Noname manuscript No.
(will be inserted by the editor)

A Fast Distributed Algorithm for Sparse Semidefinite Programs

Abdulrahman Kalbat · Javad Lavaei

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 coupling and overlapping (consistency) constraints with regards to the agent's neighbors. Every arbitrary SDP problem can be cast as a decomposable SDP, where the number of nodes and the connectivity of the underlying graph both depend on the sparsity of the original SDP problem. 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 large-scale sparse conic optimization problems. The proposed algorithm is applied to several randomly generated SDP problems with over 10 million variables, which are solved in less than 20 minutes.

Keywords Semidefinite programming · Alternating direction method of multipliers · Decomposition

Mathematics Subject Classification (2000) 90C06 · 90C222 · 68W15

Abdulrahman Kalbat
Electrical Engineering Department, United Arab Emirates University
Al-Ain, United Arab Emirates
E-mail: akalbat@uaeu.ac.ae

Javad Lavaei (✉)
Department of Industrial Engineering and Operations Research, University of California, Berkeley
Berkeley, CA 94720
E-mail: lavaei@berkeley.edu

1 Introduction

Alternating direction method of multipliers (ADMM) is a first-order optimization algorithm proposed in the mid-1970s [7,9]. Recently, this method has attracted significant amount of attention since it can be used for large-scale optimization problems and be implemented in parallel and distributed computational environments [5,3]. Compared to second-order methods that are able to achieve a high accuracy via expensive iterations, ADMM relies on low-complexity iterations and can achieve a modest accuracy in tens of iterations. Inspired by Nesterov's scheme for accelerating gradient methods [22], great effort has been devoted to accelerating ADMM and attaining a high accuracy in a reasonable number of iterations [11]. Since the performance of the ADMM algorithm is affected by the condition number of the problem's data, diagonal rescaling is proposed in [8] 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 [13,20] under the assumptions of closed convex sets and convex functions. In [27], the $\mathcal{O}(\frac{1}{n})$ convergence rate is obtained for an asynchronous ADMM algorithm. The recent paper [23] represents ADMM in the context of dynamical systems and then reduces the problem of proving the linear convergence of ADMM to verifying the stability of a dynamical system [23].

Semidefinite programs (SDPs) are attractive due in part to three reasons. First, positive semidefinite constraints appear in many applications [16]. Second, SDPs can be used to study and approximate hard combinatorial optimization problems [10]. 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 [26]. However, these methods are less practical for large-scale SDPs due to computation time and memory issues. However, it is possible to reduce the complexity by exploiting possible structures 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 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 coupling and 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 computational perspective, each agent is treated as a processing unit and each edge of the graph specifies what agents can communicate with one another. 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 2. The distributed multi-agent SDP problem is formalized in Section 3. An ADMM-based parallel algorithm is developed in Section 4, 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 5. A method for distributing the computational load is proposed in Section 6. Finally, a summary is given in Section 7.

Notations: $\mathbb{R}^n$ and $\mathbb{S}^n$ denote the sets of $n\times 1$ real vectors and $n\times n$ symmetric matrices, respectively. Bold lower case letters (e.g., $\mathbf{x}$) represent vectors, and bold upper case letters (e.g., $\mathbf{W}$) represent matrices. A vector $\mathbf{x}\in\mathbb{R}^n$ is defined as $\mathbf{x}=[x_1,\ldots,x_n]^T$. Given a matrix $\mathbf{W}$, its (l,m) entry is denoted as $W(l,m)$. The symbol $\mathbf{tr}\{\mathbf{W}\}$ denotes the trace of a matrix $\mathbf{W}$, and the notation $\mathbf{W}\succeq 0$ means that $\mathbf{W}$ is symmetric and positive semidefinite. The symbols $(\cdot)^T$, $\|\cdot\|_2$ and $\|\cdot\|_F$ denote the transpose, ℓ_2-norm and Frobenius norm 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 sequence of variables $x_1,\ldots,x_n$ is denoted by $\{x_i\}_{i=1}^n$. For an $m\times n$ matrix $\mathbf{W}$, the notation $\mathbf{W}(\mathcal{X},\mathcal{Y})$ denotes the submatrix of $\mathbf{W}$ whose rows and columns are chosen from $\mathcal{X}$ 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 open set of neighbors of vertex $i\in\mathcal{V}$ is denoted as $N(i)$. The closed set of neighbors of vertex $i\in\mathcal{V}$ is denoted as $N[i]$, which is simply the neighbors of node i including node i itself ($N[i]=N(i)\cup\{i\}$). To orient the edges of $\mathcal{G}$, we define a new edge set $\mathcal{E}^+=\{(i,j)\mid(i,j)\in\mathcal{E}\ \text{ and }\ i<j\}$.

2 Alternating Direction Method of Multipliers

Consider the optimization problem

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

$$\text{subject to} \quad \mathbf{A}\mathbf{x}+\mathbf{B}\mathbf{y}=\mathbf{c} \tag{1b}$$

where $f(\mathbf{x})$ and $g(\mathbf{y})$ are convex functions, $\mathbf{A}$ and $\mathbf{B}$ are two known matrices, and $\mathbf{c}$ 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

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

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

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

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

$$\boldsymbol{\lambda}^{t+1} := \boldsymbol{\lambda}^t + \boldsymbol{\alpha}^t(\mathbf{A}\mathbf{x}^{t+1}+\mathbf{B}\mathbf{y}^{t+1}-\mathbf{c}) \tag{3c}$$

for $t = 0,1,2,...$, with an arbitrary initialization $(\mathbf{x}^0,\mathbf{y}^0,\boldsymbol{\lambda}^0)$, where $\boldsymbol{\alpha}^t$ 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

$$\mathcal{L}_\mu(\mathbf{x},\mathbf{y},\boldsymbol{\lambda}) = f(\mathbf{x}) + g(\mathbf{y}) + \boldsymbol{\lambda}^T(\mathbf{A}\mathbf{x}+\mathbf{B}\mathbf{y}-\mathbf{c}) + \frac{\mu}{2}\|\mathbf{A}\mathbf{x}+\mathbf{B}\mathbf{y}-\mathbf{c}\|_2^2 \tag{4}$$

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 $(\mathbf{x},\mathbf{y})$. However, this augmentation creates a coupling between $\mathbf{x}$ and $\mathbf{y}$. The iterations corresponding to the method of multipliers are

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

$$\boldsymbol{\lambda}^{t+1} := \boldsymbol{\lambda}^t + \mu(\mathbf{A}\mathbf{x}^{t+1}+\mathbf{B}\mathbf{y}^{t+1}-\mathbf{c}) \tag{5b}$$

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

In order to avoid solving a joint optimization with respect to $\mathbf{x}$ and $\mathbf{y}$ at every iteration, the *alternating direction method of multipliers* (ADMM) can be used. The main idea is to first update $\mathbf{x}$ by freezing $\mathbf{y}$ at its latest value, and then update $\mathbf{y}$ based on the most recent value of $\mathbf{x}$. This leads to the 2-block ADMM problem with the iterations [3]:

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

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

$$\text{Dual:}\quad \boldsymbol{\lambda}^{t+1} := \boldsymbol{\lambda}^t + \mu(\mathbf{A}\mathbf{x}^{t+1}+\mathbf{B}\mathbf{y}^{t+1}-\mathbf{c}) \tag{6c}$$

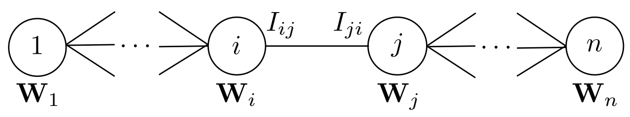


Fig. 1: An example of a graph representation of the distributed multi-agent SDP.

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.

3 Problem Formulation

Consider an arbitrary 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 illustrated 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 work, each agent is in charge of computing a positive semidefinite matrix variable $\mathbf{W}_i$, and each edge $(i, j) \in \mathcal{E}$ specifies a possible overlap between the matrix variables $\mathbf{W}_i$ and $\mathbf{W}_j$ of agents i and j. More precisely, each edge (i, j) is accompanied by two arbitrary integer-valued index sets $I_{i,j}$ and $I_{j,i}$ to capture the overlap between $\mathbf{W}_i$ and $\mathbf{W}_j$ through the equation $\mathbf{W}_i(I_{i,j}, I_{i,j}) = \mathbf{W}_j(I_{j,i}, I_{j,i})$. 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 $\mathbf{W}$, which is then reformulated in terms of certain submatrices of $\mathbf{W}$, named $\mathbf{W}_1, ..., \mathbf{W}_n$, using the Chordal extension and matrix completion theorems [18]. The objective of this paper is to solve the decomposable SDP problem (interchangeably referred to as distributed multi-agent SDP) given below.

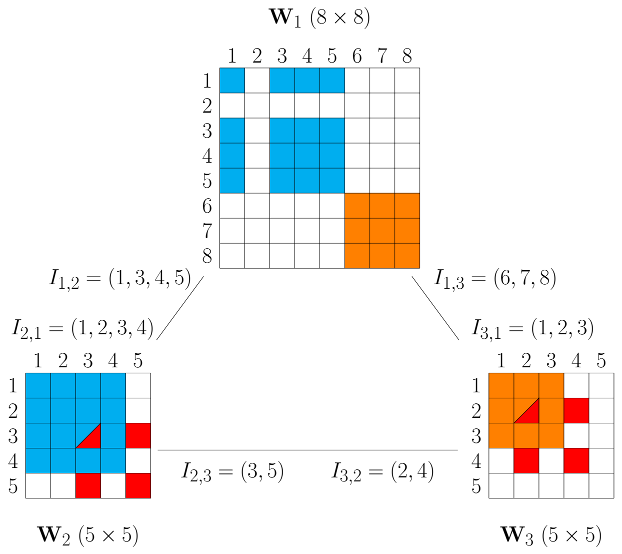


Fig. 2: An illustration of the definitions of $I_{i,j}$ and $I_{j,i}$ for three overlapping submatrices $\mathbf{W}_1$, $\mathbf{W}_2$ and $\mathbf{W}_3$

Decomposable SDP:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i\in\mathcal{V}} \mathbf{tr}(\mathbf{A}_i\mathbf{W}_i) && && \text{(7a)}\\
\text{subject to:} \quad & \mathbf{tr}(\mathbf{B}_j^i\mathbf{W}_i)=b_j^i && \forall\ j=1,\ldots,p_i \ \text{ and } \ i\in\mathcal{V} && \text{(7b)}\\
& \mathbf{tr}(\mathbf{C}_j^i\mathbf{W}_i)\leq c_j^i && \forall\ j=1,\ldots,q_i \ \text{ and } \ i\in\mathcal{V} && \text{(7c)}\\
& \mathbf{W}_i\succeq 0 && \forall\ i\in\mathcal{V} && \text{(7d)}\\
& \sum_{j\in N[i]} \mathbf{tr}(\mathbf{D}_k^{i,j}\mathbf{W}_j)=d_k^{(i)} && \forall\ k=1,\ldots,r_i \ \text{ and } \ i\in\mathcal{V} && \text{(7e)}\\
& \sum_{j\in N[i]} \mathbf{tr}(\mathbf{E}_k^{i,j}\mathbf{W}_j)\leq e_k^{(i)} && \forall\ k=1,\ldots,s_i \ \text{ and } \ i\in\mathcal{V} && \text{(7f)}\\
& \mathbf{W}_i(I_{i,j},I_{i,j})=\mathbf{W}_j(I_{j,i},I_{j,i}) && \forall\ (i,j)\in\mathcal{E}^+ && \text{(7g)}
\end{aligned}
$$

with the variables $\mathbf{W}_i\in\mathbb{S}^{n_i}$ for $i=1,\ldots,n$, where

- the superscript in $(\cdot)^i$ is not an exponent and it implies that the expression is local to agent $i\in\mathcal{V}$;
- the superscript in $(\cdot)^{(i)}$ means that the expression is local to agent $i\in\mathcal{V}$ and yet involved in a coupling constraint with other agents;
- n_i denotes the size of the submatrix $\mathbf{W}_i$, and p_i, q_i, r_i and s_i show the numbers of local equality, local inequality, coupling equality and coupling inequality constraints for agent i, respectively;

- b_j^i, c_j^i, $d_k^{(i)}$ and $e_k^{(i)}$ denote the j^{th} elements of the vectors $\mathbf{b}_i \in \mathbb{R}^{p_i}$ and $\mathbf{c}_i \in \mathbb{R}^{q_i}$ and the k^{th} elements of the vectors $\mathbf{d}_{(i)} \in \mathbb{R}^{r_i}$ and $\mathbf{e}_{(i)} \in \mathbb{R}^{s_i}$ for agent i, as defined below:

$$\mathbf{b}_i \triangleq [b_1^i, \ldots, b_{p_i}^i]^T, \qquad \mathbf{c}_i \triangleq [c_1^i, \ldots, c_{q_i}^i]^T$$

$$\mathbf{d}_{(i)} \triangleq [d_1^{(i)}, \ldots, d_{r_i}^{(i)}]^T, \qquad \mathbf{e}_{(i)} \triangleq [e_1^{(i)}, \ldots, e_{s_i}^{(i)}]^T$$

- the matrices $\mathbf{A}_i$, $\mathbf{B}_j^i$, and $\mathbf{C}_j^i$ are locally known to agent $i \in \mathcal{V}$ and the matrices $\mathbf{D}_k^{i,j}$ and $\mathbf{E}_k^{i,j}$ are involved in coupling constraints with agent $i \in \mathcal{V}$ and are locally known to agent $j \in N[i]$.

The formulation in (7) has four main ingredients:

- **Local objective function:** each agent $i \in \mathcal{V}$ has its own local objective function $\mathbf{tr}(\mathbf{A}_i \mathbf{W}_i)$ with respect to the local matrix variable $\mathbf{W}_i$. 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).
- **Coupling constraints:** each agent $i \in \mathcal{V}$ is allowed to have coupling equality and inequality constraints (7e) and (7f), respectively, with any other agent $j \in N(i)$.
- **Overlapping constraints:** constraint (7g) states that certain entries of $\mathbf{W}_i$ and $\mathbf{W}_j$ 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 $\mathbf{W}_i$ while meeting its own constraints as well as all coupling and overlapping constraints with the other agents. 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 $\mathbf{W}$, and then convert it into a distributed SDP with multiple but smaller positive semidefinite matrices $\mathbf{W}_i$ 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 [6], which exploits the fact that a matrix $\mathbf{W}$ has a positive semidefinite completion if and only if certain submatrices of $\mathbf{W}$, denoted as $\mathbf{W}_1, ..., \mathbf{W}_n$, are positive semidefinite [12].

