Jiang, Wei; Charalambous, Themistoklis

Distributed Alternating Direction Method of Multipliers using Finite-Time Exact Ratio Consensus in Digraphs

Published in:
2021 European Control Conference, ECC 2021

DOI:
10.23919/ECC54610.2021.9654976

Published: 01/01/2021

Please cite the original version:
Distributed Alternating Direction Method of Multipliers using Finite-Time Exact Ratio Consensus in Digraphs

Wei Jiang and Themistoklis Charalambous

Abstract—In this work, we consider the distributed optimization problem in which each node has its own convex cost function and can communicate directly only with its neighbors, as determined by a directed communication topology (directed graph or digraph). First, we reformulate the optimization problem so that Alternating Direction Method of Multipliers (ADMM) can be utilized. Then, we propose an algorithm, herein called Distributed Alternating Direction Method of Multipliers using Finite-Time Exact Ratio Consensus (D-ADMM-FTERC), to solve the multi-node convex optimization problem, in which every node performs iterative computations and exchanges information with its neighbors. At every iteration of D-ADMM-FTERC, each node solves a local convex optimization problem for the one of the primal variables and utilizes a finite-time exact optimal consensus protocol to compute the optimal value of the other variable, since the cost function for the second primal variable is not decomposable. If the individual cost functions are convex and not-necessarily differentiable, the proposed algorithm converges at a rate of $O(1/k)$, where $k$ is the iteration counter. The efficacy of D-ADMM-FTERC is demonstrated via a distributed least square optimization example. Additionally, comparisons with other state-of-the-art algorithms are provided on both small and large scale systems showing the superior precision and time-efficient performance of D-ADMM-FTERC.

Index Terms—Distributed optimization, directed graphs, ADMM, consensus.

I. INTRODUCTION

The main objective is the solution of an additive cost optimization problem over a digraph in a distributed fashion, where each individual cost is known solely to the node; this type of problems is often referred to as distributed optimization problem and a wide variety of engineering problems (e.g., wireless sensor networks [1] and machine learning [2], [3]) fall within this framework. For this reason, even though such problems were targeted already in the 80’s [4], [5], the field of distributed optimization has attracted a lot of attention by the research community again recently; see, for example, [6]–[11] and references therein.

There are two main research strands for solving distributed optimization methods in the literature: (i) primal and (ii) dual-based optimization methods. Our work falls in the strand of dual-based optimization methods, and more specifically on distributed approaches for realizing the ADMM. In that direction, there are two main communication topologies considered: (i) master-workers communication topology and (ii) multi-node communication topology. When the ADMM has a master-worker communication topology, the worker nodes optimize their local objectives and communicate their local variables to the master node which updates the global optimization variable and send it back to the workers. When the ADMM has no master node, the optimization problem is solved over a network of nodes. Here in, we focus on the ADMM realized on multi-node communication topologies.

There have been several ADMM algorithms proposed for the case which the multi-node communication topology assumes that every communication link is bidirectional, thus forming a communication topology represented by an undirected graph; see, for example, [9], [12], [13]. In the case for which some communication links are not necessarily bidirectional, these approaches fail to converge to the optimal solution. Distributed ADMM approaches for digraphs are very limited. The first distributed ADMM approach for directed graphs with convergence guarantees [14], and the inspiration for this work, proposes a consensus-based approach to compute one of the primal variables of ADMM for digraphs. Specifically, at every step, while one of the primal variables and the Lagrange multiplier are computed at the node itself, the other primal variable is approximated by running an asymptotic consensus algorithm for a finite number of steps and, as a consequence, an approximate solution at every optimization step is obtained.

In this work, we make use of a distributed protocol proposed in [15], [16], where each node in a digraph can observe and store the evolution of its own values over a minimal number of steps and, hence, compute the average consensus value immediately from the successive observations. Therefore, we propose a distributed ADMM algorithm that solves exactly the multi-node convex optimization problem in digraphs. At every iteration of the algorithm, each node solves a local convex optimization problem for one of the primal variables and utilizes a finite-time consensus protocol to compute the exact optimal of the other primal variable.

