Distributed synchronous and asynchronous algorithms for semi-definite programming with diagonal constraints

Xia Jiang, Xianlin Zeng, Member, IEEE, Jian Sun, Member, IEEE, and Jie Chen, Fellow, IEEE

Abstract—This paper develops distributed synchronous and asynchronous algorithms for the large-scale semi-definite programming with diagonal constraints, which has wide applications in combination optimization, image processing and community detection. The information of the semi-definite programming is allocated to multiple interconnected agents such that each agent aims to find a solution by communicating to its neighbors. Based on low-rank property of solutions and the Burer-Monteiro factorization, we transform the original problem into a distributed optimization problem over unit spheres to reduce variable dimensions and ensure positive semi-definiteness without involving semi-definite projections, which are computationally expensive. For the distributed optimization problem, we propose distributed synchronous and asynchronous algorithms, both of which reduce computational burden and storage space compared with existing centralized algorithms. Specifically, the distributed synchronous algorithm almost surely escapes strict saddle points and converges to the set of optimal solutions to the optimization problem. In addition, the proposed distributed asynchronous algorithm allows communication delays and converges to the set of critical points to the optimization problem under mild conditions. By applying proposed algorithms to image segmentation applications, we illustrate the efficiency and convergence performance of the two proposed algorithms.

Index Terms—Semi-definite programming with diagonal constraints, synchronous and asynchronous algorithms, low-rank matrices, distributed optimization.

I. INTRODUCTION

SEMI-DEFINITE programming (SDP) is an active subfield of convex optimization and has attracted considerable attention due to its widely applications in diverse fields such as control theory [1]–[3], combinatorial optimization [4], [5], operations research [6], [7], and machine learning [8]–[10]. Formally, it aims to maximize or minimize a linear objective function subject to a constraint that is an affine combination of positive semi-definite matrices. One important class of SDP problem is the SDP with diagonal constraints, which is a relaxation of the “maximum cut” problem [11] and also appears in phase retrieval [12] and $\mathbb{Z}/2\mathbb{Z}$ synchronization [13].

Various algorithms have been developed to solve SDP with diagonal constraints but tend to be computational demanding as variable dimensions scale. On one hand, the arithmetic cost scales badly as the dimension of matrices increases, especially for high-order algorithms. For example, each iteration costs $O(n^3)$ arithmetic operations with an interior-point solver, which solves SDP in polynomial time [14], and the computation may run out of memory and time if $n$ is greater than several thousands [15]. On the other hand, the storage cost of each iteration may scale beyond the memory of single computer if the number of unknowns reaches tens of thousands [16]. Hence, the design of efficient algorithms for large-scale SDP with diagonal constraints is still a challenging problem.

To reduce the computational burden of large-scale SDP, there are many explorations of efficient centralized works in recent years [17]–[21]. One key idea is using the Burer-Monteiro factorization that exploits the low-rank property of matrix solutions by replacing the original large scale positive semi-definite matrix as the product of two “tall” matrices with lower dimensions to reduce storage cost and avoid computing the semi-definite projection. Using this idea, in [15], [17], [20], [21], authors transform general SDP into non-convex optimization problems by making use of the low-rank property of solutions, and propose augmented Lagrangian algorithms and Riemannian manifold methods. In addition, the challenging positive semi-definite constraints are eliminated with the cost of introducing non-convexity to the optimization problems. Surprisingly, this change to non-convex problems does not cause many difficulties because local solutions tend to recover the optimal solution in practice. Despite these advances, the existing augmented Lagrangian algorithms and Riemannian manifold methods do not guarantee converging to global optima, and suffer from slow convergence and difficulties in selecting step sizes. For SDP with diagonal constraints, some recent works [18], [22] developed block-coordinate algorithms with rigorous convergence analysis, which have free parameters and better optimization performance than prior works [19]–[22]. All these centralized algorithms own fine
practical evidences for the transformed non-convex optimizations. However, as the matrix dimension grows too large, the lower dimensional matrices in these algorithms may still take too much storage space for a single computer, such as these in some image processing problems. In addition, some information and data of practical problems may be generated and stored at different locations and cannot be communicated due to privacy considerations. Hence, these centralized algorithms cannot be applied directly to large scale problems with distributed information and distributed algorithms are in need for large-scale SDP.

Distributed optimization algorithms offer a promising approach to address large scale matrix problems by using the problem setup that the information is allocated over different agents\textsuperscript{22}. In distributed setting, agents have access to local information and communicate with their neighbors to seek for a global optimal solution\textsuperscript{23}. For large-scale SDP, many works in\textsuperscript{1, 33–36} exploited the sparse structure of SDP and introduced additional consensus constraints to the transformed distributed problems. These works proposed distributed algorithms based on alternating direction method of multipliers (ADMM) with iterative message-passing. Whereas, in ADMM, agents need to solve sub-semidefinite problems at each iteration and have considerable computational burden. Focusing on SDP with tree structures,\textsuperscript{37} proposed a distributed primal-dual interior-point algorithm for constrained semi-definite programming without introducing consensus constraints. The algorithm in\textsuperscript{37} is a second-order algorithm that conducts a recursion over the tree structure to compute the exact search directions and factorizes a relatively small matrix during each iteration. Recently,\textsuperscript{38} proposed a distributed optimization design for solving continuous-time algebraic Riccati inequalities, which have applications in distributed control of multi-agent systems. This design is a first-order algorithm and has well intuitive interpretations, but it needs computing semi-definite projections, which are expensive for large-scale matrices.

In this paper, we develop distributed first-order algorithms for large-scale semi-definite programs with diagonal constraints by taking advantage of low-rank property of solutions and the inherent sparsity of problems. The contributions of this paper are summarized as follows.

- This paper proposes a study on the distributed algorithms for SDP with diagonal constraints and distributed coefficient matrices information. This study extends the works in\textsuperscript{15, 18, 20, 22} to distributed setups, which have wide applications in power flow problems\textsuperscript{24, 25} and distributed state estimation/control\textsuperscript{26, 27}. In addition, the SDP problem in this paper does not require tree structures as in\textsuperscript{37}.
- This paper designs distributed synchronous and asynchronous algorithms for SDP with diagonal constraints by solving an equivalent nonconvex optimization problem, which is obtained using the Burer-Monteiro factorization. In particular, the distributed algorithms reduce the computational burden and storage cost on single agent compared with the existing centralized algorithms\textsuperscript{18, 22} for SDP with diagonal constraints and show a superior numerical performance in simulation experiments.

With the Burer-Monteiro factorization, the proposed algorithms avoid the computational burden of projection to semi-definite cone\textsuperscript{28}.\textsuperscript{29}. Compared with the distributed second-order interior-point algorithm in\textsuperscript{37}, the proposed first-order algorithms have lower complexity and the distributed asynchronous algorithm performs well without a global synchronous clock.

- This paper analyzes the convergence of our proposed distributed algorithms. For the distributed synchronous algorithm, we show that the variables converge to the set of global optimal solutions almost surely under random initializations, despite of the non-convexity of feasible sets. For the distributed asynchronous algorithm, we show that the variables converge to the set of critical points of the nonconvex problem under mild conditions.

The remainder of the paper is organized as follows. Mathematical notations are given in section II. The semi-definite programming description and distributed algorithms are proposed in section III. The convergence properties of the proposed algorithms are analyzed theoretically in section IV. The efficiency of distributed algorithms is verified by simulations in section V and the conclusion is made in section VI.

II. MATHEMATICAL NOTATIONS

We denote $\mathbb{R}$ as the set of real numbers, $\mathbb{R}^n$ as the set of $n$-dimensional real column vectors, $\mathbb{R}^{n \times m}$ as the set of $n$-by-$m$ real matrices, $\mathbb{N}$ as the set of natural numbers, $\mathbb{S}^n$ as the set of $n$ by $n$ symmetric matrices, $\mathbb{S}_+^n$ as the set of positive semi-definite matrices, $\emptyset$ as the empty set, respectively. All vectors in the paper are column vectors, unless otherwise noted. The notation $0_n$ denotes an $n \times 1$ vector with all elements of 0. For a real vector $v$, $\|v\|$ is the Euclidean norm and $\|v\|_1$ is 1-norm defined by the sum of absolute values of elements. We denote $A'$ as the transpose of matrix $A$, $\lambda_{\min}(A)$ as the minimum eigenvalue of the matrix $A$. For a symmetric matrix $A$, $A \succeq 0$ denotes that $A$ is positive semi-definite and $A_{(i,j)}$ is the $(i,j)$th element of matrix $A$. For real matrices $A$ and $B$ with same dimensions, $(A,B)$ denotes the Frobenius inner product of two real matrices such that $(A,B) = tr(A'B) = \sum_{i,j} A_{(i,j)} B_{(i,j)}$. In addition, $A \circ B$ denotes Hadamard product of two matrices, whose elements are defined by $[A \circ B]_{(i,j)} = A_{(i,j)} B_{(i,j)}$. For a twice-continuously differentiable function $f(x)$, its gradient and Hessian matrix are denoted as $\nabla f(x)$ and $\nabla^2 f(x)$.