In this paper, we propose an iterative algorithm for solving the decomposable SDP problem (7) using the first-order ADMM method. We show that

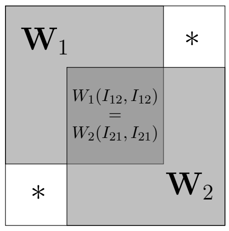


Fig. 3: Positive semidefinite matrix $\mathbf{W}$ (two blocks)

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. The paper [29] 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. The work [6] uses the clique-tree conversion method to decompose sparse SDPs with a 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, the paper [25] proposes to solve the decomposed SDP created by [6] 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).

4 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 $\mathbf{W}_1$ and $\mathbf{W}_2$. Later on, we will derive the iterations for the general case with an arbitrary graph $\mathcal{G}$.

4.1 Two-Agent Case

Assume that there are two overlapping matrices $\mathbf{W}_1$ and $\mathbf{W}_2$ embedded in a global SDP matrix variable $\mathbf{W}$ as shown in Figure 3, where "*" submatrices of $\mathbf{W}$ 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 $|\mathcal{V}| = 2$:

$$
\begin{aligned}
\min_{\substack{\mathbf{W}_1 \in \mathbb{S}^{n_1} \\ \mathbf{W}_2 \in \mathbb{S}^{n_2}}} \quad & \mathbf{tr}(\mathbf{A}_1\mathbf{W}_1) + \mathbf{tr}(\mathbf{A}_2\mathbf{W}_2) && && \text{(8a)}\\
\text{subject to} \quad & \mathbf{tr}(\mathbf{B}_j^1\mathbf{W}_1) = b_j^1 && \forall\, j = 1, \ldots, p_1 && \text{(8b)}\\
& \mathbf{tr}(\mathbf{B}_j^2\mathbf{W}_2) = b_j^2 && \forall\, j = 1, \ldots, p_2 && \text{(8c)}\\
& \mathbf{tr}(\mathbf{C}_j^1\mathbf{W}_1) \le c_j^1 && \forall\, j = 1, \ldots, q_1 && \text{(8d)}\\
& \mathbf{tr}(\mathbf{C}_j^2\mathbf{W}_2) \le c_j^2 && \forall\, j = 1, \ldots, q_2 && \text{(8e)}\\
& \mathbf{tr}(\mathbf{D}_k^{1,1}\mathbf{W}_1) + \mathbf{tr}(\mathbf{D}_k^{1,2}\mathbf{W}_2) = d_k^{(1)} && \forall\, k = 1, \ldots, r_1 && \text{(8f)}\\
& \mathbf{tr}(\mathbf{D}_k^{2,1}\mathbf{W}_1) + \mathbf{tr}(\mathbf{D}_k^{2,2}\mathbf{W}_2) = d_k^{(2)} && \forall\, k = 1, \ldots, r_2 && \text{(8g)}\\
& \mathbf{tr}(\mathbf{E}_k^{1,1}\mathbf{W}_1) + \mathbf{tr}(\mathbf{E}_k^{1,2}\mathbf{W}_2) \le e_k^{(1)} && \forall\, k = 1, \ldots, s_1 && \text{(8h)}\\
& \mathbf{tr}(\mathbf{E}_k^{2,1}\mathbf{W}_1) + \mathbf{tr}(\mathbf{E}_k^{2,2}\mathbf{W}_2) \le e_k^{(2)} && \forall\, k = 1, \ldots, s_2 && \text{(8i)}\\
& \mathbf{W}_1, \mathbf{W}_2 \succeq 0 && && \text{(8j)}\\
& \mathbf{W}_1(I_{1,2}, I_{1,2}) = \mathbf{W}_2(I_{2,1}, I_{2,1}) && && \text{(8k)}
\end{aligned}
$$

where the data matrices $\mathbf{A}_1$, $\mathbf{B}_j^1$, $\mathbf{C}_j^1$, $\mathbf{D}_k^{1,1}$, $\mathbf{D}_k^{2,1}$, $\mathbf{E}_k^{1,1}$, $\mathbf{E}_k^{2,1} \in \mathbb{S}^{n_1}$, the matrix variable $\mathbf{W}_1 \in \mathbb{S}^{n_1}$ and the vectors $\mathbf{b}_1 \in \mathbb{R}^{p_1}$, $\mathbf{c}_1 \in \mathbb{R}^{q_1}$, $\mathbf{d}_{(1)} \in \mathbb{R}^{r_1}$ and $\mathbf{e}_{(1)} \in \mathbb{R}^{s_1}$ correspond to agent 1, whereas the data matrices $\mathbf{A}_2$, $\mathbf{B}_j^2$, $\mathbf{C}_j^2$, $\mathbf{D}_k^{2,2}$, $\mathbf{D}_k^{1,2}$, $\mathbf{E}_k^{2,2}$, $\mathbf{E}_k^{1,2} \in \mathbb{S}^{n_2}$, the matrix variable $\mathbf{W}_2 \in \mathbb{S}^{n_2}$ and the vectors $\mathbf{b}_2 \in \mathbb{R}^{p_2}$, $\mathbf{c}_2 \in \mathbb{R}^{q_2}$, $\mathbf{d}_{(2)} \in \mathbb{R}^{r_2}$ and $\mathbf{e}_{(2)} \in \mathbb{R}^{s_2}$ correspond to agent 2. The local constraints of agent 1 and agent 2 are represented by (8b)-(8e) and (8j) and the coupling constraints between the two agents is represented by (8f)-(8i). Constraint (8k) states that the $(I_{1,2}, I_{1,2})$ submatrix of $\mathbf{W}_1$ overlaps with the $(I_{2,1}, I_{2,1})$ submatrix of $\mathbf{W}_2$. With no loss of generality, assume that the overlapping part occurs at the lower right corner of $\mathbf{W}_1$ and the upper left corner of $\mathbf{W}_2$, as illustrated in Figure 3. The dual of the 2-agent SDP problem in (8) can be expressed as

$$\min \quad \sum_{i=1}^{2} \left(\mathbf{b}_i^T\mathbf{u}_i + \mathbf{c}_i^T\mathbf{v}_i + \mathbf{d}_{(i)}^T\mathbf{x}_{(i)} + \mathbf{e}_{(i)}^T\mathbf{y}_{(i)}\right) \tag{9a}$$

subject to

$$-\sum_{j=1}^{p_1} u_j^1\mathbf{B}_j^1 - \sum_{j=1}^{q_1} v_j^1\mathbf{C}_j^1 - \sum_{j=1}^{r_1} x_j^{(1)}\mathbf{D}_j^{1,1} - \sum_{j=1}^{r_2} x_j^{(2)}\mathbf{D}_j^{2,1} - \sum_{j=1}^{s_1} y_j^{(1)}\mathbf{E}_j^{1,1} - \sum_{j=1}^{s_2} y_j^{(2)}\mathbf{E}_j^{2,1} + \mathbf{R}_1 - \begin{bmatrix} 0 & 0 \\ 0 & \mathbf{H}_{(1,2)} \end{bmatrix} = \mathbf{A}_1 \tag{9b}$$

$$-\sum_{j=1}^{p_2} u_j^2\mathbf{B}_j^2 - \sum_{j=1}^{q_2} v_j^2\mathbf{C}_j^2 - \sum_{j=1}^{r_2} x_j^{(2)}\mathbf{D}_j^{2,2} - \sum_{j=1}^{r_1} x_j^{(1)}\mathbf{D}_j^{1,2} - \sum_{j=1}^{s_2} y_j^{(2)}\mathbf{E}_j^{2,2} - \sum_{j=1}^{s_1} y_j^{(1)}\mathbf{E}_j^{1,2} + \mathbf{R}_2 - \begin{bmatrix} \mathbf{H}_{(1,2)} & 0 \\ 0 & 0 \end{bmatrix} = \mathbf{A}_2 \tag{9c}$$

$$\mathbf{v}_1, \mathbf{v}_2 \geq 0 \tag{9d}$$

$$\mathbf{y}_{(1)}, \mathbf{y}_{(2)} \geq 0 \tag{9e}$$

$$\mathbf{R}_1, \mathbf{R}_2 \succeq 0 \tag{9f}$$

with the variables $\mathbf{u}_1$, $\mathbf{u}_2$, $\mathbf{v}_1$,$\mathbf{v}_2$,$\mathbf{x}_{(1)}$,$\mathbf{x}_{(2)}$,$\mathbf{y}_{(1)}$,$\mathbf{y}_{(2)}$,$\mathbf{R}_1$,$\mathbf{R}_2$, $\mathbf{H}_{(1,2)}$, where $\mathbf{u}_1 \in \mathbb{R}^{p_1}$, $\mathbf{u}_2 \in \mathbb{R}^{p_2}$, $\mathbf{v}_1 \in \mathbb{R}^{q_1}$, $\mathbf{v}_2 \in \mathbb{R}^{q_2}$, $\mathbf{x}_{(1)} \in \mathbb{R}^{r_1}$, $\mathbf{x}_{(2)} \in \mathbb{R}^{r_2}$, $\mathbf{y}_{(1)} \in \mathbb{R}^{s_1}$ and $\mathbf{y}_{(2)} \in \mathbb{R}^{s_2}$ are the Lagrange multipliers corresponding to the constraints in (8b)-(8i), respectively, and the dual matrix variables $\mathbf{R}_1 \in \mathbb{S}^{n_1}$ and $\mathbf{R}_2 \in \mathbb{S}^{n_2}$ are the Lagrange multiplier corresponding to the constraint (8j). The dual matrix variable $\mathbf{H}_{(1,2)}$ is the Lagrange multiplier corresponding to the constraint (8k). More specifically, it is the Lagrange multiplier corresponding to the submatrix $\mathbf{W}_1(I_{1,2}, I_{1,2})$ of $\mathbf{W}_1$ and at the same time the Lagrange multiplier corresponding to the submatrix $\mathbf{W}_2(I_{2,1}, I_{2,1})$ of $\mathbf{W}_2$ while preserving the overlap locations, as shown in (9b) and (9c), respectively.

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 the constraints (9b) and (9c) through the variables $\mathbf{x}_{(1)}$,$\mathbf{x}_{(2)}$,$\mathbf{y}_{(1)}$,$\mathbf{y}_{(2)}$, $\mathbf{H}_{(1,2)}$, which then requires updating these variables jointly by agent 1 and agent 2 and this destroys the parallel nature of the algorithm. This issue can be resolved by introducing a local copy of each coupling variable for each agent. For example, agent 1 introduces $\mathbf{x}_{1,1}$ as a local copy of $\mathbf{x}_{(1)}$ in the constraint (9b) and adds the constraint $\mathbf{x}_{1,1} = \mathbf{x}_{(1)}$. Similarly, agent 2 introduces $\mathbf{x}_{2,1}$ as a local copy of $\mathbf{x}_{(1)}$ in the constraint (9c) and adds the constraint $\mathbf{x}_{2,1} = \mathbf{x}_{(1)}$. Following the same technique for $\mathbf{x}_{(2)}$,$\mathbf{y}_{(1)}$,$\mathbf{y}_{(2)}$,$\mathbf{H}_{(1,2)}$, the constraints (9b) and (9c) become completely decoupled. Moreover, to make the update of $\mathbf{v}_1$ and $\mathbf{v}_2$ easier, we do not impose positivity constraints directly on $\mathbf{v}_1$ and $\mathbf{v}_2$ as in (9d). Instead, we impose the positivity on two new vectors $\mathbf{z}_1, \mathbf{z}_2 \geq 0$ and then add the additional constraints $\mathbf{v}_1 = \mathbf{z}_1$ and $\mathbf{v}_2 = \mathbf{z}_2$. By applying the previous modifications, (9) could be rewritten in a decomposable form as

$$\min \sum_{i=1}^{2}\Big(\mathbf{b}_i^T\mathbf{u}_i+\mathbf{c}_i^T\mathbf{v}_i+\mathbf{d}_{(i)}^T\mathbf{x}_{(i)}+\mathbf{e}_{(i)}^T\mathbf{y}_{(i)}+I_+(\mathbf{z}_i)+I_+(\mathbf{R}_i)+I_+(\mathbf{y}_{(i)})\Big) \quad (10a)$$

subject to

$$-\sum_{j=1}^{p_1}u_j^1\mathbf{B}_j^1-\sum_{j=1}^{q_1}v_j^1\mathbf{C}_j^1-\sum_{j=1}^{r_1}x_j^{1,1}\mathbf{D}_j^{1,1}-\sum_{j=1}^{r_2}x_j^{1,2}\mathbf{D}_j^{2,1}-\sum_{j=1}^{s_1}y_j^{1,1}\mathbf{E}_j^{1,1}-\sum_{j=1}^{s_2}y_j^{1,2}\mathbf{E}_j^{2,1}$$
$$+\mathbf{R}_1-\begin{bmatrix}0 & 0\\ 0 & \mathbf{H}_{1,2}\end{bmatrix}=\mathbf{A}_1 \quad (10b)$$

$$-\sum_{j=1}^{p_2}u_j^2\mathbf{B}_j^2-\sum_{j=1}^{q_2}v_j^2\mathbf{C}_j^2-\sum_{j=1}^{r_2}x_j^{2,2}\mathbf{D}_j^{2,2}-\sum_{j=1}^{r_1}x_j^{2,1}\mathbf{D}_j^{1,2}-\sum_{j=1}^{s_2}y_j^{2,2}\mathbf{E}_j^{2,2}-\sum_{j=1}^{s_1}y_j^{2,1}\mathbf{E}_j^{1,2}$$
$$+\mathbf{R}_2-\begin{bmatrix}\mathbf{H}_{2,1} & 0\\ 0 & 0\end{bmatrix}=\mathbf{A}_2 \quad (10c)$$

$$\mathbf{H}_{1,2}=\mathbf{H}_{(1,2)},\qquad \mathbf{H}_{2,1}=\mathbf{H}_{(1,2)} \quad (10d)$$
$$\mathbf{x}_{1,1}=\mathbf{x}_{(1)},\qquad \mathbf{x}_{2,1}=\mathbf{x}_{(1)} \quad (10e)$$
$$\mathbf{x}_{2,2}=\mathbf{x}_{(2)},\qquad \mathbf{x}_{1,2}=\mathbf{x}_{(2)} \quad (10f)$$
$$\mathbf{y}_{1,1}=\mathbf{y}_{(1)},\qquad \mathbf{y}_{2,1}=\mathbf{y}_{(1)} \quad (10g)$$
$$\mathbf{y}_{2,2}=\mathbf{y}_{(2)},\qquad \mathbf{y}_{1,2}=\mathbf{y}_{(2)} \quad (10h)$$
$$\mathbf{v}_1=\mathbf{z}_1 \quad (10i)$$
$$\mathbf{v}_2=\mathbf{z}_2 \quad (10j)$$

with the variables $\mathbf{u}_1$, $\mathbf{u}_2$, $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{z}_1$, $\mathbf{z}_2$, $\mathbf{x}_{1,1}$, $\mathbf{x}_{1,2}$, $\mathbf{x}_{2,1}$, $\mathbf{x}_{2,2}$, $\mathbf{y}_{1,1}$, $\mathbf{y}_{1,2}$, $\mathbf{y}_{2,1}$, $\mathbf{y}_{2,2}$, $\mathbf{x}_{(1)}$, $\mathbf{x}_{(2)}$, $\mathbf{y}_{(1)}$, $\mathbf{y}_{(2)}$, $\mathbf{R}_1$, $\mathbf{R}_2$, $\mathbf{H}_{1,2}$, $\mathbf{H}_{2,1}$, $\mathbf{H}_{(1,2)}$, where $I_+(\mathbf{R}_i)$ is equal to 0 if $\mathbf{R}_i \succeq 0$ and is $+\infty$ otherwise, and $I_+(\mathbf{z}_i)$ and $I_+(\mathbf{y}_{(i)})$ are equal to 0 if $\mathbf{z}_i, \mathbf{y}_{(i)} \geq 0$ and are $+\infty$ otherwise.

