Kojima, K. Murota, and K. Nakata, "Exploiting sparsity in semidefinite programming via matrix completion I: General framework," *SIAM Journal on Optimization*, vol. 11, no. 3, pp. 647–674, 2001.

[17] R. Grone, C. R. Johnson, E. M. Sá, and H. Wolkowicz, "Positive definite completions of partial hermitian matrices," *Linear algebra and its applications*, vol. 58, pp. 109–124, 1984.

[18] Z. Wen, D. Goldfarb, and W. Yin, "Alternating direction augmented lagrangian methods for semidefinite programming," *Mathematical Programming Computation*, vol. 2, no. 3-4, pp. 203–230, 2010.

[19] Y. Sun, M. S. Andersen, and L. Vandenberghe, "Decomposition in conic optimization with partially separable structure," *SIAM Journal on Optimization*, vol. 24, no. 2, pp. 873–897, 2014.

[20] B. He and X. Yuan, "On non-ergodic convergence rate of Douglas–Rachford alternating direction method of multipliers," *Numerische Mathematik*, pp. 1–11, 2012.

[21] H. Mittelmann, "An independent benchmarking of SDP and SOCP solvers," *Mathematical Programming*, vol. 95, no. 2, pp. 407–430, 2003. [Online]. Available: http://dx.doi.org/10.1007/s10107-002-0355-5