For a set $S$, $|S|$ denotes the number of elements in the set $S$. For sets $S_1$ and $S_2$, $S_1 \subset S_2$ means that $S_1$ is a subset of $S_2$, $S_1 \cup S_2$ is the union of $S_1$ and $S_2$, $S_1 \cap S_2$ is the intersection of $S_1$ and $S_2$, and $S_1 \setminus S_2 = S_1 - (S_1 \cap S_2)$. For a real number $a$, $[a]$ is the smallest integer greater than $a$. For a non-zero vector $x \in \mathbb{R}^n$, the notation normal$(x)$ is $\frac{x}{\|x\|}$ and $A = \text{diag}(x) \in \mathbb{R}^{n \times n}$ denotes a matrix with diagonal element $A_{(i,i)} = x_i$. Let $\mathcal{X}$ be a smooth manifold. Let $f : \mathcal{X} \to \mathbb{R}$ be a real-valued twice-continuously differentiable function. A point $x^*$ is a critical point of $f$ if $\nabla f(x^*) = 0$. If, in addition, $\lambda_{\min} (\nabla^2 f(x^*)) < 0$, $x^*$ is a strict saddle point of $f$. 
III. PROBLEM DESCRIPTION AND DISTRIBUTED ALGORITHM DESIGN

In this section, we present the problem of solving semi-definite programming with diagonal constraints in a distributed way. Then, we reformulate the problem into a distributed non-convex optimization using the low-rank property of solutions, and propose distributed synchronous and asynchronous discrete-time algorithms.

A. Problem description and transformation

Let $G = (\mathcal{V}, \mathcal{E})$ be an arbitrary undirected and connected graph with the node set $\mathcal{V} = \{1, \ldots, m\}$ and the edge set $\mathcal{E} = \mathcal{V} \times \mathcal{V}$. Let $X \in \mathbb{S}^m_+$ be the variable. For each $i \in \mathcal{V}$, define $J_i \subseteq \{1, \ldots, n\}$ as an ordered set, where $n > m$. Throughout this paper, we use $\langle \cdot \rangle$ to denote the index of agent in $G$ and $\langle \cdot \rangle$ to denote an element of $J_i$. For example, $\tilde{i} \in \mathcal{V}$ denotes agent $\tilde{i}$ of $G$ and $j \in J_i$ is element $j$ in $J_i$. If $J_i \cap J_j \neq \emptyset$, then $(\tilde{i}, \tilde{j}) \in E$ such that agents $\tilde{i}$ and $\tilde{j}$ can communicate with each other.

The distributed semi-definite programming with diagonal constraints is

$$
\min_{X \in \mathbb{S}^n_+} \sum_{i=1}^m (\mathcal{M}^i, X),
$$

subject to $X_{(j,j)} = 1, \quad j \in \{1, \ldots, n\}$,

where $\mathcal{M}^i \in \mathbb{S}^n$ is a coefficient matrix such that $\mathcal{M}^i_{(j,k)} = 0$ if $(j, k) \notin J_i \times J_i$, and $\sum_{i=1}^m \mathcal{M}^i = M$. Define $E_{J_i \times J_i} \in \mathbb{R}^{n \times n}$ as the $0-1$ matrix with $E_{J_i \times J_i} = 1$ for $(l, k) \in J_i \times J_i$ and $E_{(l,k)} = 0$ otherwise. It is clear that $(\mathcal{M}^i, X) = (\mathcal{M}^i, E_{J_i \times J_i} \circ X)$. Without affecting solutions, the local variable $\tilde{X}^i \in \mathbb{S}^n_+$ to be determined by agent $\tilde{i} \in \mathcal{V}$ is defined as $\tilde{X}^i = E_{J_i \times J_i} \circ X$. The objective of this paper is to design a distributed algorithm for solving (1) such that each agent $\tilde{i} \in \mathcal{V}$ only knows local information $\tilde{X}^i \in \mathbb{S}^n$.

Example 3.1: Figure 1 illustrates the relationship of global optimization variable $X$ and local variables $\tilde{X}^i (i \in \{1, \ldots, 4\})$, where elements with same indices in different colored matrices are the same.

Remark 3.1: The centralized version of problem (1) is

$$
\min_{X \in \mathbb{S}^n_+} \langle \mathcal{M}, X \rangle , \quad \text{s.t.} \quad X_{i,i} = 1, \quad i \in \{1, \ldots, n\},
$$

which is a special case of generic semi-definite programming. It appears as a convex relaxation to many problems, such as the maximum cut (MAXCUT) problem, community detection and image segmentation. In practical problems such as roadmaps or social networks, the dimension $n$ may be several millions or even billions, which makes centralized computation hard. Hence, the development of distributed algorithms for (1) is of great importance.

Remark 3.2: There are two scenarios in which the problem (1) arises. In the first scenario, an arbitrary sparse SDP problem in the standard centralized form is converted into a distributed SDP with multiple positive semi-definite matrices $X^i$ by the idea of chordal decomposition of positive semi-definite cones in [41]. In the second scenario, it is assumed that the SDP is associated with a multi-agent network and matches the formulation in (1) exactly, such as large-scale image segmentation by multiple agents in section Simulation.

Since $(\mathcal{M}^i, X)$ only depends on elements with indices in $J_i \times J_i$ of variable $X \in \mathbb{S}^n_+$, define matrix $M^i \in \mathbb{S}^{|J_i|}_+ (X^i \in \mathbb{S}^{|J_i|}_+)$ as the remaining matrix by deleting elements of $\mathcal{M}^i \in \mathbb{S}^n_+ (\tilde{X}^i \in \mathbb{S}^n_+)$, whose indices are not in $J_i \times J_i$. For ease of notation, we define $X^i_{(j,k)} \triangleq \tilde{X}^i_{(j,k)}$ for $(j, k) \in J_i \times J_i$ and $\tilde{i} \in \mathcal{V}$. Similarly, define $M^i_{(j,k)} \triangleq \mathcal{M}^i_{(j,k)}$ for $(j, k) \in J_i \times J_i$ and $\tilde{i} \in \mathcal{V}$. Hence, problem (1) is equivalent to

$$
\min_{X^i \in \mathbb{S}^{|J_i|}_+, \tilde{i} \in \mathcal{V}} \sum_{i=1}^m f_i(X^i), \quad \quad f_i(X^i) = \langle M^i, X^i \rangle,
$$

subject to $X^i_{\tilde{j}, \tilde{j}} = 1, \quad \tilde{j} \in J_i, \quad X^i_{\tilde{i}, \tilde{k}} = X^i_{\tilde{k}, \tilde{i}}, \quad \forall \tilde{l}, \tilde{k} \in J_i \cap J_j, \quad (\tilde{i}, \tilde{j}) \in \mathcal{E}$,

where agent $\tilde{i} \in \mathcal{V}$ knows $M^i$ and computes a positive semi-definite matrix variable $X^i \in \mathbb{S}^{|J_i|}_+$, $(\tilde{i}, \tilde{j}) \in \mathcal{E}$ specifies an overlap between the local variables $X^i$ and $X^j$ of agents $\tilde{i}$ and $\tilde{j}$.

In most existing distributed works [9, 10] for (1), the updating of local variable $X^i$ often involves a projection operator to the positive semi-definite cone. If the dimension of local $X^i$ of large-scale SDP is large, the projection operator is difficult and time-consuming. Hence, based on prior works on the low-rank property of matrix variables, we further reduce the computational and storage burden by representing $X \in \mathbb{R}^{n \times n}$ by $V^TV$ with $V = [v_1, \ldots, v_n] \in \mathbb{R}^{p \times n}$ to avoid the projection operator. It is well-known that the rank of an optimal solution is at most $\lfloor \sqrt{2n} \rfloor$ (see [42]). Let $v^s$ be the estimate of $v_s$ by agent $\tilde{i}$ for $s \in J_i$ and $\tilde{i} \in \mathcal{V}$. Without causing confusions, we define $V^i = [v^s_{i,1}, \ldots, v^s_{i,J_i}] \in \mathbb{R}^{p \times |J_i|}$, where $\{s_1, \ldots, s_{|J_i|}\} = J_i, s_1 < \cdots < s_{|J_i|}$, and $\tilde{i} \in \mathcal{V}$. Hence, the local variable $X^i$ is replaced by $\tilde{X}^i = V^iV^i$, where $V^i \in \mathbb{R}^{p \times |J_i|}$, $p > \sqrt{2n}$. Then the semi-definite programming (1) is rewritten as the following non-convex optimization problem on unit spheres:

$$
\min_{V^i, \tilde{i} \in \mathcal{V}} \sum_{i=1}^m f_i(V^i), \quad \quad f_i(V^i) = \langle M^i, V^iV^i \rangle,
$$

subject to $\|v^s_{i,j}\| = 1, \quad \forall \tilde{i} \in \mathcal{V}, \quad j \in J_i, \quad (\tilde{i}, \tilde{j}) \in \mathcal{E}$,

where $V^i \in \mathbb{R}^{p \times |J_i|}$ is local variable and $M^i \in \mathbb{S}^{|J_i|}_+$ is local coefficient matrix known by agent $\tilde{i} \in \mathcal{V}$.