To streamline the presentation, define

$$\mathbf{B}_i^{\text{sum}}=\textstyle\sum_{j=1}^{p_i}u_j^i\mathbf{B}_j^i,\qquad \mathbf{C}_i^{\text{sum}}=\sum_{j=1}^{q_i}v_j^i\mathbf{C}_j^i,\qquad i=1,2$$

$$\mathbf{D}_i^{\text{sum}}=\textstyle\sum_{k=1}^{2}\sum_{j=1}^{r_k}x_j^{i,k}\mathbf{D}_j^{k,i},\qquad \mathbf{E}_i^{\text{sum}}=\sum_{k=1}^{2}\sum_{j=1}^{s_k}y_j^{i,k}\mathbf{E}_j^{k,i},\qquad i=1,2$$

and

$$\mathbf{H}_{1,2}^{\text{full}}=\begin{bmatrix}0 & 0\\ 0 & \mathbf{H}_{1,2}\end{bmatrix},\qquad \mathbf{H}_{2,1}^{\text{full}}=\begin{bmatrix}-\mathbf{H}_{2,1} & 0\\ 0 & 0\end{bmatrix}$$

Note that $\mathbf{B}_i^{\text{sum}}$, $\mathbf{C}_i^{\text{sum}}$, $\mathbf{D}_i^{\text{sum}}$, $\mathbf{E}_i^{\text{sum}}$, $\mathbf{H}_{1,2}^{\text{full}}$ and $\mathbf{H}_{2,1}^{\text{full}}$ are functions of the variables $\mathbf{u}_i$, $\mathbf{v}_i$, $\mathbf{x}_{i,k}$, $\mathbf{y}_{i,k}$, $\mathbf{H}_{1,2}$ and $\mathbf{H}_{2,1}$, respectively, but the arguments are dropped for notational simplicity. The augmented Lagrangian function for

(10) can be obtained as

$$
\begin{aligned}
&\frac{2}{\mu}\mathcal{L}_{\mu}\left(\mathcal{F},\mathcal{M}\right)= \\
&\frac{2}{\mu}\sum_{i=1}^{2}\Big(\mathbf{b}_i^T\mathbf{u}_i+\mathbf{c}_i^T\mathbf{v}_i+\mathbf{d}_{(i)}^T\mathbf{x}_{(i)}+\mathbf{e}_{(i)}^T\mathbf{y}_{(i)}+I_+(\mathbf{z}_i)+I_+(\mathbf{R}_i)+I_+(\mathbf{y}_{(i)})\Big) \\
&+\left\|-\mathbf{B}_1^{\text{sum}}-\mathbf{C}_1^{\text{sum}}-\mathbf{D}_1^{\text{sum}}-\mathbf{E}_1^{\text{sum}}+\mathbf{R}_1-\mathbf{H}_{1,2}^{\text{full}}-\mathbf{A}_1+\frac{\mathbf{G}_1}{\mu}\right\|_F^2 \\
&+\left\|-\mathbf{B}_2^{\text{sum}}-\mathbf{C}_2^{\text{sum}}-\mathbf{D}_2^{\text{sum}}-\mathbf{E}_2^{\text{sum}}+\mathbf{R}_2-\mathbf{H}_{2,1}^{\text{full}}-\mathbf{A}_2+\frac{\mathbf{G}_2}{\mu}\right\|_F^2 \\
&+\left\|\mathbf{H}_{1,2}-\mathbf{H}_{(1,2)}+\frac{\mathbf{G}_{1,2}}{\mu}\right\|_F^2+\left\|\mathbf{H}_{2,1}-\mathbf{H}_{(1,2)}+\frac{\mathbf{G}_{2,1}}{\mu}\right\|_F^2 \\
&+\left\|\mathbf{x}_{1,1}-\mathbf{x}_{(1)}+\frac{\mathbf{f}_{1,1}}{\mu}\right\|_2^2+\left\|\mathbf{x}_{2,1}-\mathbf{x}_{(1)}+\frac{\mathbf{f}_{2,1}}{\mu}\right\|_2^2 \\
&+\left\|\mathbf{x}_{2,2}-\mathbf{x}_{(2)}+\frac{\mathbf{f}_{2,2}}{\mu}\right\|_2^2+\left\|\mathbf{x}_{1,2}-\mathbf{x}_{(2)}+\frac{\mathbf{f}_{1,2}}{\mu}\right\|_2^2 \\
&+\left\|\mathbf{y}_{1,1}-\mathbf{y}_{(1)}+\frac{\mathbf{g}_{1,1}}{\mu}\right\|_2^2+\left\|\mathbf{y}_{2,1}-\mathbf{y}_{(1)}+\frac{\mathbf{g}_{2,1}}{\mu}\right\|_2^2 \\
&+\left\|\mathbf{y}_{2,2}-\mathbf{y}_{(2)}+\frac{\mathbf{g}_{2,2}}{\mu}\right\|_2^2+\left\|\mathbf{y}_{1,2}-\mathbf{y}_{(2)}+\frac{\mathbf{g}_{1,2}}{\mu}\right\|_2^2 \\
&+\left\|\mathbf{v}_1-\mathbf{z}_1+\frac{\boldsymbol{\lambda}_1}{\mu}\right\|_2^2+\left\|\mathbf{v}_2-\mathbf{z}_2+\frac{\boldsymbol{\lambda}_2}{\mu}\right\|_2^2
\end{aligned}
\tag{11}
$$

where $\mathcal{F}=\big(\mathbf{u}_1,\ \mathbf{u}_2,\mathbf{v}_1,\ \mathbf{v}_2,\ \mathbf{z}_1,\ \mathbf{z}_2,\mathbf{x}_{(1)},\ \mathbf{x}_{(2)},\ \mathbf{y}_{(1)},\ \mathbf{y}_{(2)},\ \mathbf{R}_1,\ \mathbf{R}_2,\ \mathbf{x}_{1,1},\ \mathbf{x}_{1,2},\ \mathbf{x}_{2,1},\ \mathbf{x}_{2,2},\ \mathbf{y}_{1,1},\ \mathbf{y}_{1,2},\ \mathbf{y}_{2,1},\ \mathbf{y}_{2,2},\ \mathbf{H}_{1,2},\ \mathbf{H}_{2,1},\ \mathbf{H}_{(1,2)}\big)$ is the set of optimization variables and $\mathcal{M}=(\mathbf{G}_1,\ \mathbf{G}_2,\ \mathbf{G}_{1,2},\ \mathbf{G}_{2,1},\ \mathbf{f}_{1,1},\ \mathbf{f}_{2,1},\ \mathbf{f}_{2,2},\ \mathbf{f}_{1,2},\ \mathbf{g}_{1,1},\ \mathbf{g}_{2,1},\ \mathbf{g}_{2,2},\ \mathbf{g}_{1,2},\ \boldsymbol{\lambda}_1,\ \boldsymbol{\lambda}_2)$ is the set of Lagrange multipliers whose elements correspond to constraints (10b) - (10j), respectively. Note that the augmented Lagrangian in (11) is obtained using the identity

$$
\mathbf{tr}\left[\mathbf{X}^T(\mathbf{A}-\mathbf{B})\right]+\frac{\mu}{2}\left\|\mathbf{A}-\mathbf{B}\right\|_F^2=\frac{\mu}{2}\left\|\mathbf{A}-\mathbf{B}+\frac{\mathbf{X}}{\mu}\right\|_F^2+\text{constant} \tag{12}
$$

In order to proceed, we need to split the set of optimization variables $\mathcal{F}$ into two blocks of variables. To this end, define $\mathcal{X}=(\mathbf{z}_1\,,\ \mathbf{z}_2,\ \mathbf{x}_{(1)},\ \mathbf{x}_{(2)},\ \mathbf{y}_{(1)},\ \mathbf{y}_{(2)},\ \mathbf{R}_1,\ \mathbf{R}_2,\ \mathbf{H}_{(1,2)}\big)$ and $\mathcal{Y}=(\mathbf{u}_1,\ \mathbf{u}_2,\ \mathbf{v}_1,\ \mathbf{v}_2,\ \mathbf{x}_{1,1},\ \mathbf{x}_{1,2},\ \mathbf{x}_{2,1},\ \mathbf{x}_{2,2},\ \mathbf{y}_{1,1},\ \mathbf{y}_{1,2},\ \mathbf{y}_{2,1},\ \mathbf{y}_{2,2},\ \mathbf{H}_{1,2},\ \mathbf{H}_{2,1})$. Using the method delineated in Section 2, the two-block ADMM iterations can be obtained as

$$\text{(Block 1)}\quad \mathcal{X}^{t+1} = \underset{\mathcal{X}}{\operatorname{argmin}}\ \mathcal{L}_\mu\left(\mathcal{X}, \mathcal{Y}^t, \mathcal{M}^t\right) \tag{13a}$$

$$\text{(Block 2)}\quad \mathcal{Y}^{t+1} = \underset{\mathcal{Y}}{\operatorname{argmin}}\ \mathcal{L}_\mu\left(\mathcal{X}^{t+1}, \mathcal{Y}, \mathcal{M}^t\right) \tag{13b}$$

$$\mathbf{G}_1^{t+1} = \mathbf{G}_1^t + \mu\left(-\overset{t+1}{\mathbf{B}_1^{\text{sum}}} - \overset{t+1}{\mathbf{C}_1^{\text{sum}}} - \overset{t+1}{\mathbf{D}_1^{\text{sum}}} - \overset{t+1}{\mathbf{E}_1^{\text{sum}}} + \mathbf{R}_1^{t+1} - \overset{t+1}{\mathbf{H}_{1,2}^{\text{full}}} - \mathbf{A}_1\right) \tag{13c}$$

$$\mathbf{G}_2^{t+1} = \mathbf{G}_2^t + \mu\left(-\overset{t+1}{\mathbf{B}_2^{\text{sum}}} - \overset{t+1}{\mathbf{C}_2^{\text{sum}}} - \overset{t+1}{\mathbf{D}_2^{\text{sum}}} - \overset{t+1}{\mathbf{E}_2^{\text{sum}}} + \mathbf{R}_2^{t+1} - \overset{t+1}{\mathbf{H}_{2,1}^{\text{full}}} - \mathbf{A}_2\right) \tag{13d}$$

$$\mathbf{G}_{1,2}^{t+1} = \mathbf{G}_{1,2}^t + \mu\left(\mathbf{H}_{1,2}^{t+1} - \mathbf{H}_{(1,2)}^{t+1}\right) \tag{13e}$$

$$\mathbf{G}_{2,1}^{t+1} = \mathbf{G}_{2,1}^t + \mu\left(\mathbf{H}_{2,1}^{t+1} - \mathbf{H}_{(1,2)}^{t+1}\right) \tag{13f}$$

$$\mathbf{f}_{1,1}^{t+1} = \mathbf{f}_{1,1}^t + \mu\left(\mathbf{x}_{1,1}^{t+1} - \mathbf{x}_{(1)}^{t+1}\right) \qquad \mathbf{f}_{2,1}^{t+1} = \mathbf{f}_{2,1}^t + \mu\left(\mathbf{x}_{2,1}^{t+1} - \mathbf{x}_{(1)}^{t+1}\right) \tag{13g}$$

$$\mathbf{f}_{2,2}^{t+1} = \mathbf{f}_{2,2}^t + \mu\left(\mathbf{x}_{2,2}^{t+1} - \mathbf{x}_{(2)}^{t+1}\right) \qquad \mathbf{f}_{1,2}^{t+1} = \mathbf{f}_{1,2}^t + \mu\left(\mathbf{x}_{1,2}^{t+1} - \mathbf{x}_{(2)}^{t+1}\right) \tag{13h}$$

$$\mathbf{g}_{1,1}^{t+1} = \mathbf{g}_{1,1}^t + \mu\left(\mathbf{y}_{1,1}^{t+1} - \mathbf{y}_{(1)}^{t+1}\right) \qquad \mathbf{g}_{2,1}^{t+1} = \mathbf{g}_{2,1}^t + \mu\left(\mathbf{y}_{2,1}^{t+1} - \mathbf{y}_{(1)}^{t+1}\right) \tag{13i}$$

$$\mathbf{g}_{2,2}^{t+1} = \mathbf{g}_{2,2}^t + \mu\left(\mathbf{y}_{2,2}^{t+1} - \mathbf{y}_{(2)}^{t+1}\right) \qquad \mathbf{g}_{1,2}^{t+1} = \mathbf{g}_{1,2}^t + \mu\left(\mathbf{y}_{1,2}^{t+1} - \mathbf{y}_{(2)}^{t+1}\right) \tag{13j}$$

$$\boldsymbol{\lambda}_1^{t+1} = \boldsymbol{\lambda}_1^t + \mu\left(\mathbf{v}_1^{t+1} - \mathbf{z}_1^{t+1}\right) \tag{13k}$$

$$\boldsymbol{\lambda}_2^{t+1} = \boldsymbol{\lambda}_2^t + \mu\left(\mathbf{v}_2^{t+1} - \mathbf{z}_2^{t+1}\right) \tag{13l}$$

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 9 separate optimization subproblems with respect to the individual variables $\mathbf{z}_1$, $\mathbf{z}_2$, $\mathbf{x}_{(1)}$, $\mathbf{x}_{(2)}$, $\mathbf{y}_{(1)}$, $\mathbf{y}_{(2)}$, $\mathbf{R}_1$, $\mathbf{R}_2$, $\mathbf{H}_{(1,2)}$. In addition, the optimization over $\mathcal{Y}$ in Block 2 is equivalent to 2 separate optimization subproblems with the variables $(\mathbf{u}_1, \mathbf{v}_1, \mathbf{x}_{1,1}, \mathbf{x}_{1,2}, \mathbf{y}_{1,1}, \mathbf{y}_{1,2}, \mathbf{H}_{1,2})$ and $(\mathbf{u}_2, \mathbf{v}_2, \mathbf{x}_{2,2}, \mathbf{x}_{2,1}, \mathbf{y}_{2,2}, \mathbf{y}_{2,1}, \mathbf{H}_{2,1})$, 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 (14) and (15) (given in the next page). Note that these agents need to perform local computation in every iteration according to (14) and (15) and then exchange the updated values of the pairs $\left(\mathbf{x}_{(1)}, \mathbf{y}_{(1)}, \mathbf{x}_{1,2}, \mathbf{f}_{1,2}, \mathbf{y}_{1,2}, \mathbf{g}_{1,2}, \mathbf{H}_{1,2}, \mathbf{G}_{1,2}\right)$ and $\left(\mathbf{x}_{(2)}, \mathbf{y}_{(2)}, \mathbf{x}_{2,1}, \mathbf{f}_{2,1}, \mathbf{y}_{2,1}, \mathbf{g}_{2,1}, \mathbf{H}_{2,1}, \mathbf{G}_{2,1}\right)$ with one another.

To elaborate on (14) and (15), the positive semidefinite matrices $\mathbf{R}_1$ and $\mathbf{R}_2$ are updated through the operator $(\cdot)_+$, where $\mathbf{X}_+$ is defined as the projection of an arbitrary symmetric matrix $\mathbf{X}$ onto the set of positive semidefinite matrices by replacing its negative eigenvalues with 0 in the eigenvalue decomposition[29]. The positive vectors $\mathbf{z}_1$, $\mathbf{z}_2$, $\mathbf{y}_{(1)}$ and $\mathbf{y}_{(2)}$ are also updated

Iterations for Agent 1

$$\mathbf{R}_1^{t+1} = \left(\mathbf{B}_1^{\overset{t}{\mathrm{sum}}} + \mathbf{C}_1^{\overset{t}{\mathrm{sum}}} + \mathbf{D}_1^{\overset{t}{\mathrm{sum}}} + \mathbf{E}_1^{\overset{t}{\mathrm{sum}}} + \mathbf{H}_{1,2}^{\overset{t}{\mathrm{full}}} + \mathbf{A}_1 - \frac{\mathbf{G}_1^t}{\mu}\right)_+ \tag{14a}$$

$$\mathbf{z}_1^{t+1} = \left(\mathbf{v}_1^t + \frac{\boldsymbol{\lambda}_1^t}{\mu}\right)_+ \tag{14b}$$

$$\mathbf{x}_{(1)}^{t+1} = \frac{1}{2}\left(\mathbf{x}_{1,1}^t + \mathbf{x}_{2,1}^t + \frac{\mathbf{f}_{1,1}^t + \mathbf{f}_{2,1}^t}{\mu} - \frac{\mathbf{d}_{(1)}}{\mu}\right) \tag{14c}$$

$$\mathbf{y}_{(1)}^{t+1} = \frac{1}{2}\left(\mathbf{y}_{1,1}^t + \mathbf{y}_{2,1}^t + \frac{\mathbf{g}_{1,1}^t + \mathbf{g}_{2,1}^t}{\mu} - \frac{\mathbf{e}_{(1)}}{\mu}\right)_+ \tag{14d}$$

$$\mathbf{H}_{(1,2)}^{t+1} = \frac{1}{2}\left(\mathbf{H}_{1,2}^t + \mathbf{H}_{2,1}^t + \frac{\mathbf{G}_{1,2}^t + \mathbf{G}_{2,1}^t}{\mu}\right) \tag{14e}$$

$$(\mathbf{u}_1,\mathbf{v}_1,\mathbf{x}_{1,1},\mathbf{x}_{1,2},\mathbf{y}_{1,1},\mathbf{y}_{1,2},\mathbf{H}_{1,2})^{t+1} = \mathrm{Lin}\Big(\big(\mathbf{R}_1,\mathbf{z}_1,\mathbf{x}_{(1)},\mathbf{y}_{(1)},\mathbf{H}_{(1,2)}\big)^{t+1} \big(\mathbf{x}_{(2)},\mathbf{y}_{(2)},\mathbf{G}_1,\boldsymbol{\lambda}_1,\mathbf{f}_{1,1},\mathbf{f}_{1,2},\mathbf{g}_{1,1},\mathbf{g}_{1,2},\mathbf{G}_{1,2}\big)^t\Big) \tag{14f}$$

$$\mathbf{G}_1^{t+1} = \mathbf{G}_1^t + \mu\left(-\mathbf{B}_1^{\overset{t+1}{\mathrm{sum}}} - \mathbf{C}_1^{\overset{t+1}{\mathrm{sum}}} - \mathbf{D}_1^{\overset{t+1}{\mathrm{sum}}} - \mathbf{E}_1^{\overset{t+1}{\mathrm{sum}}} + \mathbf{R}_1^{t+1} - \mathbf{H}_{1,2}^{\overset{t+1}{\mathrm{full}}} - \mathbf{A}_1\right) \tag{14g}$$

$$\mathbf{f}_{1,1}^{t+1} = \mathbf{f}_{1,1}^t + \mu\left(\mathbf{x}_{1,1}^{t+1} - \mathbf{x}_{(1)}^{t+1}\right) \qquad \mathbf{f}_{1,2}^{t+1} = \mathbf{f}_{1,2}^t + \mu\left(\mathbf{x}_{1,2}^{t+1} - \mathbf{x}_{(2)}^t\right) \tag{14h}$$

$$\mathbf{g}_{1,1}^{t+1} = \mathbf{g}_{1,1}^t + \mu\left(\mathbf{y}_{1,1}^{t+1} - \mathbf{y}_{(1)}^{t+1}\right) \qquad \mathbf{g}_{1,2}^{t+1} = \mathbf{g}_{1,2}^t + \mu\left(\mathbf{y}_{1,2}^{t+1} - \mathbf{y}_{(2)}^t\right) \tag{14i}$$

$$\mathbf{G}_{1,2}^{t+1} = \mathbf{G}_{1,2}^t + \mu\left(\mathbf{H}_{1,2}^{t+1} - \mathbf{H}_{(1,2)}^{t+1}\right) \tag{14j}$$

$$\boldsymbol{\lambda}_1^{t+1} = \boldsymbol{\lambda}_1^t + \mu\left(\mathbf{v}_1^{t+1} - \mathbf{z}_1^{t+1}\right) \tag{14k}$$

through the operator $(\mathbf{x})_+$, which replaces any negative entry in an arbitrary vector $\mathbf{x}$ with 0 while keeping the nonnegative entries. Using the first-order optimality conditions $\nabla_{\mathbf{x}_{(1)}}\mathcal{L}_\mu(\cdot) = 0$ and $\nabla_{\mathbf{x}_{(2)}}\mathcal{L}_\mu(\cdot) = 0$, one could easily find the closed-form solutions for $\mathbf{x}_{(1)}$ and $\mathbf{x}_{(2)}$ as shown in (14c) and (15c). Similarly, using the optimality condition $\nabla_{\mathbf{H}_{(1,2)}}\mathcal{L}_\mu(\cdot) = 0$, the closed-form solution for $\mathbf{H}^{(1,2)}$ could be found as shown in (14e) and (15e). By combining the conditions $\nabla_{\mathbf{u}_1}\mathcal{L}_\mu(\cdot) = 0$, $\nabla_{\mathbf{v}_1}\mathcal{L}_\mu(\cdot) = 0$, $\nabla_{\mathbf{x}_{1,1}}\mathcal{L}_\mu(\cdot) = 0$, $\nabla_{\mathbf{x}_{1,2}}\mathcal{L}_\mu(\cdot) = 0$, $\nabla_{\mathbf{y}_{1,1}}\mathcal{L}_\mu(\cdot) = 0$, $\nabla_{\mathbf{y}_{1,2}}\mathcal{L}_\mu(\cdot) = 0$ and $\nabla_{\mathbf{H}_{1,2}}\mathcal{L}_\mu(\cdot) = 0$, the updates of ($\mathbf{u}_1$, $\mathbf{v}_1$, $\mathbf{x}_{1,1}$, $\mathbf{x}_{1,2}$, $\mathbf{y}_{1,1}$, $\mathbf{y}_{1,2}$, $\mathbf{H}_{1,2}$) and ($\mathbf{u}_2$, $\mathbf{v}_2$, $\mathbf{x}_{2,2}$, $\mathbf{x}_{2,1}$, $\mathbf{y}_{2,2}$, $\mathbf{y}_{2,1}$, $\mathbf{H}_{2,1}$) reduce to a (not necessarily unique) linear mapping, denoted as $\mathrm{Lin}(\cdot)$ in (14f) and (15f) (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 (14g)-(14k) for agent 1 and in (15g)-(15k) for agent 2.

4.2 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

Iterations for Agent 2

$$\mathbf{R}_2^{t+1} = \left({\mathbf{B}_2^{\mathrm{sum}}}^{t} + {\mathbf{C}_2^{\mathrm{sum}}}^{t} + {\mathbf{D}_2^{\mathrm{sum}}}^{t} + {\mathbf{E}_2^{\mathrm{sum}}}^{t} + {\mathbf{H}_{2,1}^{\mathrm{full}}}^{t} + \mathbf{A}_2 - \frac{\mathbf{G}_2^t}{\mu} \right)_+ \tag{15a}$$

$$\mathbf{z}_2^{t+1} = \left(\mathbf{v}_2^t + \frac{\boldsymbol{\lambda}_2^t}{\mu} \right)_+ \tag{15b}$$

$$\mathbf{x}_{(2)}^{t+1} = \frac{1}{2} \left(\mathbf{x}_{2,2}^t + \mathbf{x}_{1,2}^t + \frac{\mathbf{f}_{2,2}^t + \mathbf{f}_{1,2}^t}{\mu} - \frac{\mathbf{d}_{(2)}}{\mu} \right) \tag{15c}$$

$$\mathbf{y}_{(2)}^{t+1} = \frac{1}{2} \left(\mathbf{y}_{2,2}^t + \mathbf{y}_{1,2}^t + \frac{\mathbf{g}_{2,2}^t + \mathbf{g}_{1,2}^t}{\mu} - \frac{\mathbf{e}_{(2)}}{\mu} \right)_+ \tag{15d}$$

$$\mathbf{H}_{(1,2)}^{t+1} = \frac{1}{2} \left(\mathbf{H}_{1,2}^t + \mathbf{H}_{2,1}^t + \frac{\mathbf{G}_{1,2}^t + \mathbf{G}_{2,1}^t}{\mu} \right) \tag{15e}$$

$$(\mathbf{u}_2,\mathbf{v}_2,\mathbf{x}_{2,2},\mathbf{x}_{2,1},\mathbf{y}_{2,2},\mathbf{y}_{2,1},\mathbf{H}_{2,1})^{t+1} = \mathrm{Lin}\Big((\mathbf{R}_2, \mathbf{z}_2, \mathbf{x}_{(2)}, \mathbf{y}_{(2)}, \mathbf{H}_{(1,2)})^{t+1} \\ (\mathbf{x}_{(1)}, \mathbf{y}_{(1)}, \mathbf{G}_2, \boldsymbol{\lambda}_2, \mathbf{f}_{2,2}, \mathbf{f}_{2,1}, \mathbf{g}_{2,2}, \mathbf{g}_{2,1}, \mathbf{G}_{2,1})^t \Big) \tag{15f}$$

$$\mathbf{G}_2^{t+1} = \mathbf{G}_2^t + \mu \left(-{\mathbf{B}_2^{\mathrm{sum}}}^{t+1} - {\mathbf{C}_2^{\mathrm{sum}}}^{t+1} - {\mathbf{D}_2^{\mathrm{sum}}}^{t+1} - {\mathbf{E}_2^{\mathrm{sum}}}^{t+1} + \mathbf{R}_2^{t+1} - {\mathbf{H}_{2,1}^{\mathrm{full}}}^{t+1} - \mathbf{A}_2 \right) \tag{15g}$$

$$\mathbf{f}_{2,2}^{t+1} = \mathbf{f}_{2,2}^t + \mu \left(\mathbf{x}_{2,2}^{t+1} - \mathbf{x}_{(2)}^{t+1} \right) \qquad \mathbf{f}_{2,1}^{t+1} = \mathbf{f}_{2,1}^t + \mu \left(\mathbf{x}_{2,1}^{t+1} - \mathbf{x}_{(1)}^t \right) \tag{15h}$$

$$\mathbf{g}_{2,2}^{t+1} = \mathbf{g}_{2,2}^t + \mu \left(\mathbf{y}_{2,2}^{t+1} - \mathbf{y}_{(2)}^{t+1} \right) \qquad \mathbf{g}_{2,1}^{t+1} = \mathbf{g}_{2,1}^t + \mu \left(\mathbf{y}_{2,1}^{t+1} - \mathbf{y}_{(1)}^t \right) \tag{15i}$$

$$\mathbf{G}_{2,1}^{t+1} = \mathbf{G}_{2,1}^t + \mu \left(\mathbf{H}_{2,1}^{t+1} - \mathbf{H}_{(1,2)}^{t+1} \right) \tag{15j}$$

$$\boldsymbol{\lambda}_2^{t+1} = \boldsymbol{\lambda}_2^t + \mu \left(\mathbf{v}_2^{t+1} - \mathbf{z}_2^{t+1} \right) \tag{15k}$$