The following assumption is needed.

Assumption 3.1:

(1) Graph $G$ is undirected and connected.

(2) Each element of global coefficient matrix $M = \sum_{i=1}^m \mathcal{M}^i$ is non-negative and diagonal elements of $M$ are zero.
Fig. 1. Matrix decomposition of global variable $X$ over four agents.

Remark 3.3: Assumption 3.1 (1) is general in distributed optimization. Since the norm of column variables $v_i$ is fixed as one, $M_{i,i} = 0$ does not affect the solution of optimization problem. In addition, because elements of coefficient matrix in MAXCUT problems and community detection are non-negative, Assumption 3.1 (2) is practical for problem (1). Assumption 3.1 (3) is a sufficient condition that optimal solutions for $V$'s recover optimal solutions for $X$'s.

B. Distributed synchronous optimization algorithm

In this subsection, we propose a distributed synchronous algorithm for the transformed distributed non-convex optimization problem (4).

To present the algorithm, we need additional definitions and notations. We define "children" and "parents" in graph $G$. For the problem (4), there is an edge between agents $i$ and $j$ if $J_i \cap J_j \neq \emptyset$. Without loss of generality, if indices $j < i$ and $J_i \cap J_j \neq \emptyset$, the agent $j$ is called the parent of agent $i$, denoted by $\text{par}(i)$, and the set of children of agent $i$ is denoted by $\text{ch}(i)$. The message passed from agent $\tilde{r} \in \text{ch}(i)$ to agent $\tilde{i} \in V$ is denoted by $\omega_{\tilde{r},\tilde{i}}$. Define the coupling set of indices between column variables of agent $i$ and its parent $\text{par}(i)$ as $S_{i,\text{par}(i)} = J_i \cap J_{\text{par}(i)}$ and the uncoupling set of indices as $R_{i,\text{par}(i)} = J_i \setminus S_{i,\text{par}(i)}$. In addition, column variables $v^j_i$'s with $j \in S_{i,\text{par}(i)}$ are called coupling variables of agent $i$ and its parent $\text{par}(i)$, and column variables $v^j_i$'s with $j \in R_{i,\text{par}(i)}$ are called uncoupling variables of agent $i$ and its parent $\text{par}(i)$. Accordingly, we have the similar notations $S_{\text{ch}(i),\tilde{i}}$ and $R_{\text{ch}(i),\tilde{i}}$.

Example 3.2: Fig. 2 gives a network to show the previous definitions of $S_{i,\cdot}$ and $R_{i,\cdot}$. Noted that $S_{\text{ch}(i),\tilde{i}}$ is an empty set for agent $\tilde{i}$ with no children, such as agents $3, 4, 5$. Accordingly, the message $\omega_{\tilde{r},\tilde{i}}$, $\tilde{r} \in \text{ch}(i)$ passed from children agents is zero for $\tilde{i} = 3, 4, 5$. The similar case $S_{i,\text{par}(i)} = \emptyset$ holds for the root agent, which have no parents.

For each agent $\tilde{i} \in V$ and $j \in R_{i,\text{par}(i)}$, define

$$p^j_i(t) \triangleq \sum_{l \in J_i} M^j_{i,l} v^l_i(t) + \sum_{l > j, l \in J_i} M^j_{l,i} v^l_i(t).$$

Define the step-size $\theta^j_i$ as

$$\theta^j_i \in (0, \frac{1}{\|M^j_{\{j,\cdot}\}}}), \ j \in R_{i,\text{par}(i)}$$

where $M^j_{\{j,\cdot\}}$ denotes the $j$th row of matrix $\overline{M}$.

The message passed from agent $\tilde{r} \in \text{ch}(i)$ to its parent $\tilde{i} \in V$ is defined as

$$\omega_{\tilde{r},\tilde{i}}(t + 1) = \sum_{l \in J_{\tilde{r}}} M^l_{\tilde{r},l} v^l_{\tilde{r}}(t + 1) \quad \forall j \in S_{\tilde{r},\tilde{i}}.$$  \hspace{1cm} (6)

and

$$\omega_{\tilde{r},\tilde{i}}(t + 1) = 0 \quad \forall j \in R_{\tilde{r},\text{par}(i)} \setminus S_{\tilde{r},\tilde{i}}.$$

For $\tilde{i} \in V$ and $j \in R_{i,\text{par}(i)}$, the variable $v^j_i(t + 1)$ is updated as

$$v^j_i(t + 1) = \text{normal}(v^j_i(t) - \theta^j_i[p^j_i(t) + \sum_{\tilde{r} \in \text{ch}(i)} \omega_{\tilde{r},\tilde{i}}(t + 1)])$$  \hspace{1cm} (7)

The message sent from agent $\tilde{i} \in V$ to its parent $\text{par}(i)$ is

$$\omega_{\tilde{i},\text{par}(i)} = \sum_{l \in J_i} M^l_{\tilde{i},l} v^l_{\tilde{i}}(t + 1) + \sum_{\tilde{r} \in \text{ch}(i)} \omega_{\tilde{r},\tilde{i}}(t + 1),$$  \hspace{1cm} (8)

Define $V = (V^1, \cdots, V^m)$. The distributed synchronous algorithm is given in Algorithm II.
Algorithm 1 Distributed Synchronous Algorithm (DSA)

1: Initialization: Initialize $v_j^0 = v_0 \in \mathbb{R}^p$ such that $\|v_0\| = 1$
   for all $i \in \mathcal{V}$ and $j \in J_i$.
2: while the stopping criteria is not satisfied do
3:   for $i = m$ to $i = 1$ do
4:     for $j \in R_{i,\text{par}(i)}$ do
5:       if $j \in S_{\text{ch}(i),i}$ then
6:         agent $i$ receives information $\omega_j^{\tilde{r},\tilde{i}}(t)$ (computed by (6)) from its child $\tilde{r} \in \text{ch}(i)$.
7:         agent $i$ updates variable $v_j^i(t+1)$ following (7).
8:         if $j \in S_{\text{ch}(i),i}$ then
9:           agent $i$ sends $v_j^i(t+1)$ to its child $\tilde{r} \in \text{ch}(i)$,
10:          i.e.,
11:           $v_j^\tilde{r}(t+1) = v_j^i(t+1)$. (9)
12:       end if
13:     end for
14:   end for
15: end while

The convergence performance of Algorithm 1 is provided in the following theorem, whose proof is given in the next section.

Theorem 3.1: Let Assumption 3.1 hold and $\{V(t)\}$ be a sequence generated by Algorithm 1. Then $V(t)$ converges to global optimal solutions to (4) almost surely under random initialization as $t \to \infty$.

Remark 3.4: Note that there may be several parents for each agent in practice. In this paper, we only consider the case that each agent owns one parent for convenience of analysis. However, the algorithm can be easily extended to cases where agents have multiple parents. We provide examples in simulations to demonstrate the cases of tree-structured graphs of which one agent has a unique parent and graphs of which one agent may have multiple parents.

Remark 3.5: Compared with the existing centralized works [18], [43], the proposed algorithm decentralizes the storage space and computational burden of large-scale SDP over different agents at the cost of network communication. In addition, the proposed algorithm is applicable for the scenario where global information is located on geographically separated agents such that centralized algorithms can not handle.

C. Distributed asynchronous optimization algorithm

The synchronous algorithm given by Algorithm 1 needs a global clock and the updating rate of variables is limited by the slowest agent. Whereas, asynchronous algorithms update variables by local clocks and allow communication time-delays, then the variables updating of one agent will not be limited by other agents. Hence, in this subsection, we provide a distributed asynchronous algorithm for solving problem (4).

Let $T^\ast$ be the set of times at which agent $i$ updates local variable $V^i$. Noted that agent $i$ may not have access to the most recent value of other agents’ variables. Then, we define that the updating time of variables $v_j$ at parent or child node, $\tau_j^i(t)$, satisfies

$$0 \leq \tau_j^i(t) \leq t, \quad \forall t \in T^\ast,$$

where $j \in J_{\text{ch}(i)} \cup J_{\text{par}(i)} \cup J_i$. If $j \in R_{i,\text{par}(i)}$, then $\tau_j^i(t) = t$. The difference $(t - \tau_j^i(t))$ between the current time $t$ and $\tau_j^i(t)$ is viewed as a form of communication delay between agent $i$ and its parents or children.

For the communication delay between agents, we assume that the following condition holds.

Assumption 3.2: (Partial Asynchronism) There exists a positive integer $B$ such that:

- For each agent $i$ and $t \geq 0$, at least one element of the set $\{t, t + 1, \cdots, t + B - 1\}$ belongs to $T^\ast$.
- There holds $\max\{0, t - B + 1\} \leq \tau_j^i(t) \leq t,$

for all agent $i \in \mathcal{V}$ and $j \in \{1, \cdots, n\}$ and all $t \geq 0$.

In the distributed asynchronous Algorithm 2 local variable $V^i$ is updated by the time-delayed messages communicated from neighbors and local information. For each agent $i$, define $p_j^i(t) = \sum_{l \in J_i} M_{l,j,i}^i v_j^l(t)$, and the step-size $\theta_j^i$ satisfies

$$\theta_j^i = \frac{1}{(1 + B + nB)L^i},$$

where $L = \max_{i \in \mathcal{V}} \{l_1, \cdots, l_m\}$, $l_i$ is the best Lipschitz constant of $\nabla f_j(V^i)$ [$f_j(V^i)$ was defined in (4)] and $B$ is defined in Assumption 3.2.

The message passed from child $\tilde{r} \in \text{ch}(i)$ to parent $\tilde{i}$ is defined as

$$\omega_j^{\tilde{r},\tilde{i}}(\tau_i^i(t)) = \sum_{l \in J_i} M_{l,j,i}^i v_j^l(\tau_i^i(t))$$

and for variables with indices $j \in R_{i,\text{par}(i)} \setminus S_{\tilde{r},\tilde{i}}$,

$$\omega_j^{\tilde{r},\tilde{i}}(\tau_i^i(t)) = 0.$$

If $t \in T^\ast$, the variable $v_j^i(t + 1)$ is updated as

$$v_j^i(t + 1) = \text{normal} \left( v_j^i(t) - \theta_j^i [p_j^i(t) + \sum_{\tilde{r} \in \text{ch}(i)} \omega_j^{\tilde{r},\tilde{i}}(\tau_i^i(t))] \right),$$

$$j \in R_{i,\text{par}(i)}, \quad \tau_i^i(t) \in T^\ast,$$ (12a)

$$v_j^i(t + 1) = v_j^{\text{par}(i)}(\tau_i^i(t)),$$

$$j \in S_{i,\text{par}(i)}.$$ (12b)

For times $t \notin T^\ast$, the variable $v_j^i$ is unchanged,

$$v_j^i(t + 1) = v_j^i(t), \quad \forall j \in J_i.$$
The message sent from agent $i \in \mathcal{V}$ to its parent $par(i)$ is
\[
\bar{w}^{\text{par}(i)}_j(t) = \sum_{l \in J_i} M_{l,j}^i v^l_i(t + 1) + \sum_{r \in \text{ch}(i)} \bar{w}^{r,i}_j(r^r(i), j \in \hat{J}_i).
\]
(13)

Algorithm 2 Distributed Asynchronous Algorithm (DAA) - from the view of agent $i$

1. **Initialization:** Initialize $v^i_j = v_0 \in \mathbb{R}^p$ such that $\|v_0\| = 1$
   for all $i \in \mathcal{V}$ and $j \in \hat{J}_i$.
2. **while** the stopping criteria is not satisfied do
   3. for each agent $i$, keep receiving information $\bar{w}^{r,i}_j(r^r(i))$ from children and receiving information $v^{\text{par}(i)}_j(r^r(i))$ from parent.
   4. if $t \in T^i$ then
   5. for $j \in \hat{J}_i$ do
   6. agent $i$ updates variable $v^i_j(t + 1)$ following (12).
   7. if $j \in \mathcal{S}_{i,\text{par}(i)}$ then
   8. agent $i$ sends message $\bar{w}^{\text{par}(i)}_j$ (computed by (13)) to its parent,
   9. end if
   10. if $j \in \mathcal{S}_{i,j}$, $\hat{i} \in \text{ch}(i)$ then
   11. agent $i$ sends local variable $v^i_j(t + 1)$ to each child $\hat{i}$ that has coupling variable $v^\hat{i}_j, j \in \mathcal{S}_{\hat{i},j}$.
   12. end if
   13. end for
   14. end if
   15. $t \leftarrow t + 1$.
16. end while

Before providing the convergence performance of DAA, we need one additional assumption, which is vital in the transformation of proposed algorithm.

Assumption 3.3: Each element of matrix $M$ is only accessible to one agent, which implies that for each agent $i \in \mathcal{V}$, $M^i_{l,k} = M_{l,k}$ holds for $(l, k) \in J_i \times \hat{J}_i$.

Next, the convergence performance of DAA is provided in the following theorem, whose analysis is shown in the section IV-B.

**Theorem 3.2:** Under Assumptions 3.1-3.3, the sequence $\{V(t)\}$ generated by DAA converges to critical points to (4) as $t \to \infty$.

**Remark 3.6:** In the implementation of DAA, each agent receives communication information from its neighbors and stores it in local buffer. The local received data may be out-of-date due to time-delays. Each agent $i \in \mathcal{V}$ updates local variables using data in local buffer at the time $t \in T^i$ and does not have to wait for the point when other local communicating messages become available. It allows some agents to compute faster and execute more iterations than others.

IV. THEORETICAL ANALYSIS

In this section, we present theoretical proofs for the convergence properties of proposed distributed synchronous and asynchronous algorithms, respectively.

A. Convergence analysis for synchronous algorithm

Firstly, we develop a compact form of the proposed synchronous algorithm containing all agent’s updates.

For any $j \in \{1, \ldots, n\}$, let $v_j = v^i_j$, where $j \in \mathcal{R}_{i,\text{par}(i)}$ and $v^i_j$ is a column vector of the variable of agent $i \in \mathcal{V}$. Then we define the global variable
\[
V = [v_1, \ldots, v_n] \in \mathbb{R}^{p \times n}.
\]
(14)

**Remark 4.1:** For $j \in \mathcal{S}_{i,\text{par}(i)}$, by the updating design (9), there is a parent of $\hat{i}$, $\hat{j} \in \text{par}(\hat{i})$, such that $j \in \mathcal{R}_{\hat{j},\text{par}(\hat{j})}$ and $v^j_j(t) = v^\hat{j}_j(t)$. Hence, in the following analysis, for convenience, we consider the global variable $V$ instead of $V$.

Without loss of generality, we make the following assumptions in the analysis.

Assumption 4.1: Take any $\tilde{i} \in \mathcal{V}$. Indices $j \in \mathcal{R}_{\text{ch}(\tilde{i})}, i$, $l \in \mathcal{R}_{i,\text{par}(i)}$ and $p \in \mathcal{S}_{i,\text{par}(i)}$ satisfy that $p > l > j$.

Notice that for each column vector $v_j$ with index $j \in \mathcal{S}_{i,\text{par}(i)}$, there must exist one agent $\hat{j}$ such that $j \in \mathcal{R}_{\hat{j},\text{par}(\hat{j})}$.

**Assumption 4.2:** We assume that there is no shared variables between agents $\text{ch}(i)$ and $\text{par}(i)$ for any $i \in \mathcal{V}$. That is, $\mathcal{S}_{\text{ch}(i), i} \subset \mathcal{R}_{i,\text{par}(i)}$ for all $i \in \mathcal{V}$.

Note that Assumptions 4.1 and 4.2 are not needed in the proposed synchronous algorithm, which is developed for the convenience of proof.

Define $\Theta \in \mathbb{R}^{n \times n}$ as a diagonal matrix with diagonal elements,
\[
\Theta_{(j,j)} = \theta^\hat{j}_j \quad \text{if} \quad j \in \mathcal{R}_{i,\text{par}(i)}.
\]
(15)

Then, under the above assumptions, we obtain the following result.

**Lemma 4.1:** Under Assumptions 4.1 and 4.2, the distributed synchronous algorithm is equivalent to
\[
v_j(t + 1) = \text{normal}\left(v_j(t) - \Theta_{(j,j)} g_j(t)\right), \quad \forall j \in \{1, \ldots, n\},
\]
(16)
where $g_j(t) = \sum_{l \in \{1, \ldots, j-1\}} M_{j,l} v_l(t + 1) + \sum_{l \in \{j+1, \ldots, n\}} M_{j,l} v_l(t)\]

**Proof:** For variables $v_j$ of agent $\tilde{i}$ such that $j \in \mathcal{R}_{i,\text{par}(i)}$, we substitute (6) to the updating (7) and get
\[
v^\tilde{i}_j(t + 1) = \text{normal}\left(v^\tilde{i}_j(t) - \theta^\tilde{i}_j g^\tilde{i}_j(t)\right), \quad j \in \mathcal{R}_{i,\text{par}(i)}
\]
(17)
where
\[
g^\tilde{i}_j(t) = \sum_{l \in \{1, \ldots, j-1\}} M^l_{j,l} v^l_i(t + 1) + \sum_{l \in \{j+1, \ldots, n\}} M^l_{j,l} v^l_i(t) + \sum_{r \in \text{ch}(\tilde{i})} \sum_{l \in J_r} M^r_{l,j} v^r_l(t + 1).
\]
(18)

The condition $j \in \mathcal{R}_{i,\text{par}(i)}$ holds in the whole process of proof. For convenience, in the following analysis, we omit this condition.
Since \(M_{\{j,l\}}^f = \overline{M}_{\{j,l\}}\), for \((j,l) \in J_F \times J_F\) and \(\tilde{r} \in \mathcal{V}\), we have

\[
\sum_{\tilde{r} \in I(\text{ch}(\tilde{r}))} M_{\{j,l\}}^f = \sum_{\tilde{r} = 1}^{m} \overline{M}_{\{j,l\}} = M_{\{j,l\}}^f. \tag{19}
\]

Then, by substituting (20) to \(g_{i,j}^f = \sum_{l \in J_F} M_{\{j,l\}}^f v_l^f(t + 1)\), the sum of \(MS\) and \(MR\) is

\[
MS + MR = \sum_{\tilde{r} \in I(\text{ch}(\tilde{r}))} \sum_{l \in I(\text{ch}(\tilde{r})) \cap \{l < j\}} M_{\{j,l\}}^f v_l^f(t + 1)
\]

where the first equality holds because the condition \(\mathcal{S}_{\text{ch}(\tilde{r}),i} \subseteq \mathcal{R}_{\text{i,par}(\tilde{r})}\), and the last equality holds due to the relationship (19).