(10), can be expressed in the decomposable form:

$$\min \sum_{i \in \mathcal{V}} \Big(\mathbf{b}_i^T \mathbf{u}_i + \mathbf{c}_i^T \mathbf{v}_i + \mathbf{d}_{(i)}^T \mathbf{x}_{(i)} + \mathbf{e}_{(i)}^T \mathbf{y}_{(i)} + I_+(\mathbf{z}_i) + I_+(\mathbf{R}_i) + I_+(\mathbf{y}_{(i)}) \Big) \tag{16a}$$

subject to

$$-\mathbf{B}_i^{\mathrm{sum}} - \mathbf{C}_i^{\mathrm{sum}} - \mathbf{D}_i^{\mathrm{sum}} - \mathbf{E}_i^{\mathrm{sum}} + \mathbf{R}_i - \mathbf{H}_i^{\mathrm{sum}} = \mathbf{A}_i \qquad \forall\, i \in \mathcal{V} \tag{16b}$$

$$\mathbf{x}_{i,k} = \mathbf{x}_{(k)} \qquad \forall\, k \in N[i] \quad \text{and} \quad i \in \mathcal{V} \tag{16c}$$

$$\mathbf{y}_{i,k} = \mathbf{y}_{(k)} \qquad \forall\, k \in N[i] \quad \text{and} \quad i \in \mathcal{V} \tag{16d}$$

$$\mathbf{H}_{i,k} = \mathbf{H}_{(i,k)_\preceq} \qquad \forall\, k \in N(i) \quad \text{and} \quad i \in \mathcal{V} \tag{16e}$$

$$\mathbf{v}_i = \mathbf{z}_i \qquad \forall\, i \in \mathcal{V} \tag{16f}$$

with the variables $\Big(\mathbf{u}_i, \mathbf{v}_i, \mathbf{z}_i, \mathbf{x}_{(i)}, \mathbf{y}_{(i)}, \mathbf{R}_i, \{\mathbf{x}_{i,k}, \mathbf{y}_{i,k}\}_{k \in N[i]}, \{\mathbf{H}_{i,k}, \mathbf{H}_{(i,k)_\preceq}\}_{k \in N(i)}\Big)$ for every $i \in \mathcal{V}$, where $\mathbf{B}_i^{\mathrm{sum}} = \sum_{j=1}^{p_i} u_j^i \mathbf{B}_j^i$, $\mathbf{C}_i^{\mathrm{sum}} = \sum_{j=1}^{q_i} v_j^i \mathbf{C}_j^i$, $\mathbf{D}_i^{\mathrm{sum}} = \sum_{k \in N[i]} \sum_{j=1}^{r_k} x_j^{i,k} \mathbf{D}_j^{k,i}$, $\mathbf{E}_i^{\mathrm{sum}} = \sum_{k \in N[i]} \sum_{j=1}^{s_k} y_j^{i,k} \mathbf{E}_j^{k,i}$ and $\mathbf{H}_i^{\mathrm{sum}} = \sum_{k \in N(i)} \mathbf{H}_{i,k}^{\mathrm{full}}$. Note that $\mathbf{u}_i \in \mathbb{R}^{p_i}$, $\mathbf{v}_i \in \mathbb{R}^{q_i}$, $\mathbf{x}_{(i)} \in \mathbb{R}^{r_i}$ and $\mathbf{y}_{(i)} \in \mathbb{R}^{s_i}$ are the Lagrange multipliers corresponding to the constraints (7b), (7c), (7e) and (7f) respectively, and that $\mathbf{R}_i \in \mathbb{S}^{n_i}$ is the Lagrange multiplier corresponding to the constraint (7d). Each element $h_{i,k}^{\mathrm{full}}(a,b)$ of $\mathbf{H}_{i,k}^{\mathrm{full}}$ is either zero or equal to the Lagrange

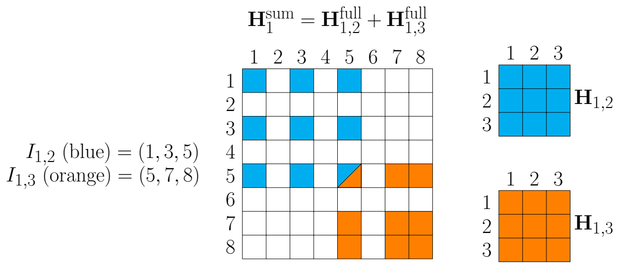

$\mathbf{H}_1^{\mathrm{sum}} = \mathbf{H}_{1,2}^{\mathrm{full}} + \mathbf{H}_{1,3}^{\mathrm{full}}$
$I_{1,2}$ (blue) $= (1,3,5)$
$I_{1,3}$ (orange) $= (5,7,8)$
$\mathbf{H}_{1,2}$
$\mathbf{H}_{1,3}$

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

multiplier corresponding to an overlapping element $W_i(a,b)$ between $\mathbf{W}_i$ and $\mathbf{W}_k$. For a better understanding of the difference between $\mathbf{H}_{i,j}^{\mathrm{full}}$, $\mathbf{H}_{i,j}$ and $\mathbf{H}_i^{\mathrm{sum}}$, 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 (17) for each agent $i \in \mathcal{V}$.

Consider the parameters defined in (19) for every $i \in \mathcal{V}$ and time $t \in \{1,2,3,....\}$. Define V^t as

$$V^t = \sum_{i\in\mathcal{V}} \Big(\left(\Delta_{p1}^t\right)_i + \left(\Delta_{p5}^t\right)_i + \left(\Delta_{d1}^t\right)_i + \left(\Delta_{d2}^t\right)_i + \left(\Delta_{d3}^t\right)_i + \left(\Delta_{d4}^t\right)_i \Big) \\ + \sum_{i\in\mathcal{V}} \Bigg(\sum_{k\in N[i]} \Big(\left(\Delta_{p2}^t\right)_{i,k} + \left(\Delta_{p3}^t\right)_{i,k} \Big) + \sum_{k\in N(i)} \Big(\left(\Delta_{p4}^t\right)_{i,k} + \left(\Delta_{d5}^t\right)_{i,k} \Big) \Bigg) \tag{18}$$

Note that $(\boldsymbol{\Delta}_{p1}, \boldsymbol{\Delta}_{p2}, \boldsymbol{\Delta}_{p3}, \boldsymbol{\Delta}_{p4}, \boldsymbol{\Delta}_{p5})$, $(\boldsymbol{\Delta}_{d1}, \boldsymbol{\Delta}_{d2}, \boldsymbol{\Delta}_{d3}, \boldsymbol{\Delta}_{d4}, \boldsymbol{\Delta}_{d5})$, and $\mathbf{V}$ are the primal residues, dual residues and aggregate residue for the decomposed problem (16). It should be noticed that the dual residues are only considered for the variables in the block $\mathcal{X} = \Big(\mathbf{z}_i, \mathbf{x}_{(i)}, \mathbf{y}_{(i)}, \mathbf{R}_i, \{\mathbf{H}_{(i,k)_{\preceq}}\}_{k\in N(i)}\Big)_{i\in\mathcal{V}}$. Since $\mathbf{x}_{(i)}$ and $\mathbf{y}_{(i)}$ appear $|N[i]|$ times in (16), the norm in the residues Δ_{d2} and Δ_{d3}, respectively, are multiplied by $|N[i]|$. If the residue Δ_{d5} is considered for all $i \in \mathcal{V}$, then the expression in (19j) is equivalent to $(\Delta_{d5}^t)_{i,j} = 2 \times \left\|\mathbf{H}_{(i,j)}^t - \mathbf{H}_{(i,j)}^{t-1}\right\|_F^2$ for all $(i,j) \in \mathcal{E}^+$ since $\mathbf{H}_{(i,j)}$ appears twice in (16). 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* (17)*. The following statements hold:*

– *The aggregate residue V^t attenuates to 0 in a non-increasing way as t goes to $+\infty$.*

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

$$\mathbf{R}_i^{t+1} = \left(\overset{t}{\mathbf{B}_i^{\text{sum}}} + \overset{t}{\mathbf{C}_i^{\text{sum}}} + \overset{t}{\mathbf{D}_i^{\text{sum}}} + \overset{t}{\mathbf{E}_i^{\text{sum}}} + \overset{t}{\mathbf{H}_i^{\text{sum}}} + \mathbf{A}_i - \frac{\mathbf{G}_i^t}{\mu}\right)_+ \tag{17a}$$

$$\mathbf{z}_i^{t+1} = \left(\mathbf{v}_i^t + \frac{\boldsymbol{\lambda}_i^t}{\mu}\right)_+ \tag{17b}$$

$$\mathbf{x}_{(i)}^{t+1} = \frac{1}{|N[i]|}\left(\sum_{j \in N[i]}\left(\mathbf{x}_{j,i}^t + \frac{\mathbf{f}_{j,i}^t}{\mu}\right) - \frac{\mathbf{d}_{(i)}}{\mu}\right) \tag{17c}$$

$$\mathbf{y}_{(i)}^{t+1} = \frac{1}{|N[i]|}\left(\sum_{j \in N[i]}\left(\mathbf{y}_{j,i}^t + \frac{\mathbf{g}_{j,i}^t}{\mu}\right) - \frac{\mathbf{e}_{(i)}}{\mu}\right)_+ \tag{17d}$$

$$\mathbf{H}_{(i,k)_{\preceq}}^{t+1} = \frac{1}{2}\left(\mathbf{H}_{i,k}^t + \mathbf{H}_{k,i}^t + \frac{\mathbf{G}_{i,k}^t + \mathbf{G}_{k,i}^t}{\mu}\right) \qquad \forall k \in N(i) \tag{17e}$$

$$\left(\mathbf{u}_i, \mathbf{v}_i, \left\{\mathbf{x}_{i,k}, \mathbf{y}_{i,k}\right\}_{k \in N[i]}, \left\{\mathbf{H}_{i,k}\right\}_{k \in N(i)}\right)^{t+1} = \text{Lin}\left(\left(\mathbf{R}_i, \mathbf{z}_i, \mathbf{x}_{(i)}, \mathbf{y}_{(i)}, \left\{\mathbf{H}_{i,k}\right\}_{k \in N(i)}\right)^{t+1}, \left(\mathbf{G}_i, \boldsymbol{\lambda}_i, \left\{\, \mathbf{f}_{i,k}, \mathbf{g}_{i,k}\right\}_{k \in N[i]}, \left\{\, \mathbf{x}_{(k)}, \mathbf{y}_{(k)}, \mathbf{G}_{i,k}\right\}_{k \in N(i)}\right)^t\right) \tag{17f}$$

$$\mathbf{G}_i^{t+1} = \mathbf{G}_i^t + \mu\left(-\overset{t+1}{\mathbf{B}_i^{\text{sum}}} - \overset{t+1}{\mathbf{C}_i^{\text{sum}}} - \overset{t+1}{\mathbf{D}_i^{\text{sum}}} - \overset{t+1}{\mathbf{E}_i^{\text{sum}}} + \mathbf{R}_i^{t+1} - \overset{t+1}{\mathbf{H}_i^{\text{sum}}} - \mathbf{A}_i\right) \tag{17g}$$

$$\mathbf{f}_{i,k}^{t+1} = \mathbf{f}_{i,k}^t + \mu\left(\mathbf{x}_{i,k}^{t+1} - \mathbf{x}_{(k)}^{\hat{t}}\right) \qquad \forall k \in N[i] \ \ \left(\text{where } \hat{t} = t+1 \text{ if } i = k, \ \text{else } \hat{t} = t\right) \tag{17h}$$

$$\mathbf{g}_{i,k}^{t+1} = \mathbf{g}_{i,k}^t + \mu\left(\mathbf{y}_{i,k}^{t+1} - \mathbf{y}_{(k)}^{\hat{t}}\right) \qquad \forall k \in N[i] \ \ \left(\text{where } \hat{t} = t+1 \text{ if } i = k, \ \text{else } \hat{t} = t\right) \tag{17i}$$

$$\mathbf{G}_{i,k}^{t+1} = \mathbf{G}_{i,k}^t + \mu\left(\mathbf{H}_{i,k}^{t+1} - \mathbf{H}_{(i,k)_{\preceq}}^{t+1}\right) \qquad \forall k \in N(i) \tag{17j}$$

$$\boldsymbol{\lambda}_i^{t+1} = \boldsymbol{\lambda}_i^t + \mu\left(\mathbf{v}_i^{t+1} - \mathbf{z}_i^{t+1}\right) \tag{17k}$$

Aggregate residue parameters

$$\left(\Delta_{p1}^t\right)_i = \left\|\overset{t}{\mathbf{B}_i^{\text{sum}}} + \overset{t}{\mathbf{C}_i^{\text{sum}}} + \overset{t}{\mathbf{D}_i^{\text{sum}}} + \overset{t}{\mathbf{E}_i^{\text{sum}}} + \overset{t}{\mathbf{H}_i^{\text{sum}}} + \mathbf{A}_i - \mathbf{R}_i^t\right\|_F^2 \tag{19a}$$

$$\left(\Delta_{p2}^t\right)_{i,k} = \left\|\mathbf{x}_{i,k}^t - \mathbf{x}_{(k)}^t\right\|_2^2 \qquad \forall k \in N[i] \tag{19b}$$

$$\left(\Delta_{p3}^t\right)_{i,k} = \left\|\mathbf{y}_{i,k}^t - \mathbf{y}_{(k)}^t\right\|_2^2 \qquad \forall k \in N[i] \tag{19c}$$

$$\left(\Delta_{p4}^t\right)_{i,k} = \left\|\mathbf{H}_{i,k}^t - \mathbf{H}_{(i,k)_{\preceq}}^t\right\|_F^2 \qquad \forall k \in N(i) \tag{19d}$$

$$\left(\Delta_{p5}^t\right)_i = \left\|\mathbf{v}_i^t - \mathbf{z}_i^t\right\|_2^2 \tag{19e}$$

$$\left(\Delta_{d1}^t\right)_i = \left\|\mathbf{z}_i^t - \mathbf{z}_i^{t-1}\right\|_2^2 \tag{19f}$$

$$\left(\Delta_{d2}^t\right)_i = |N[i]| \times \left\|\mathbf{x}_{(i)}^t - \mathbf{x}_{(i)}^{t-1}\right\|_2^2 \tag{19g}$$

$$\left(\Delta_{d3}^t\right)_i = |N[i]| \times \left\|\mathbf{y}_{(i)}^t - \mathbf{y}_{(i)}^{t-1}\right\|_2^2 \tag{19h}$$

$$\left(\Delta_{d4}^t\right)_i = \left\|\mathbf{R}_i^t - \mathbf{R}_i^{t-1}\right\|_F^2 \tag{19i}$$

$$\left(\Delta_{d5}^t\right)_{i,k} = \left\|\mathbf{H}_{(i,k)_{\preceq}}^t - \mathbf{H}_{(i,k)_{\preceq}}^{t-1}\right\|_F^2 \qquad \forall k \in N(i) \tag{19j}$$