By Assumption 4.1, any index \(l \in \mathcal{R}_{\text{ch}(\tilde{r})}\) satisfies \(l < j\) for \(j \in \mathcal{R}_{\text{i,par}(\tilde{r})}\). Then, with (22),

\[
\zeta_j(t + 1) = \sum_{l \in (\text{ch}(\tilde{r}))} M_{\{j,l\}}^f v_l^f(t + 1) = \sum_{l \in (\mathcal{R}_{\text{i,par}(\tilde{r})})} M_{\{j,l\}}^f v_l^f(t + 1) \tag{23}
\]

(2) Consider \(t_j\) composed of variables with indices \(l\) in the set \(\{l | l > j, j \in \mathcal{R}_{\text{i,par}(\tilde{r})}\} \) \(M_{\{j,l\}}^f v_l^f(t)\) in \(t_j(t)\) satisfies

\[
\sum_{l \in (\text{ch}(\tilde{r}))} M_{\{j,l\}}^f v_l^f(t) = \sum_{l \in (\mathcal{R}_{\text{i,par}(\tilde{r})})} M_{\{j,l\}}^f v_l^f(t) \tag{24}
\]

It follows from a similar analysis in (22) that the sum of \(\sum_{l \in (\text{ch}(\tilde{r}))} \sum_{l \in (\text{ch}(\tilde{r})) \cap (l > j)} M_{\{j,l\}}^f v_l^f(t)\) in \(t_j(t)\) and \(\sum_{l \in (\mathcal{R}_{\text{i,par}(\tilde{r})}) \cap (l > j)} M_{\{j,l\}}^f v_l^f(t)\) in (24) is

\[
\sum_{l \in (\text{ch}(\tilde{r}))} \sum_{l \in (\text{ch}(\tilde{r})) \cap (l > j)} M_{\{j,l\}}^f v_l^f(t) + \sum_{l \in (\mathcal{R}_{\text{i,par}(\tilde{r})}) \cap (l > j)} M_{\{j,l\}}^f v_l^f(t) = \sum_{l \in (\mathcal{R}_{\text{i,par}(\tilde{r})}) \cap (l > j)} M_{\{j,l\}}^f v_l^f(t). \tag{25}
\]

Then, consider the term \(\sum_{l \in (\text{ch}(\tilde{r}))} \sum_{l \in (\text{ch}(\tilde{r})) \cap (l > j)} M_{\{j,l\}}^f v_l^f(t)\) in (24). Because all column variables are initialized as a same value and the condition \(\mathcal{S}_{\text{ch}(\tilde{r}),i} \subseteq \mathcal{R}_{\text{i,par}(\tilde{r})}\), we have, for \(j \in \mathcal{R}_{\text{i,par}(\tilde{r})}\),

\[
\sum_{l \in (\text{ch}(\tilde{r})) \cap (l > j)} M_{\{j,l\}}^f v_l^f(t) = \sum_{l \in (\mathcal{R}_{\text{i,par}(\tilde{r})}) \cap (l > j)} M_{\{j,l\}}^f v_l^f(t) = \sum_{l \in (\mathcal{R}_{\text{i,par}(\tilde{r})}) \cap (l > j)} M_{\{j,l\}}^f v_l^f(t). \tag{26}
\]

where the equality holds because for each child \(\tilde{r} \in \text{ch}(\tilde{r})\), \(M_{\{j,l\}}^f = 0\), for \(j \in \mathcal{R}_{\text{i,par}(\tilde{r})}, l \in \mathcal{R}_{\text{i,par}(\tilde{r})}\)
Then, with (25) and (26),
\[
i_j(t) = \sum_{i \in R_{i,par(i)} \cap \{t > j\}} M_{i,j}(t) v_i(t) + \sum_{l \in S_{i,par(i)} \cap \{t > j\}} M_{j,l}(t) v_l(t) = \sum_{i \in \{j+1, \ldots, n\}} M_{i,j}(t). \tag{27}
\]
Hence, by (21), (23) and (27), the updating (17) of agent $i$ for any vector variable with index $j \in R_{i,par(i)}$ is
\[
v_i(t+1) = \text{normal}(v_i(t) - \Theta_{i,j}(t) g_j(t)), \tag{28}
\]
where $g_j(t) = \sum_{i \in \{t < j \in R_{i,par(i)} \}} M_{i,j}(t) v_i(t) + 1 + \sum_{i \in \{j+1, \ldots, n\} \cap \{t \in R_{i,par(i)} \}} M_{j,i}(t) v_j(t)$ and $\Theta_{i,j} = \theta_j^i$, which is defined in (5). Since (28) holds for each agent $i$, we obtain the desired result (16) with (14).

Next, we will discuss the convergence properties of proposed distributed synchronous algorithm. Because the optimization problem (4) is a non-convex minimization problem, there may exist several local minima and saddle points, which are regarded as major obstacles for global minima search over continuous spaces. For the semi-definite programming like (1), it has been known that the low-rank transformed problem (4) has no local optima except the global ones if $p > \sqrt{2n}$ [15].

Thus, the main work is to discuss whether the proposed algorithm escapes strict saddle points and converges to global optimal solutions. At first, we provide the definition of unstable critical points. Denote the update of $V$ generated by DSA as $V(t+1) = h_{SM}(V(t))$.

**Definition 4.1:** Define unstable critical points as the set of critical points where the Jacobian of variable updating $h_{SM}(V) = \{V: h_{SM}(V) = V, \max_i |\lambda_i(Dh_{SM}(V))| > 1\}$.

Following the work in [43], we have the following property, which was investigated in [18].

**Lemma 4.2:** If $p > \sqrt{2n}$, each strict saddle point $V^*$ of the updating $h_{SM}$ is an unstable critical point, meaning $\lambda^* \in A^*$, where $\lambda^*$ is the set of strict saddle points.

**Proof:** Consider the equivalent form in Lemma 4.1 of the proposed synchronous algorithm. It follows from the proof in [18] that the Jacobi of proposed algorithm has eigenvalues containing those of the Jacobi of a standard Gauss-Seidel updating proposed in [43]. Based on the discussions of standard Gauss-Seidel updating in [43], we obtain the desired result.\[\]

From Lemma 4.2, we deduce that all non-optimal critical points are unstable fixed points. Next, we will prove that the updating $h_{SM}$ is a diffeomorphism, which is an invertible function that maps one differentiable manifold to another such that both the function and its inverse are smooth.

**Lemma 4.3:** Under Assumptions 4.1 and 4.2 the distributed synchronous updating $h_{SM}$ is a diffeomorphism.

**Proof:** By the designed variable updating in Algorithm 1 and Lemma 4.1, $h_{SM}$ is equivalent to
\[
h_{SM}(V) = [\psi_n(\psi_{n-1}(\cdots \psi_1(V)))]
\]
where each column variable updating is defined as
\[
(\psi_i(V))_{s=1 \cdots n} = \begin{cases} \frac{v_i - \Theta_{i,i}(t) VM_{i,i}}{v_i - \Theta_{i,i}(t) VM_{i,i}} & \text{if } s = i \\ v_s & \text{otherwise}. \end{cases}
\]

Because a composition of diffeomorphisms is still a diffeomorphism [19], to prove this lemma, we only need to prove that $\psi_i(V)$ is a diffeomorphism for $i = 1, \cdots, n$.

In (16), the step size takes a constant $\Theta_{i,i}(t) = (1 - \sigma) \frac{1}{\|M_{i,i}(t)\|}$. Because $M$ is symmetric, it is equivalent to taking $1 - \sigma \in (0, 1)$ and from the triangular inequality,
\[
\|VM_{i,i}\| = \left\| \sum_{j=1}^{n} M_{i,j} v_j \right\| \leq \sum_{j=1}^{n} \|M_{i,j}\| \|v_j\| = \|M_{i,i}\|.
\]

Hence, $\|\Theta_{i,i}(t) VM_{i,i}\| \leq 1 - \sigma < 1$. Thus, we have $\|v_i - \Theta_{i,i}(t) VM_{i,i}\| \geq 1 - \|\Theta_{i,i}(t) VM_{i,i}\| \geq \sigma > 0$. Note that the function $\psi_i$ is only non-smooth at the point where the denominator term $|v_i - \Theta_{i,i}(t) VM_{i,i}| = 0$ and we have proved that the term is greater than 0. Therefore, the function $\psi_i$ and its inverse function are valid and smooth. By the work in [18] Lemma C.2, $\psi_i$ is a diffeomorphism. Since $h_{SM}(\cdot)$ is the composition of $\psi_i(\cdot)s$, the mapping of $h_{SM}(V)$ is also a diffeomorphism.

Next, with the definition of global variable $V$ in (14), we provide one equivalent form of objective function in optimization problem (4), which will be used in the analysis of Lemma 4.5.

**Lemma 4.4:** With the definition (14) and Assumptions 4.1 and 4.2, the objective function of (4) is at time $t$
\[
\sum_{i=1}^{m} \langle M_i, V_i(t) V_i(t) \rangle = \langle M, V(t) V(t) \rangle.
\]

**Proof:** The objective function of optimization problem (4) is $\sum_{i=1}^{m} \langle M_i, V_i^2 \rangle$. By the algorithm design, after one iteration, $v_i(t) = v_j(t)$ for all $j \in J_i$, which holds because the coupling variables $v_j$ with $j \in S_{i,par(i)}$ are equal to the uncoupling variables $v_{par(i)}$. Hence, the constraints (4b) and (4c) in (4) hold. Then, we obtain
\[
\sum_{i=1}^{m} \langle M_i, V_i^2 \rangle (t) V_i(t) \rangle = \langle \sum_{i=1}^{m} M_i(t, h) v_i(t) v_i(t) \rangle = \langle \sum_{i=1}^{m} M_i(t, h) v_i(t) v_i(t) \rangle = \langle \sum_{i=1}^{m} (M_i(t, h) v_i(t) v_i(t)) \rangle = \langle \sum_{i=1}^{m} M_i(t, h) v_i(t) v_i(t) \rangle = \langle \sum_{i=1}^{m} M_i(t, h) v_i(t) v_i(t) \rangle = \langle \sum_{i=1}^{m} M_i(t, h) v_i(t) v_i(t) \rangle = \langle M, V(t) V(t) \rangle.
\]
With Lemma 4.4 before updating $v_i$, all variable $v_j$ except for $v_i$ are given and fixed, then the global function is rewritten as

$$
f(V) = <M, V'V> = \sum_{i=1}^{n} \sum_{j=1}^{n} M_{i,j} v_i' v_j = 2h \theta_i g_i + \text{constant}, \tag{29}
$$

where the last equation holds since the matrix $M$ is symmetric. Note that $g_i$ is defined in Lemma 4.1 and is independent of $v_i$ because $M_{i,i} = 0$. Then, we have the following Lemma stating the monotonous decreasing property of the global function value generated by the proposed synchronous algorithm.

**Lemma 4.5**: For the proposed synchronous algorithm with step size $\Theta$, let $V(t+1) = h_{SM}(V(t))$. Under Assumptions 4.1 and 4.2, we have

$$
f(V(t)) - f(V(t+1)) = \sum_{i=1}^{n} \frac{1 + y_i(t)}{\Theta(i,i)} \|v_i(t) - v_i(t+1)\|^2,
$$

where $y_i(t) = \|v_i(t) - \Theta(i,i) g_i(t)\|$ and $g_i(t) = \sum_{l<s} M_{i,l} v_l(t+1) + \sum_{l>s} M_{i,l} v_l(t)$.

**Proof**: By (29), the function difference after updating $v_i(t)$ to $v_i(t+1)$ is $2g_i'(v_i(t) - v_i(t+1))$. Then, by the updating in (16), $v_i(t+1) = (v_i(t) - \Theta(i,i) g_i(t))/y_i(t)$, we have

$$
2g_i(t) + \frac{y_i(t) - \Theta(i,i) g_i(t)}{\Theta(i,i)} (v_i(t) - v_i(t+1))
$$

$$
= 2v_i(t) - \Theta(i,i) g_i(t) + \frac{2y_i(t)}{\Theta(i,i)} (v_i(t) - v_i(t+1))
$$

$$
= \frac{1 + y_i(t)}{\Theta(i,i)} \|v_i(t) - v_i(t+1)\|^2,
$$

where the third equality holds due to the condition $\|v_i\| = 1$. Then, the result holds from summing the above equation over $i = 1, \ldots, n$. \hfill \blacksquare

Now, we are ready to prove the result in Theorem 3.3.

**Proof of Theorem 3.1**: Assume Assumptions 4.1 and 4.2 hold. From Lemma 4.2 and the nonexistence of local optima, all non-optimal critical points are unstable fixed points. Recall that $h_{SM}$ is a diffeomorphism by Lemma 4.3 and non-optimal critical points of $h_{SM}(\cdot)$ are unstable fixed points by Lemma 4.2. It follows from the center-stable manifold theorem (Theorem III.5 of [45]) that the proposed algorithm escapes all non-optimal critical points almost surely under random initialization. By Lemma 4.3 and the fact that $\frac{1 + y_i(t)}{\Theta(i,i)}$ in (30) is always positive over iterations, the objective function value is strictly decreasing. Because the objective function value generated by the proposed algorithm is strictly decreasing and the objective value is lower bounded, the generated variables converge to the set of first-order critical points. Thus, the almost sure divergence from the non-optimal critical points and the convergence to critical points imply that $v_i(t) \in J_i$ in the updating $h_{SM}$ converges to corresponding column of global optimal solutions of (4) almost surely under random initialization.

Next, we show that the result of this theorem holds if Assumptions 4.1 and 4.2 are removed. If Assumption 4.1 does not hold, the indices of variables can be rearranged manually such that the indices in uncoupling set $S_{i,\text{par}(i)}$ are smaller than the indices in coupling set $S_{i,\text{ch}(i)}$ for each agent $i$. Hence, the above analysis still holds without Assumptions 4.1.

If Assumption 4.2 does not hold, in (26) of Lemma 4.1 for each variable $v_i, \ell \in S_{\text{ch}(i),\text{i}} \cap S_{i,\text{par}(i)}$, there must be a parent agent $j$ such that $l \in R_{j,\text{par}(j)}$ by algorithm design. Then, $v_i(t) = v_j(t) = v_l(t)$. Thus, the analysis in (26) analogously holds and the rest of theoretical deductive is true. \hfill \blacksquare

**B. Convergence analysis for asynchronous algorithm**

For distributed asynchronous algorithm 2, let $T^i$ be the set of times at which agent $i$ updates variable $V^i$. In addition, agent $i$ may not have access to the most recent value of other agents’ variables. To collect communicated information from neighbors, define a set $J^i$ as the union $J^i \cup R_{\text{ch}(i),i}$. Thus, define one possibly outdated variable of agent $i$, $V^i(t) \in \mathbb{R}^{P \times |J^i|}$, as

$$
V^i(t) = [v_{s_1}^i(t), \ldots, v_{s_{|J^i|}}^i(t)],
$$

where $\{s_1, \ldots, s_{|J^i|}\} = J^i \cup R_{\text{ch}(i),i}$. Let $\tau^i_j(t)$ is assumed to satisfy the condition (10). Recall that, if $s_j \in R_{j,\text{par}(j)}$, then $\tau^i_j(t) = t$. $V^i \in \mathbb{R}^{P \times |J^i|}$ collects time-delayed transmitted information from children and parents.

With additional assumption in Assumption 3.3 that each element of global coefficient matrix $M$ is only accessible to one agent, the relationship $M^i_{t,k} = M_{t,k}$ holds for all agent $i$. Then, the updating proposed in DAA is rewritten as a compact form as shown in the following lemma, where each column variable $v_j$ of global variable $V$ defined in (14) is expressed by outdated transmitted information.

**Lemma 4.6**: Under Assumption 3.3 each column variable $v_j$ generated by DAA with index $j \in R_{i,\text{par}(i)}$ is equivalent to

$$
v_j(t+1) = \text{normal}(v_j(t) - \Theta(j,j) h_j(V^i(t)), \ t \in T^i,
$$

$$
v_j(t+1) = v_j(t), \ t \notin T^i\tag{33}
$$

where $h_j(V^i(t)) = \left[\sum_{l \in R_{j,\text{par}(j)}} M_{j,l} v_l(t) + \sum_{l \in S_{j,\text{par}(j)}} M_{j,l} v_l(\tau^i_j(t)) + \sum_{l \in \text{ch}(i)} M_{j,l} v_l(\tau^i_j(t))\right]$. $\Theta(j,j) = \theta^i_j$ was defined in (11).
Proof: By (12) in DAA, agent \( \hat{i} \) updates local uncoupling variable \( v_j \) with index in \( j \in R_{i,\text{par}(i)} \) according to
\[
v_j(t+1) = \text{normal} \left( v_j(t) - \Theta_{i,j} \left[ \sum_{l \in R_{i,\text{par}(i)}} M_{j,l} v_l(t) + \sum_{\hat{i} \in \text{ch}(i)} \sum_{l \in J_{i,\hat{i}}} M_{j,l} v_{\hat{i}}(\tau_{\hat{i}}(t)) \right] \right)
\]
where the last equality holds because \( M_{j,l}^i = 0, \) for \( j \in R_{i,\text{par}(i)}, \) \( l \in S_{\hat{i}} \), by the assumption that each element of \( M \) is only accessible to one agent.