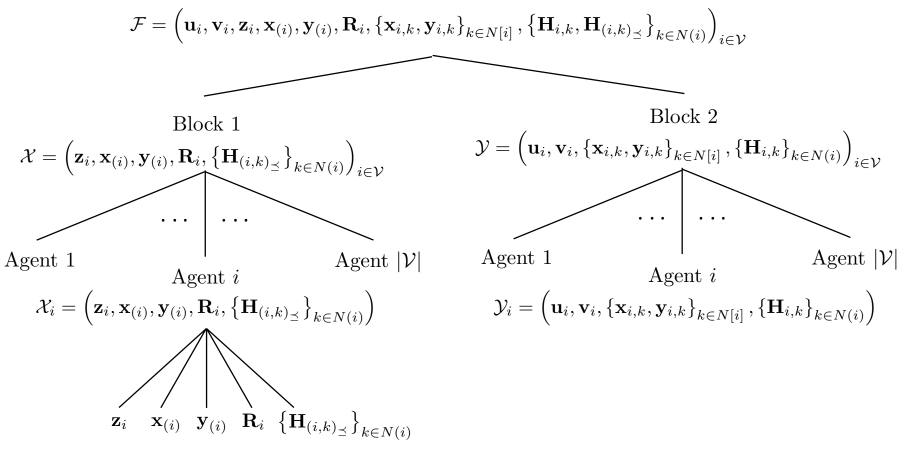

$\mathcal{F} = \Big(\mathbf{u}_i, \mathbf{v}_i, \mathbf{z}_i, \mathbf{x}_{(i)}, \mathbf{y}_{(i)}, \mathbf{R}_i, \{\mathbf{x}_{i,k}, \mathbf{y}_{i,k}\}_{k\in N[i]}, \{\mathbf{H}_{i,k}, \mathbf{H}_{(i,k)_\preceq}\}_{k\in N(i)}\Big)_{i\in\mathcal{V}}$
Block 1
$\mathcal{X} = \Big(\mathbf{z}_i, \mathbf{x}_{(i)}, \mathbf{y}_{(i)}, \mathbf{R}_i, \{\mathbf{H}_{(i,k)_\preceq}\}_{k\in N(i)}\Big)_{i\in\mathcal{V}}$
Block 2
$\mathcal{Y} = \Big(\mathbf{u}_i, \mathbf{v}_i, \{\mathbf{x}_{i,k}, \mathbf{y}_{i,k}\}_{k\in N[i]}, \{\mathbf{H}_{i,k}\}_{k\in N(i)}\Big)_{i\in\mathcal{V}}$
Agent 1
Agent i
Agent $|\mathcal{V}|$
Agent 1
Agent i
Agent $|\mathcal{V}|$
$\mathcal{X}_i = \Big(\mathbf{z}_i, \mathbf{x}_{(i)}, \mathbf{y}_{(i)}, \mathbf{R}_i, \{\mathbf{H}_{(i,k)_\preceq}\}_{k\in N(i)}\Big)$
$\mathcal{Y}_i = \Big(\mathbf{u}_i, \mathbf{v}_i, \{\mathbf{x}_{i,k}, \mathbf{y}_{i,k}\}_{k\in N[i]}, \{\mathbf{H}_{i,k}\}_{k\in N(i)}\Big)$
$\mathbf{z}_i$
$\mathbf{x}_{(i)}$
$\mathbf{y}_{(i)}$
$\mathbf{R}_i$
$\{\mathbf{H}_{(i,k)_\preceq}\}_{k\in N(i)}$

Fig. 5: This diagram shows the subproblems resulted from the two-block ADMM procedure. The optimization variables set $\mathcal{F}$ is split into two blocks of variables $\mathcal{X}$ and $\mathcal{Y}$ such that the optimization over $\mathcal{X}$ is fully decomposable into separate subproblems with respect to the individual variables $\mathbf{z}_i, \mathbf{x}_{(i)}, \mathbf{y}_{(i)}, \mathbf{R}_i, \{\mathbf{H}_{(i,k)_\preceq}\}_{k\in N(i)}$ for all $i \in \mathcal{V}$ while the optimization over $\mathcal{Y}$ is equivalent to $|\mathcal{V}|$ separate optimization subproblems.

– *For every $i \in \mathcal{V}$, the limit of $(\mathbf{G}_1^t, \mathbf{G}_2^t, ..., \mathbf{G}_n^t)$ at $t = +\infty$ is an optimal solution for $(\mathbf{W}_1, \mathbf{W}_2, ..., \mathbf{W}_n)$.*

Proof After realizing that (17) is obtained from a two-block ADMM procedure, as shown in Figure 5, the theorem follows from [14] 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 [19], we terminate the algorithm as soon as $\max\{\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3, \mathbf{D}_1, \mathbf{D}_2, \mathbf{D}_3, \mathbf{D}_4, \mathbf{D}_5, \text{Gap}\}$ becomes smaller than a pre-specified tolerance, where

$$(\mathrm{P}_1)_i = \frac{\left\|\overline{\mathbf{B}}_i^T\overline{\mathbf{W}}_i - \mathbf{b}_i\right\|_2 + \left\|\max\left(\overline{\mathbf{C}}_i^T\overline{\mathbf{W}}_i - \mathbf{c}_i, \mathbf{0}\right)\right\|_2}{1 + \|\mathbf{c}_i\|_2} \tag{20a}$$

$$(\mathrm{P}_2)_i = \frac{\left\|\sum_{j\in N[i]}\overline{\mathbf{D}}_{i,j}^T\overline{\mathbf{W}}_j - \mathbf{d}_{(i)}\right\|_2 + \left\|\max\left(\sum_{j\in N[i]}\overline{\mathbf{E}}_{i,j}^T\overline{\mathbf{W}}_j - \mathbf{e}_{(i)}, \mathbf{0}\right)\right\|_2}{1 + \left\|\mathbf{d}_{(i)}\right\|_2} \tag{20b}$$

$$(\mathrm{P}_3)_{i,j} = \frac{\|\mathbf{W}_i(I_{i,j}, I_{i,j}) - \mathbf{W}_j(I_{j,i}, I_{j,i})\|_F}{1 + \|\mathbf{W}_i(I_{i,j}, I_{i,j})\|_F + \|\mathbf{W}_j(I_{j,i}, I_{j,i})\|_F} \tag{20c}$$

$$(\mathrm{D}_1)_i = \frac{\|-\mathbf{B}_i^{\mathrm{sum}} - \mathbf{C}_i^{\mathrm{sum}} - \mathbf{D}_i^{\mathrm{sum}} - \mathbf{E}_i^{\mathrm{sum}} + \mathbf{R}_i - \mathbf{H}_i^{\mathrm{sum}} - \mathbf{A}_i\|_F}{1 + \|\mathbf{A}_i\|_1} \tag{20d}$$

$$(\mathrm{D}_2)_{i,k} = \frac{\left\|\mathbf{x}_{i,k} - \mathbf{x}_{(k)}\right\|_2}{1 + \|\mathbf{x}_{i,k}\|_2 + \left\|\mathbf{x}_{(k)}\right\|_2} \qquad \forall k \in N[i] \tag{20e}$$

$$(\mathrm{D}_3)_{i,k} = \frac{\left\|\mathbf{y}_{i,k} - \mathbf{y}_{(k)}\right\|_2}{1 + \|\mathbf{y}_{i,k}\|_2 + \left\|\mathbf{y}_{(k)}\right\|_2} \qquad \forall k \in N[i] \tag{20f}$$

$$(\mathrm{D}_4)_{i,k} = \frac{\left\|\mathbf{H}_{i,k} - \mathbf{H}_{(i,k)_\preceq}\right\|_F}{1 + \|\mathbf{H}_{i,k}\|_F + \left\|\mathbf{H}_{(i,k)_\preceq}\right\|_F} \qquad \forall k \in N(i) \tag{20g}$$

$$(\mathrm{D}_5)_i = \frac{\|\mathbf{v}_i - \mathbf{z}_i\|_2}{1 + \|\mathbf{v}_i\|_2 + \|\mathbf{z}_i\|_2} \tag{20h}$$

$$\mathrm{Gap} = \frac{\left|\sum_{i\in\mathcal{V}}\left(\mathbf{b}_i^T\mathbf{u}_i + \mathbf{c}_i^T\mathbf{v}_i + \mathbf{d}_{(i)}^T\mathbf{x}_{(i)} + \mathbf{e}_{(i)}^T\mathbf{y}_{(i)} - \mathbf{tr}\left(\mathbf{A}_i\mathbf{W}_i\right)\right)\right|}{1 + \left|\sum_{i\in\mathcal{V}}\left(\mathbf{b}_i^T\mathbf{u}_i + \mathbf{c}_i^T\mathbf{v}_i + \mathbf{d}_{(i)}^T\mathbf{x}_{(i)} + \mathbf{e}_{(i)}^T\mathbf{y}_{(i)}\right)\right| + \left|\sum_{i\in\mathcal{V}}\mathbf{tr}\left(A_iW_i\right)\right|} \tag{20i}$$

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.
- $\overline{\mathbf{W}}_i$ is the vectorized version of $\mathbf{W}_i$ obtained by stacking the columns of $\mathbf{W}_i$ one under another to create a column vector.
- $\overline{\mathbf{B}}_i$ and $\overline{\mathbf{C}}_i$ are matrices whose columns are the vectorized versions of $\mathbf{B}_j^i$ and $\mathbf{C}_j^i$ for $j = 1, \ldots, p_i$ and $j = 1, \ldots, q_i$, respectively.
- $\overline{\mathbf{D}}_{i,j}$ and $\overline{\mathbf{E}}_{i,j}$ are matrices whose columns are the vectorized versions of $\mathbf{D}_k^{i,j}$ and $\mathbf{E}_k^{i,j}$ for $k = 1, \ldots, r_i$ and $k = 1, \ldots, s_i$, respectively.

The stopping criteria in (20) are based on the primal and dual infeasibilities as well as the duality gap.

5 Simulations Results

The objective of this section is to elucidate the results developed earlier on randomly generated large-scale structured SDP problems. The algorithm was implemented in a C++ code. Unless otherwise stated, the simulations were

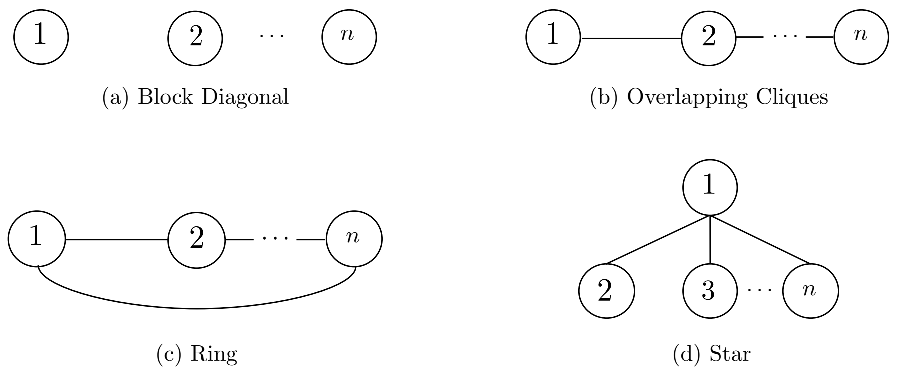


Fig. 6: The structures that are used for the simulations

run on a laptop with an Intel Core i7 quad-core 2.5 GHz CPU and 16 GB RAM.

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

- Each matrix $\mathbf{A}_i$ is chosen as $\mathbf{\Omega} + \mathbf{\Omega}^T + n_i\mathbf{I}$, where the entries of $\mathbf{\Omega}$ are uniformly chosen from the integer set $\{1, 2, 3, 4, 5\}$.
- Each matrix $\mathbf{B}_j$ (or $\mathbf{C}_j$) is chosen as $\mathbf{\Omega}+\mathbf{\Omega}^T$, where $\mathbf{\Omega}$ is generated as before. In all simulations, the coupling constraints were ignored, $r_i = s_i = 0$, and so the matrices $\mathbf{D}_{i,j}$ and $\mathbf{E}_{i,j}$ were not created.
- Each matrix variable $\mathbf{W}_i$ is assumed to be 40 by 40.
- For every two overlapping matrices W_i and W_j, only 25% of their entries overlap, leading to a 10×10 submatrix.

In order to show how general the algorithm is in handling different structures, the simulations were run over the following four overlapping structures shown in Figure 6:

- *Block Diagonal*: this is represented by the edgeless graph $\mathcal{V} = \{1, \cdots, n\}$ and $|\mathcal{E}| = \phi$, as shown in Figure 6(a). This is equivalent to solving for n independent centralized SDPs for the matrices $\mathbf{W}_1, ..., \mathbf{W}_n$ without any overlap among them.
- *Overlapping Cliques*: this is represented by the path graph $\mathcal{V} = \{1, \cdots, n\}$ and $\mathcal{E} = \{(1, 2),\ (2, 3),\ \cdots,\ (n-1, n)\}$, as shown in Figure 6(b). This is equivalent to $\mathbf{W}_i$'s that are submatrices of a full-scale matrix variable $\mathbf{W}$ in the form of Figure 3 but with n overlapping blocks.
- *Ring*: this is similar to the overlapping cliques structure only with the addition of an overlap between $\mathbf{W}_1$ and $\mathbf{W}_n$ that creates a loop, as shown in Figure 6(c).
- *Star*: this is represented by a graph in which a central node is connected to all other nodes, as shown in Figure 6(d).

In order to demonstrate the proposed algorithm on large-scale SDPs, three different values will be considered for the total number of blocks (or agents):

1000, 2000 and 4000 in case of the block diagonal, overlapping cliques and ring structures and 1000, 2000 and 2500 agents in case of the star structure. To give the reader a sense of how large the simulated SDPs are, the total number of entries of $\mathbf{W}_i$'s in the decomposed SDP problem (N_{Decomp}) and the total number of entries of $\mathbf{W}$ 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
– 2500 agents: $N_{\text{Full}} = 5.6$ billion, $N_{\text{Decomp}} = 4$ million
– 4000 agents: $N_{\text{Full}} = 14.4$ billion, $N_{\text{Decomp}} = 6.4$ million

The simulation results for each of the structures in Figure 6 are provided in Table 1, Table 2, Table 3 and Table 4, respectively, 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 minutes) 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 the tables, the simulations were run for three cases (except for the star case that was not run for the case of $p_i = 5$ and $q_i = 5$):

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

All solutions reported in the tables are based on the tolerance of 10^{-3} and an optimality degree of at least 99%. Note that the time per iteration is between 0.117 and 1.398 in a C++ implementation. Efficient and computationally cheap preconditioning methods could dramatically reduce the number of iterations, but this is outside the scope of this work.

In Table 5, a problem with 8000 overlapping cliques is simulated using an Amazon EC2 instance with 36 cores and 60 GB RAM. This problem corresponds to $N_{\text{Full}} = 57.6$ billion and $N_{\text{Decomp}} = 12.8$ million. For the case $p_i = 5$ and $q_i = 5$, the final result was found in 19.539 minutes with an optimality degree of 99.9997%.

The aggregative residue V^t for all of the previous simulations are plotted in Figure 7, Figure 8, Figure 9, Figure 10 and Figure 11. The aggregative residue is a monotonically decreasing function for all of the cases.

In order to evaluate the speedup gained from increasing the number of cores, the same instance of the random SDP with 4000 overlapping cliques and $p_i = 5$ and $q_i = 5$ (shown in Table 2) was run on Amazon EC2. While the result was found in 31.404 minutes on a laptop with 4 cores, it was found in

Cases		1000	2000	4000
$p_i = 5$ $q_i = 0$	P_{obj}	4.456450e+05	8.865502e+05	1.772719e+06
	D_{obj}	4.456330e+05	8.865390e+05	1.772702e+06
	iter	365	435	512
	t_{CPU} (min)	0.714	1.741	3.980
	t_{iter} (sec per iter)	0.117	0.240	0.466
	Optimality	99.997%	99.998%	99.99904%
$p_i = 0$ $q_i = 5$	P_{obj}	6.755040e+05	1.341025e+06	2.697884e+06
	D_{obj}	6.755032e+05	1.341018e+06	2.697884e+06
	iter	2365	3426	5640
	t_{CPU} (min)	4.720	14.008	48.204
	t_{iter} (sec per iter)	0.119	0.245	0.513
	Optimality	99.9998%	99.9994%	99.99999%
$p_i = 5$ $q_i = 5$	P_{obj}	1.002835e+06	1.990340e+06	3.985788e+06
	D_{obj}	1.002835e+06	1.990339e+06	3.985784e+06
	iter	2614	2476	2457
	t_{CPU} (min)	5.662	668.84	22.136
	t_{iter} (sec per iter)	0.130	0.270	0.541
	Optimality	99.99999%	99.99994%	99.9998%

Table 1: Simulation results for the block diagonal structure for three cases with 1000, 2000 and 4000 agents.