For the second term \( \sum_{l \in S_{i,\text{par}(i)}} M_{j,l} v_l^{\text{par}(i)}(\tau_{\hat{i}}(t)) \) of (34), similarly to the discussions of (26) in the synchronous case, we obtain
\[
\sum_{l \in S_{i,\text{par}(i)}} M_{j,l} v_l^{\text{par}(i)}(\tau_{\hat{i}}(t)) = \sum_{l \in S_{i,\text{par}(i)}} M_{j,l} v_l(\tau_{\hat{i}}(t)).
\]
(35)

Then, substituting (35) to (34), we have
\[
v_j(t+1) = \text{normal} \left( v_j(t) - \Theta_{i,j} \left[ \sum_{l \in R_{i,\text{par}(i)}} M_{j,l} v_l(t) + \sum_{\hat{i} \in \text{ch}(i)} \sum_{l \in R_{\hat{i}}} M_{j,l} v_{\hat{i}}(\tau_{\hat{i}}(t)) \right] \right),
\]
(36)
where \( h_j(V^\hat{i}(t)) = \left[ \sum_{l \in R_{i,\text{par}(i)}} M_{j,l} v_l(t) + \sum_{\hat{i} \in \text{ch}(i)} \sum_{l \in R_{\hat{i}}} M_{j,l} v_{\hat{i}}(\tau_{\hat{i}}(t)) \right], \) \( \Theta_{i,j} = \theta_j^i \) for \( j \in R_{i,\text{par}(i)} \). For \( t \neq T^\hat{i} \), \( v_j(t+1) = v_j(t) \) holds naturally.

Define the updating direction \( s_j \) \( (j \in R_{i,\text{par}(i)}) \) as,
\[
s_j(t) = \frac{1}{\Theta_{i,j}} (v_j(t+1) - v_j(t)).
\]
(37)
If \( t \in T^\hat{i} \),
\[
s_j(t) = \frac{1}{\Theta_{i,j}} \left( \text{normal}(v_j(t) - \Theta_{i,j} h_j(V^\hat{i}(t))) - v_j(t) \right),
\]
(38)
and if \( t \notin T^\hat{i} \), \( s_j(t) = 0 \).

In the following lemma, we present a vital descent property of local variable iteration, which will be used in the proof of Theorem 3.2.

Lemma 4.7: Suppose Assumption 3.3 holds. For any agent \( \hat{i} \) and time \( t \), we have
\[
s_j(t)'h_j(V^\hat{i}(t)) \leq -\|s_j(t)\|^2, \quad j \in R_{i,\text{par}(i)}.
\]
(39)
Proof: If \( t \notin T^\hat{i} \), the inequality (39) is true since both sides are zero. If \( t \in T^\hat{i} \), by the definition of \( s_j(t) \) in (37) and by Lemma 4.6, \( v_j(t+1) = (v_j(t) - \Theta_{i,j} h_j(V^\hat{i}(t))) + y_j(t) \) for \( j \in R_{i,\text{par}(i)} \), where \( y_j(t) = \|v_j(t) - \Theta_{i,j} h_j(V^\hat{i}(t))\| \), we have
\[
s_j(t)'h_j(V^\hat{i}(t)) = -\frac{1}{\Theta_{i,j}} (v_j(t) - v_j(t+1))' h_j(V^\hat{i}(t))
\]
\[
= -\frac{1}{\Theta_{i,j}} \left( h_j(V^\hat{i}(t)) + (v_j(t) - \Theta_{i,j} h_j(V^\hat{i}(t)))' (v_j(t) - v_j(t+1)) \right)
\]
\[
= -1 \Theta_{i,j} \left[ \frac{1}{\Theta_{i,j}} v_j(t)'(v_j(t) - v_j(t+1))
\right.
\]
\[
- \left. \frac{y_j(t)}{\Theta_{i,j}} v_j(t+1)'(v_j(t) - v_j(t+1)) \right]
\]
\[
= -1 \left( \frac{1}{\Theta_{i,j}} \|v_j(t+1) - v_j(t)\|^2 - (1 + y_j(t)) \right)
\]
\[
\leq -\|s_j(t)\|^2
\]
where the last inequality holds because \( y_j \) is non-negative.

Making use of Lemma 4.7, we discuss the relationship of global variable \( V \) and local variables \( V^\hat{i} \), and the gradient of objective function at the point \( V(t_k) \) when \( k \to \infty \) in Theorem 3.2. Before discussions, it should be noted that \( h_j(V^\hat{i}(t)) \) defined in Lemma 4.6 is exactly the gradient of global function \( f \) with respect to column variable \( v_j(t) \), where \( j \in R_{i,\text{par}(i)} \), so for convenience, we use \( h_j(V^\hat{i}(t)) \) in the following analysis. The proof follows the studies of gradient-like optimization algorithms in Proposition 5.1, section 7, [46].

Proof of Theorem 3.2 We follow the proof of Proposition 5.1 in [46]. By Assumption 3.1(2), the objective function of
satisfies \( \sum_{j=1}^{m} f_j \geq 0 \). In addition, with the analysis in Lemma 4.7 we have, for \( j \in \{1, \ldots, n\} \),

\[
\begin{align*}
& s_j(t) h_j(V^i(t)) \geq -\|s_j(t)\| \|h_j(V^i(t))\| \\
& - (1 + y_j(t)) \|s_j(t)\|^2 \geq -\|s_j(t)\| \|h_j(V^i(t))\| \\
& (1 + y_j(t)) \|s_j(t)\|^2 \leq \|s_j(t)\| \|h_j(V^i(t))\| \\
& \|s_j(t)\| \leq \frac{1}{1 + y_j(t)} \|h_j(V^i(t))\|,
\end{align*}
\]

where the last inequality holds because \( y_j \) is non-negative.

Then, there is a positive constant \( K_3 = 1 \) such that \( \|s_j(t)\| \leq K_3 \|h_j(V^i(t))\| \). What’s more, with the block-descent property in Lemma 4.7, the assumptions in Proposition 5.1 hold, where the product is replaced by inner product of vectors and the term \( |s_j(t)| \) is replaced by \( \|s_j(t)\| \). Then, by a similar analysis as Proposition 5.1 in [46], we obtain

\[
\lim_{t \to \infty} s_j(t) = 0,
\]

for each \( j \in \{1, \ldots, n\} \). In addition, by (37), we obtain

\[
\lim_{t \to \infty} \|V(t + 1) - V(t)\| = 0.
\]

Then, consider the boundedness of \( \|\tilde{v}^j(t) - v_j(t)\| \).

\[
\|\tilde{v}^j(t) - v_j(t)\| = \|\tilde{v}^j(\tau_j(t)) - v_j(\tau_j(t))\| \leq \Theta(\tau_j) \leq \Theta(\tau_j) \sum_{r=t-B}^{t-1} \|s_j(\tau)\|.
\]

With (40) and (42), we also obtain

\[
\lim_{t \to \infty} \|v^j(t) - v_j(t)\| = 0, \quad \forall j \in R_{i, \text{par}(i)}.
\]

In addition, for each \( j \in \{1, \ldots, n\} \), there is a unique agent \( \hat{i} \) such that \( j \in R_{i, \text{par}(i)} \) and the equation (43) holds.

Define \( V^* \) as a limit point of \( V(t) \) and \( \{\tau_k\} \) as a sequence such that \( \lim_{k \to \infty} V(\tau_k) = V^* \). Let \( \tau_k \) be such that \( |t_k - \tau_k| \leq B \) and \( \tau_k \in T^i \). Then, by equations (41) and (43), \( v_j(\tau_k) \) converges to \( v_j^* \), which is jth column of \( V^* \), and \( v^j(\tau_k) \) converges to \( v^j_\tau \) such that \( V^*(\tau_k) \) converges. Then, we have

\[
\lim_{k \to \infty} (|v^j_\tau(\tau_k) - \Theta(\tau_j) h_j(V^i(\tau_k))| - v^j_\tau(\tau_k)) \leq \lim_{k \to \infty} \Theta(\tau_j) s_j(\tau_k) = 0.
\]

Since it holds for each \( j \in \{1, \ldots, n\} \), we get the desired result.

V. SIMULATION

In this section, numerical tests and large-scale image segmentation application are presented to show the efficiency of the proposed distributed algorithms.

Example 1: We present one special sparse coupling numerical optimization problem, which has been investigated in [34], [37], [47] and of which the corresponding connected graph is one clique tree. More detailed information of clique trees can be found in [37]. Here we only introduce some brief concepts and focus on the discussions about the numerical convergence performance of proposed algorithms. For the coupling optimization (1), we assume that the dimension of global matrix variable \( X \) is \( n = 8 \), the number of local functions is \( N = 6 \), the corresponding dependent element indices set are \( C_1 = \{1, 3\} \), \( C_2 = \{1, 2, 4\} \), \( C_3 = \{4, 5\} \), \( C_4 = \{3, 4\} \), \( C_5 = \{3, 6, 7\} \), \( C_6 = \{3, 8\} \). By the clique tree transformations in [47], the corresponding clique tree owns five agents, shown in Fig. 3. It shows that the number of agents in corresponding problem (1) is \( m = 5 \), which implies that one agent has multiple local functions. The local functions assigned to ith agent are denoted by a function set \( \phi_i \). Then, the function sets of the clique tree are \( \phi_1 = \{f_2\} \), \( \phi_2 = \{f_1, f_4\} \), \( \phi_3 = \{f_3\} \), \( \phi_4 = \{f_5\} \), \( \phi_5 = \{f_6\} \). In addition, the ordered index sets are \( J_1 = \{1, 2, 4\} \), \( J_2 = \{1, 3, 4\} \), \( J_3 = \{4, 5\} \), \( J_4 = \{3, 6, 7\} \), \( J_5 = \{3, 8\} \). More specifically, we provide the decomposed diagram of sparse coefficient matrix \( M \) over five different agents as following. Elements with different colors are assigned to different agents.