Cases		1000	2000	4000
$p_i = 5$ $q_i = 0$	P_{obj}	4.931567e+05	9.826826e+05	1.965131e+06
	D_{obj}	4.930689e+05	9.825347e+05	1.964877e+06
	iter	250	276	295
	t_{CPU} (min)	0.752	1.705	3.693
	t_{iter} (sec per iter)	0.180	0.371	0.751
	Optimality	99.98%	99.98%	99.98%
$p_i = 0$ $q_i = 5$	P_{obj}	8.149802e+05	1.610153e+06	3.247932e+06
	D_{obj}	8.149710e+05	1.610142e+06	3.247909e+06
	iter	1013	1070	1289
	t_{CPU} (min)	3.177	7.047	18.397
	t_{iter} (sec per iter)	0.188	0.395	0.856
	Optimality	99.998%	99.9993%	99.9992%
$p_i = 5$ $q_i = 5$	P_{obj}	1.193110e+06	2.366842e+06	4.744581e+06
	D_{obj}	1.193106e+06	2.366835e+06	4.744566e+06
	iter	2202	2364	2353
	t_{CPU} (min)	7.702	15.738	31.404
	t_{iter} (sec per iter)	0.210	0.399	0.801
	Optimality	99.9996%	99.9997%	99.9996%

Table 2: Simulation results for the overlapping cliques structure for three cases with 1000, 2000 and 4000 agents.

8.09 minutes using Amazon EC2 instance with 36 cores, accounting for a 3.88 folds speedup.

6 Distribution of Computational Load

The main objective of this section is to study some challenges that are expected in the presence of agents that are overlapping with many other agents (i.e., high-degree nodes). The following are the challenges, which are also true for the case of the star structure simulated in Section 5:

Cases		1000	2000	4000
$p_i = 5$ $q_i = 0$	P_{obj}	4.932126e+05	9.827618e+05	1.965181e+06
	D_{obj}	4.931227e+05	9.826132e+05	1.964928e+06
	iter	250	276	295
	t_{CPU} (min)	0.778	1.732	3.759
	t_{iter} (sec per iter)	0.187	0.376	0.765
	Optimality	99.98%	99.98%	99.98%
$p_i = 0$ $q_i = 5$	P_{obj}	8.151984e+05	1.610285e+06	3.247959e+06
	D_{obj}	8.151880e+05	1.610273e+06	3.247935e+06
	iter	1013	1070	1289
	t_{CPU} (min)	3.192	6.963	20.197
	t_{iter} (sec per iter)	0.189	0.390	0.940
	Optimality	99.998%	99.9992%	99.9992%
$p_i = 5$ $q_i = 5$	P_{obj}	1.193391e+06	2.367079e+06	4.744821e+06
	D_{obj}	1.193387e+06	2.367072e+06	4.744806e+06
	iter	2202	2364	2353
	t_{CPU} (min)	7.551	18.378	37.093
	t_{iter} (sec per iter)	0.206	0.466	0.946
	Optimality	99.9996%	99.9997%	99.9996%

Table 3: Simulation results for the ring structure for three cases with 1000, 2000 and 4000 agents.

Cases		1000	2000	2500
$p_i = 5$ $q_i = 0$	P_{obj}	4.991551e+05	9.948218e+05	1.240980e+06
	D_{obj}	4.981595e+05	9.928398e+05	1.239480e+06
	iter	760	726	810
	t_{CPU} (min)	4.645	11.961	18.872
	t_{iter} (sec per iter)	0.367	0.988	1.398
	Optimality	99.8%	99.8%	99.8%
$p_i = 0$ $q_i = 5$	P_{obj}	8.333963e+05	1.653198e+06	2.070485e+06
	D_{obj}	8.333382e+05	1.653117e+06	2.070437e+06
	iter	1570	2063	2405
	t_{CPU} (min)	9.475	33.817	55.919
	t_{iter} (sec per iter)	0.362	0.984	1.395
	Optimality	99.993%	99.995%	99.997%

Table 4: Simulation results for the star structure for three cases with 1000, 2000 and 2500 agents.

	$p_i = 5, q_i = 0$	$p_i = 0, q_i = 5$	$p_i = 5, q_i = 5$
P_{obj}	3.939822e+06	6.475070e+06	9.458764e+06
D_{obj}	3.939368e+06	6.475035e+06	9.458743e+06
iter	325	1264	2810
t_{CPU} (min)	2.218	7.973	19.539
t_{iter} (sec per iter)	0.410	0.378	0.417
Optimality	99.98%	99.9994%	99.9997%

Table 5: Simulation results for the overlapping cliques structure for the case with 8000 agents simulated using an Amazon EC2 instance with 36 cores and 60 GB RAM.

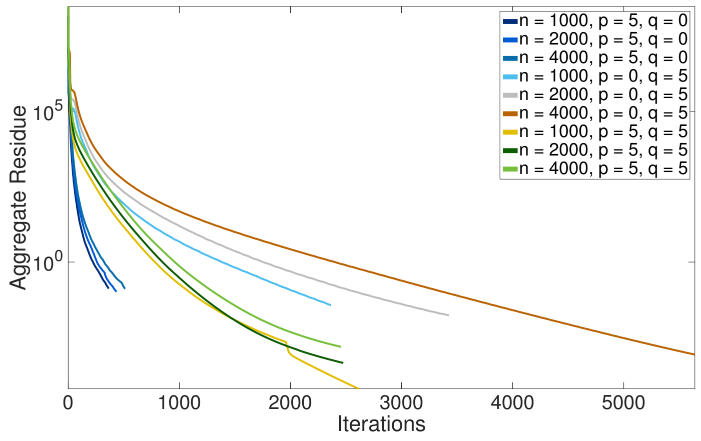


Fig. 7: Aggregate residue for all cases of the block diagonal structure simulated in Table 1

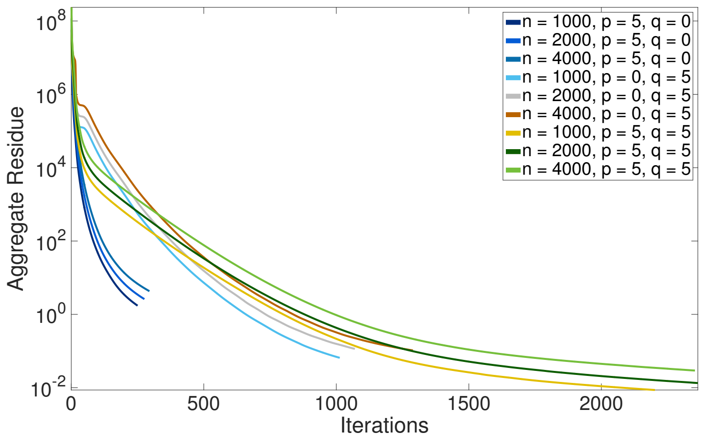


Fig. 8: Aggregate residue for all cases of the overlapping cliques structure simulated in Table 2

- The computational load performed by high-degree nodes is higher than what is performed by low-degree nodes, after considering comparable problem sizes at all nodes. This could be easily noticed from (17) since the more neighbors a node has, the longer the time needed to perform the subproblems involving the neighborhood sets $N[i]$ and $N(i)$.
- High-degree nodes need to solve large systems of linear equations, shown in (17f), which could impose memory limitations.

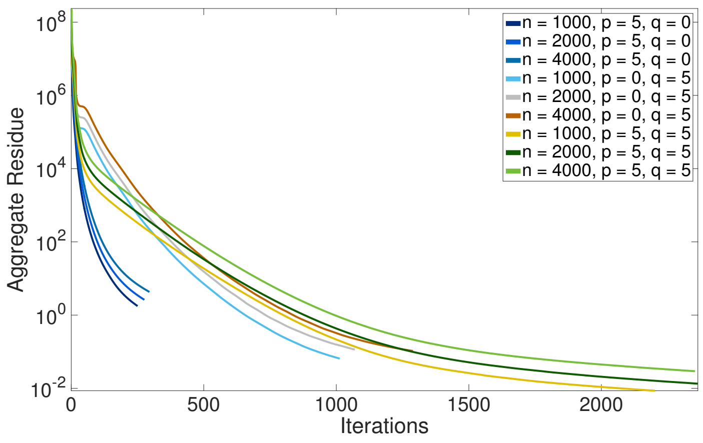


Fig. 9: Aggregate residue for all cases of the ring structure simulated in Table 3

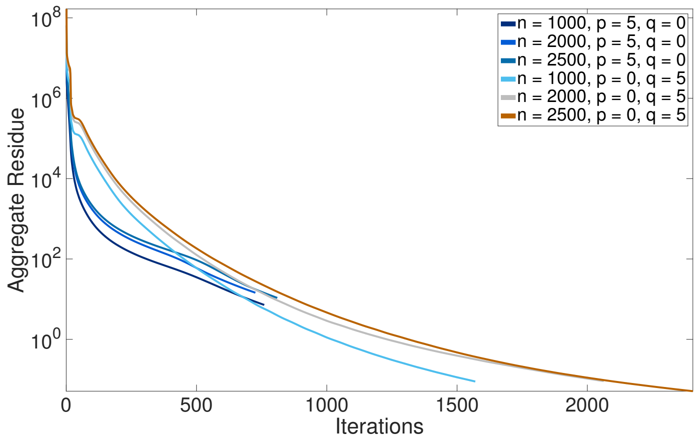


Fig. 10: Aggregate residue for all cases of the star structure simulated in Table 4

– At each iteration of the algorithm, the time needed for the high-degree nodes to complete their local computations is much higher than the time needed by the other nodes. Since our algorithm is synchronous, this will result in high idle times for the low-degree nodes.

For a better understanding of iteration, idle and busy times in a synchronous algorithm, an example is given in Figure 12 for the case of n agents. Below is a brief definition of each time after ignoring the time needed for massage passing :

– $T_{\text{busy}\ i}(t)$: the time spent by agent $i \in \mathcal{V}$ at iteration t to complete the local computation.

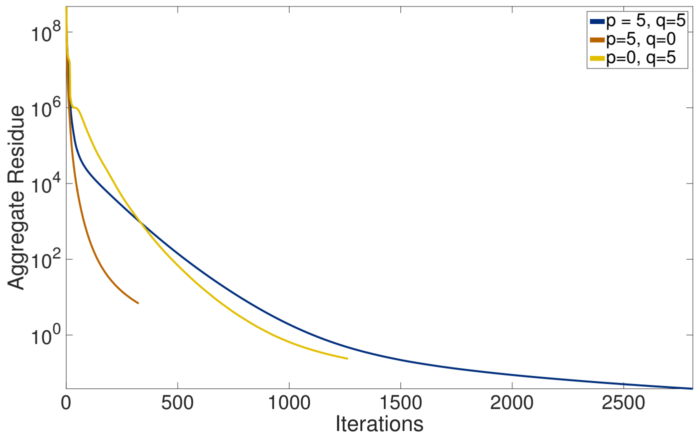


Fig. 11: Aggregate residue for the case of 8000 overlapping cliques simulated using Amazon EC2 for all cases shown in Table 5

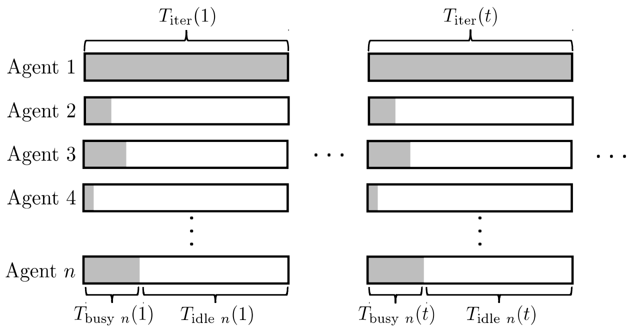
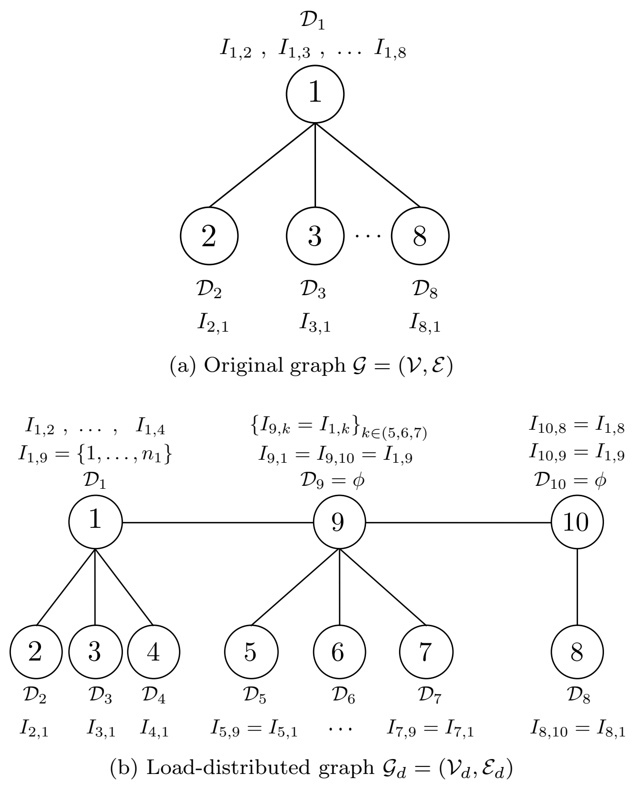


Fig. 12: An illustration of the difference between $T_{\text{busy } i}(t)$, $T_{\text{iter}}(t)$ and $T_{\text{idle } i}(t)$ for all $i \in \mathcal{V}$ and $t = 1, 2, 3, \ldots$.

– $T_{\text{iter}}(t)$: the time spent by the slowest agent at iteration t to complete the local computation (i.e. $T_{\text{iter}}(t) = \max\{T_{\text{busy } i}(t) : i \in \mathcal{V}\}$).
– $T_{\text{idle } i}(t)$: the time spent by each agent $i \in \mathcal{V}$ at iteration t between completing its local computation and then waiting for the completion of the computation at the slowest agent (i.e. $T_{\text{idle } i}(t) = T_{\text{iter}}(t) - T_{\text{busy } i}(t)$).