Since the diagonal elements will not influence the optimization result, they are assigned to any agent without effect. Let \( f^* \) be the optimal function value of the optimization problem (1), which is solved by the solver YALMIP when the dimension \( n \) is not too large.

1) We use the proposed distributed synchronous, asynchronous algorithms and the centralized algorithm SDPLR (20), which are all coded by MATLAB, to solve the sparse optimization problem. The simulation results are shown in Figs. 4(a)-(c). The original global variable value is \( X(k) = V(k) \). In Fig. 4(a) the proposed algorithms and SD-PLR all converge to the optimal function value \( f^* \), which is calculated by the solver YALMIP. It is observed that distributed algorithms converge much faster than the SDPLR algorithm for the MAXCUT problem. In Fig. 4(b) the trajectories of \( ||X(k+1) - X(k)||_F \), where the trajectories \{X(k)\} are generated by the proposed algorithms DSA and DAA respectively, are shown to converge to zeros. It shows that

\[
\begin{bmatrix}
W_{11} & W_{12} & W_{13} & W_{14} & 0 & 0 & 0 & 0 \\
W_{21} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
W_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
W_{23} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
W_{33} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
W_{44} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
W_{55} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
W_{66} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
W_{77} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
W_{88} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]
\[
\|\nabla f(k)\|_F
\]

to critical points. In addition, by the numerical experiments, for most cases, we have observed that the asynchronous algorithm converges faster than the synchronous algorithm. In addition, in the next simulation, we provide quantitative comparisons of convergence rates between DAA and DSA.

2) In order to compare the performance of the proposed distributed algorithms with the inspired centralized algorithm, which is proposed in [18], we make use of MPICH distributed model, which is a high-performance message passing interface, to develop a multi-processors environment on one computer with a Core(TM) I5-8250U CPU, 1.6GHz. Both centralized algorithm and distributed algorithms are coded by C language. For the distributed algorithms, we use five processes to deal with the optimization problem.

We provide two experiments with different dimensions and collect the number of iterations and executive time of different algorithms. In each experiment, the stop criterion of iterations reaches an expected error between the function value \(f(k)\) and optimal value \(f^*\). In addition, we use \(sn\) to denote the number of shared variables over the multi-agent network, e.g., the \(sn\) of network shown in Fig. 3 is 5. The executed time comparisons of centralized algorithm and distributed algorithms are listed in following table [1].

By the comparative test, the distributed synchronous and asynchronous algorithms both converge faster than the centralized Mixing algorithm. As the dimension of problem increases, the role of distributed design is more important, especially when communication between different agents is sparse. In addition, by the simulation, we observe that distributed asynchronous algorithm often converges faster than distributed synchronous algorithm. It should be pointed out that although communication time-delay will not make asynchronous algorithm diverge, coordinating the trade-off between communication and computation may further improve the convergence performance of distributed asynchronous algorithm in practice, which is one future research direction of our work.

![Convergent trajectories generated by proposed algorithms and SDPLR](image1.png)

![Convergent trajectories generated by proposed algorithms and SDPLR](image2.png)

![Convergent trajectories generated by proposed algorithms and SDPLR](image3.png)

Fig. 4. Convergent trajectories generated by proposed algorithms and SDPLR

The trajectories of \(X(k + 1)\) by DSA, DAA and SDPLR

\[
\|X(k + 1) - X(k)\|_F
\]

The trajectories of \(\|X(k + 1) - X(k)\|_F\) along time

\[
\|\nabla f(k)\|_F
\]

The trajectories of gradient of \(f\) along time

**TABLE I**

| Dimension | Algorithm | \(sn\) | Iterations | Time (ms) | Error |
|-----------|-----------|--------|------------|-----------|-------|
| 8         | Mixing    | 0      | 291        | 20        | 0.00023 |
|           | DSA       | 5      | 150        | 5         | 0.00023 |
|           | DAA       | 5      | 150        | 4         | 0.00023 |
| 18        | Mixing    | 0      | 600        | 41        | 0.0058  |
|           | DSA       | 8      | 390        | 16        | 0.0058  |
|           | DAA       | 8      | 301        | 10        | 0.0058  |

**Example 2:** We apply the proposed distributed asynchronous algorithm to solving MAXCUT problems from image segmentation over a multi-agent system, as shown in Fig. 5. There is one edge between agents if there exists an intersection between image pixels of different local images. Hence, there exist several parents for one agent, which is different from the first example, where each agent has only one parent. In this example, we will show that the proposed distributed algorithm efficiently achieves image segmentation.

There have been some works applying general graph cut algorithms to image segmentation [48]-[51]. For image segmentation, we need to create a graph representation of the

\[
v_i \langle v_i, g_i \rangle^2, \]
image. One algorithm is to consider assigning each pixel of the image as a node and using a four-connected neighborhood to create the edges [50], as shown in Fig. 6. We here only utilize the intensity components of the RGB of all pixels to provide one simple connected matrix $M$, whose $(i,j)$th element related to nodes $(i,j)$ is defined by the following equation [50]:

$$M_{i,j} = \max(\langle \Vert rgb(i) - rgb(j) \Vert_2 > t \rangle - 1) \langle \Vert rgb(i) - rgb(j) \Vert_2, 0 \rangle,$$

where $rgb(i)$ is the intensity vector of RGB of the $i$th pixel, $t$ is adjustable threshold value. The output of operator $\max(a, 0)$ is the bigger one of $a$ and 0. Then, the generated matrix $M$ is a typical large-scale sparse matrix. For some algorithms which add seeds to different regions, the only change is the development of matrix elements. We only use the simplest RGB information between different pixels to segment image. However, it should be noted that the proposed algorithms are applicable for general MAXCUT problems [1] that include more involved development of coefficient matrix elements. In some intelligent algorithms, the graph cut problem is often used as an important pretreatment [52]. Therefore, the large-scale sparse graph cut problem is vital in image segmentation.

We apply the proposed distributed asynchronous algorithm on images of the Berkeley database [53]. We have computed the results for three images (Airplane, Church, Bird) in Figure 7. It is seen that the proposed distributed algorithm achieves image segmentation efficiently. While the existing centralized algorithms can not deal with image segmentation because of the large dimension of image data.

![Airplane segmentation](image1.png)
(a) Airplane segmentation

![Church segmentation](image2.png)
(b) Church segmentation

![Bird segmentation](image3.png)
(c) Bird segmentation

Fig. 7. Segmentation results of images from Berkeley database

VI. CONCLUSION

This paper has studied distributed synchronous and asynchronous algorithms for solving large-scale SDP with diagonal constraints by making use of the inherent sparsity of programming and low-rank property of solutions. Each agent updates its local variables by local information and communicating messages over the underlying topology. To handle the communication delays in networks, one distributed asynchronous algorithm is proposed without global clocks. Although the transformed optimization problem is non-convex, variables of distributed synchronous and asynchronous algorithms eventually converge to optimal solutions and critical points of SDP, respectively. The efficiency of proposed distributed algorithms is verified by the numerical simulations.

Future work involves developing and analyzing communication-efficient distributed algorithms, which balance the computational and communication cost of different agents, for SDP with diagonal constraints. The objective SDP problem of this paper has diagonal constraints. In future, we will further attempt to extend the distributed algorithms to more general semi-definite programs with linear constraints.
[48] F. Yi and I. Moon, “Image segmentation: A survey of graph-cut methods,” in 2012 International Conference on Systems and Informatics (ICSAI2012), Yantai, 2012, pp. 1936–1941.

[49] S. Vicente, V. Kolmogorov, and C. Rother, “Graph cut based image segmentation with connectivity priors,” in 2008 IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, AK, 2008, pp. 1–8.

[50] S. de Sousa, Y. Haxhimusa, and W. Kropatsch, “Estimation of distribution algorithm for the Max-Cut problem,” in Graph-Based Representations in Pattern Recognition (GbRPR 2013), vol. 7877, Berlin, Heidelberg, 2013, pp. 244–253.

[51] Y. Boykov and G. Funka-Lea, “Graph cuts and efficient ND image segmentation,” International Journal of Computer Vision, vol. 70, no. 2, pp. 109–131, 2006.

[52] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, “Contour detection and hierarchical image segmentation,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 33, no. 5, pp. 898–916, 2011.

[53] D. Martin, C. Fowlkes, D. Tal, and J. Malik, “A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics,” in Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001, vol. 2, British Columbia, Canada, 2001, pp. 416–423.