In what follows, we propose a method to distribute the computational load to alleviate the previous issues. The main idea behind the method is reducing the number of agents that are overlapping with the high-degree node. The method starts by introducing new copies of the high-degree node. This is equivalent to adding new nodes that are completely overlapping with the high-degree node. Overlapping with the new copies of the high-degree node is equivalent to directly overlapping with the high-degree node. Based on this



$\mathcal{D}_1$
$I_{1,2}$, $I_{1,3}$, … $I_{1,8}$
1
2
3
8
$\mathcal{D}_2$
$\mathcal{D}_3$
$\mathcal{D}_8$
$I_{2,1}$
$I_{3,1}$
$I_{8,1}$
(a) Original graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
$I_{1,2}$, … , $I_{1,4}$
$I_{1,9} = \{1, \dots, n_1\}$
$\mathcal{D}_1$
$\{I_{9,k} = I_{1,k}\}_{k \in (5,6,7)}$
$I_{9,1} = I_{9,10} = I_{1,9}$
$\mathcal{D}_9 = \phi$
$I_{10,8} = I_{1,8}$
$I_{10,9} = I_{1,9}$
$\mathcal{D}_{10} = \phi$
1
9
10
2
3
4
5
6
7
8
$\mathcal{D}_2$ $\mathcal{D}_3$ $\mathcal{D}_4$ $\mathcal{D}_5$ $\mathcal{D}_6$ $\mathcal{D}_7$ $\mathcal{D}_8$
$I_{2,1}$ $I_{3,1}$ $I_{4,1}$ $I_{5,9} = I_{5,1}$ … $I_{7,9} = I_{7,1}$ $I_{8,10} = I_{8,1}$
(b) Load-distributed graph $\mathcal{G}_d = (\mathcal{V}_d, \mathcal{E}_d)$

Fig. 13: An example of an 8-node star structure is used to demonstrate the distribution of the computational load of a node with a high degree. A threshold of degree three is chosen in this example, where Figure (a) shows the original Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and Figure (b) shows the load-distributed graph $\mathcal{G}_d = (\mathcal{V}_d, \mathcal{E}_d)$.

equivalence, some of the nodes that are overlapping with the high-degree node could be distributed to overlap with the new nodes.

For the convenience of the reader, we illustrate the method using the 8-node star structure shown in Figure 13(a). With no loss of generality, the coupling constraints (7e) and (7f) are ignored in this part. Define the set of the data matrices, after ignoring the coupling constraints, at each agent as $\mathcal{D}_i = \mathbf{A}_i$, $\{\mathbf{B}^i_j\}_{j=1}^{p_i}$, $\{\mathbf{C}^i_j\}_{j=1}^{q_i}$ and the overlapping indices as $\{I_{i,k}\}_{k \in N(i)}$. As shown in Figure 13(b), the method starts by introducing new nodes 9 and 10 that are fully overlapping with node 1 whose computational load should be distributed. The number of these additional agents is calculated by a threshold that specifies the maximum number of agents that could be connected to the high-degree node 1 and the new nodes 9 and 10. A threshold of three was chosen in this example. While nodes 2, 3 and 4 could overlap with node 1, the remaining nodes 5, 6, 7 and 8 could overlap with the new nodes 9 and 10 instead of overlapping with node 1. The overlapping indices $I_{1,5}$, $I_{1,6}$, $I_{1,7}$ and $I_{1,8}$ should be transferred from node 1 to nodes 9 and 10. It should be noticed that the data sets for the new agents 9 and 10 are empty sets ($\mathcal{D}_9 = \phi$ and $\mathcal{D}_{10} = \phi$), which means that positive semidefiniteness constraint (7d) and the consistency constraints (7g) are the only constraints available at these new nodes.

After these modifications, the algorithm in (17) could be used without any modifications to solve the newly defined problem and is equivalent to solving the original problem (they both give the same final answer). By using this method, the problem to be solved by the high-degree node is smaller and so the memory limitation is overcome and a lower Normalized Average Idle Time $T_{\text{avg idle } i}(t)$ could be achieved for each agent, which is calculated as follows:

$$T_{\text{avg idle } i}(t) = \sum_{t=1}^{|\text{iter}|} \left(\frac{T_{\text{iter}}(t) - T_{\text{busy } i}(t)}{T_{\text{iter}}(t)} \right) \times \frac{1}{|\text{iter}|} \qquad \forall\ i \in \mathcal{V}_d$$

where $|\text{iter}|$ is the number of iterations needed to achieve a desired tolerance and $\mathcal{V}_d$ is the set of nodes in the new load-distributed graph. This represents the average fraction per iteration in which the agent was idle. The lower this value is, the better the time per iteration is.

We applied the above method to a 100-node star structure with node 1 in the center. The average idle time per agent for different values of the threshold is shown in Figure 14. The average idle time per agent is lower for most threshold values, except in case of 90, than the case of not splitting the load. Finding an optimal threshold choice for each agent would ensure a larger decrease in the average idle time but this is outside the scope of this work. The aggregative residue for different values of the threshold are plotted in Figure 15, which is a monotonically decreasing function for all of the cases. The time needed for the algorithm to terminate if implemented in a multi-machine and single-machine setting at different threshold levels are plotted in Figures 16(a) and 16(b), respectively. Mutli-machine time is calculated as an estimation of the time needed for the algorithm to find a solution if implemented in a multi-machine setting. It is simply equal to the summation of the time of the slowest agent at each iteration (ignoring any communication time needed for the message passing). Serial time is the time needed for the algorithm to find a solution if run on a single machine. Both plots show that distributing the load does not have a big effect on the total algorithm time for both implementations, which is a promising result to solve the memory limitation issue.

7 Conclusion

In this paper, a fast and parallelizable algorithm is developed 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

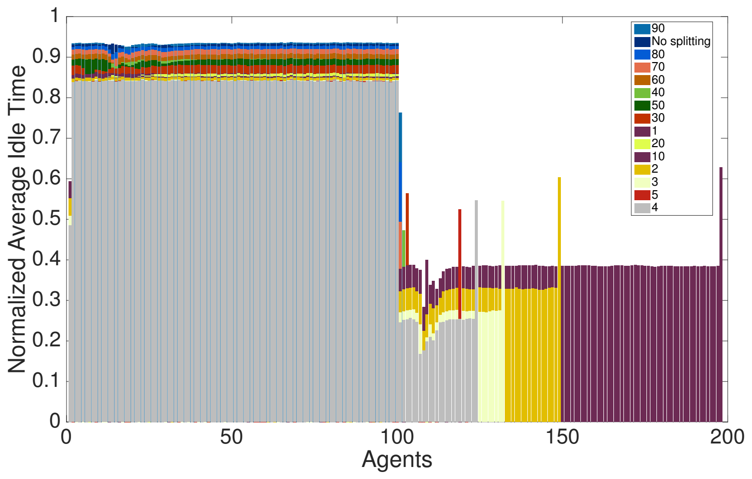


Fig. 14: This graph shows the average idle time per agent for the case of the star structure with 100 nodes and node 1 at the center. The computation time is calculated for the case where there is no splitting as well as for different values of the threshold level.

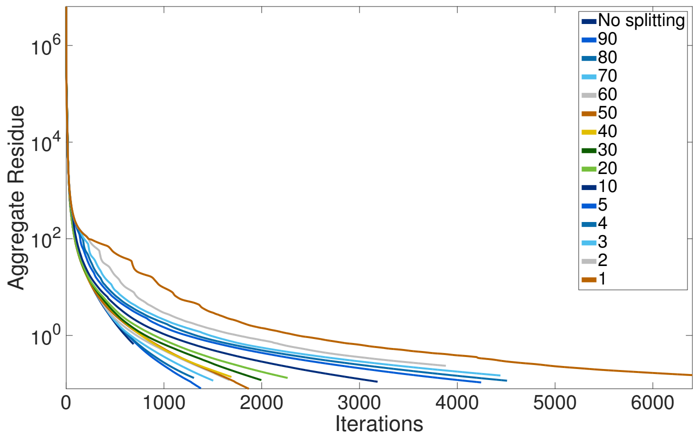
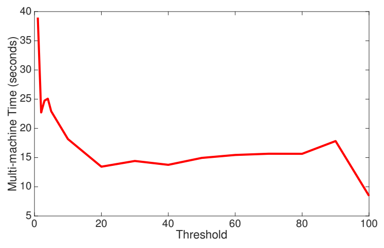
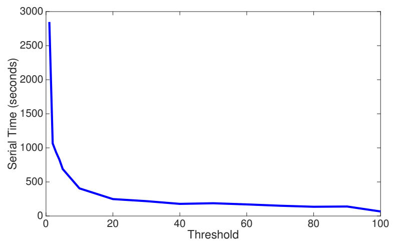


Fig. 15: Aggregate residue for different values of the threshold for the case of the star structure with 100 nodes and node 1 at the center

matrix. The main goal of each agent is to ensure that its matrix is optimal with respect to some measure and satisfies local and coupling 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



(a) Multi-machine Time at different threshold levels



(b) Serial Time at different threshold levels

Fig. 16: The multi-machine and serial time for the case of a 100-agent star structure with $p_i = 5$ and $q_i = 0$ at different threshold levels using a single core: (a) multi-machine time at different threshold levels, (b) serial time at different threshold levels.

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 the proposed technique. Finally, a method for distributing the computational load for high-degree nodes was proposed.

Acknowledgements This work was supported by DARPA YFA, ONR YIP Award, AFOSR YIP Award, NSF CAREER Award and NSF EECS Award 1406865.

References

1. Andersen, M.S., Vandenberghe, L., Dahl, J.: Linear matrix inequalities with chordal sparsity patterns and applications to robust quadratic optimization. In: Computer-Aided Control System Design (CACSD), 2010 IEEE International Symposium on. pp. 7-12. (2010)
2. Bertsekas, D.P., Tsitsiklis, J.N.: Parallel and Distributed Computation: Numerical Methods. Athena Scientific, (1997)
3. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends®in Machine Learning 3(1), 1-122 (2011).
4. Dall'Anese, E., Zhu, H., Giannakis, G.B.: Distributed optimal power flow for smart microgrids. Smart Grid, IEEE Transactions on 4(3), 1464-1475 (2013).
5. Eckstein, J., Yao, W.: Augmented Lagrangian and alternating direction methods for convex optimization: A tutorial and some illustrative computational results. RUTCOR Research Reports 32 (2012).

6. Fukuda, M., Kojima, M., Murota, K., Nakata, K.: Exploiting Sparsity in Semidefinite Programming via Matrix Completion I: General Framework. SIAM Journal on Optimization 11(3), 647-674 (2001).
7. Gabay, D., Mercier, B.: A dual algorithm for the solution of nonlinear variational problems via finite element approximation Computers and Mathematics with Applications 2(1), 17 - 40 (1976).
8. Giselsson, P., Boyd, S.: Diagonal scaling in Douglas-Rachford splitting and ADMM. 53rd IEEE Conference on Decision and Control (2014).
9. Glowinski, R., Marroco, A.: 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 9(R2), 41-76 (1975).
10. Goemans, M.X., Williamson, D.P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. Journal of the ACM (JACM) 42(6), 1115-1145 (1995).
11. Goldstein, T., O'Donoghue, B., Setzer, S., Baraniuk, R.: Fast alternating direction optimization methods. SIAM Journal on Imaging Sciences 7(3), 1588-1623 (2014).
12. Grone, R., Johnson, C.R., S, E.M., Wolkowicz, H.: Positive definite completions of partial Hermitian matrices. Linear algebra and its applications 58, 109-124 (1984).
13. He, B., Yuan, X.: On the $O(1/n)$ Convergence Rate of the Douglas-Rachford Alternating Direction Method. SIAM Journal on Numerical Analysis 50(2), 700-709 (2012).
14. He, B., Yuan, X.: On non-ergodic convergence rate of Douglas-Rachford alternating direction method of multipliers. Numerische Mathematik, 1-11 (2012).
15. Kim, S., Kojima, M., Mevissen, M., Yamashita, M.: Exploiting sparsity in linear and nonlinear matrix inequalities via positive semidefinite matrix completion. Mathematical Programming 129(1), 33-68 (2011).
16. Lavaei, J., Low, S.H.: Zero duality gap in optimal power flow problem. IEEE Transactions on Power Systems 27(1), 92-107 (2012).
17. Lessard, L., Recht, B., Packard, A.: Analysis and design of optimization algorithms via integral quadratic constraints. arXiv preprint arXiv:1408.3595 (2014).
18. Madani, R., Fazelnia, G., Sojoudi, S., Lavaei, J.: Low-rank solutions of matrix inequalities with applications to polynomial optimization and matrix completion problems. IEEE Conference on Decision and Control (2014).
19. Mittelmann, H.D.: An independent benchmarking of SDP and SOCP solvers. Mathematical Programming 95(2), 407-430 (2003).
20. Monteiro, R.D.C., Svaiter, B.F.: Iteration-Complexity of Block-Decomposition Algorithms and the Alternating Direction Method of Multipliers. SIAM Journal on Optimization 23(1), 475-507 (2013).
21. Nakata, K., Fujisawa, K., Fukuda, M., Kojima, M., Murota, K.: Exploiting sparsity in semidefinite programming via matrix completion II: implementation and numerical results. Mathematical Programming 95(2), 303-327 (2003).
22. Nesterov, Y.: A method of solving a convex programming problem with convergence rate $O(1/k^2)$. Soviet Mathematics Doklady 27(2), 372-376 (1983).
23. Nishihara, R., Lessard, L., Recht, B., Packard, A., Jordan, M.I.: A general analysis of the convergence of ADMM. arXiv preprint arXiv:1502.02009 (2015).
24. Sturm, J.F.: Using SeDuMi 1.02, A Matlab toolbox for optimization over symmetric cones. Optimization Methods and Software 11(1-4), 625-653 (1999).
25. Sun, Y., Andersen, M.S., Vandenberghe, L.: Decomposition in conic optimization with partially separable structure. SIAM Journal on Optimization 24(2), 873-897 (2014).
26. Vandenberghe, L., Boyd, S.: Semidefinite Programming. SIAM Review 38, 49-95 (1994).
27. Wei, E., Ozdaglar, A.: On the $O(1/k)$ Convergence of Asynchronous Distributed Alternating Direction Method of Multipliers. ArXiv e-prints (2013).
28. Wen, Z.: First Order Methods for Semidefinite Programming. Graduate School of Arts and Sciences, Columbia University (2009)
29. Wen, Z., Goldfarb, D., Yin, W.: Alternating direction augmented Lagrangian methods for semidefinite programming. Mathematical Programming Computation 2(3-4), 203-230 (2010).
30. Zhu, H., Giannakis, G.: Power System Nonlinear State Estimation using Distributed Semidefinite Programming. In: Selected Topics in Signal Processing, IEEE Journal of, vol. PP. vol. 99, pp. 1-12. (2014)