The remainder of the paper is organized as follows. In Section II we provide necessary notation and background knowledge for the development of our results. In Section III the problem to be solved is formulated, and in Section IV our proposed algorithm is explained, with its convergence analysis provided in Section V. Illustrative examples are presented in Section VI. Finally, in Section VII we present concluding remarks and discuss possible future directions.
II. NOTATION AND PRELIMINARIES

A. Notation

The set of real (integer) numbers is denoted by \( \mathbb{R} \) (\( \mathbb{Z} \)) and the set of positive numbers (integers) is denoted by \( \mathbb{R}_+ \) (\( \mathbb{Z}_+ \)). \( \mathbb{R}^n \) denotes the non-negative orthant of the \( n \)-dimensional real space \( \mathbb{R}^n \). Vectors are denoted by small letters whereas matrices are denoted by capital letters. \( A^T \) denotes the transpose of matrix \( A \). The \( i \)th component of a vector \( x \) is denoted by \( x_i \), and the notation \( x \geq y \) implies that \( x_i \geq y_i \) for all components \( i \). For \( A \in \mathbb{R}^{n \times n} \), \( a_{ij} \) denotes the entry in row \( i \) and column \( j \). We denote the all-ones vector by \( \mathbf{1} \) and the identity matrix (of appropriate dimensions) by \( I \). We also denote by \( e_j^r = [0, \ldots, 0, 1_{jn}, 0, \ldots, 0] \in \mathbb{R}^{1 \times n} \), where the single “1” entry is at the \( j \)th position. \( |A| \) is the element-wise absolute value of matrix \( A \) (i.e., \( |A| \equiv \sum |a_{ij}| \)), \( A \leq B \) (\( A < B \)) is the (strict) element-wise inequality between matrices \( A \) and \( B \). A matrix whose elements are nonnegative, called nonnegative matrix, is denoted by \( A \geq 0 \) and a matrix whose elements are positive, called positive matrix, is denoted by \( A > 0 \). \( \| \cdot \| \) denotes the 2-norm.

In multi-component systems with fixed communication links (edges), the exchange of information between components (nodes) can be conveniently captured by a directed graph (digraph) \( \mathcal{G}(\mathcal{V}, \mathcal{E}) \) of order \( n \) \((n \geq 2)\), where \( \mathcal{V} = \{v_1, v_2, \ldots, v_n\} \) is the set of nodes and \( \mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} \) is the set of edges. A directed edge from node \( v_j \) to node \( v_i \) is denoted by \( e_{ji} = (v_j, v_i) \in \mathcal{E} \) and represents a communication link that allows node \( v_j \) to receive information from node \( v_i \). A graph is said to be undirected if and only if \( e_{ji} \in \mathcal{E} \) implies \( e_{ij} \in \mathcal{E} \). A digraph is called strongly connected if there exists a path from each vertex \( v_i \) of the graph to each vertex \( v_j \) \((v_j \neq v_i)\). In other words, for any \( v_j, v_i \in \mathcal{V} \), \( v_j \neq v_i \), one can find a sequence of nodes \( v_1, v_2, \ldots, v_m = v_j \) such that link \((v_{s+1}, v_s) \in \mathcal{E} \) for all \( s = 1, 2, \ldots, m-1 \). The diameter \( D \) of a graph is the longest shortest path between any two nodes in the network.

All nodes that can transmit information to node \( v_j \) directly are said to be in-neighbors of node \( v_j \) and belong to the set \( \mathcal{N}_-^j = \{v_i \in \mathcal{V} \mid \exists e_{ji} \in \mathcal{E}\} \). The cardinality of \( \mathcal{N}_-^j \), is called the in-degree of \( v_j \) and is denoted by \( D_-^j = \left| \mathcal{N}_-^j \right| \). The nodes that receive information from node \( v_j \) belong to the set of out-neighbors of node \( v_j \), denoted by \( \mathcal{N}_+^j = \{v_i \in \mathcal{V} \mid \exists e_{ij} \in \mathcal{E}\} \). The cardinality of \( \mathcal{N}_+^j \), is called the out-degree of \( v_j \) and is denoted by \( D_+^j = \left| \mathcal{N}_+^j \right| \).

B. Average Consensus

In the type of algorithms we consider, we associate a positive weight \( p_{ji} \) for each edge \( e_{ji} \in \mathcal{E} \cup \{v_j, v_j\} \mid v_j \in \mathcal{V}\}. The nonnegative matrix \( P = [p_{ji}] \in \mathbb{R}^{n \times n} \) (with \( p_{ji} \) as the entry at the \( j \)th row, \( i \)th column position) is a weighted adjacency matrix (also referred to as weight matrix) that has zero entries at locations that do not correspond to directed edges (or self-edges) in the graph. In other words, apart from the main diagonal, the zero-nonzero structure of the adjacency matrix \( P \) matches exactly the given set of links in the graph. In a synchronous setting, each node \( v_j \) updates and sends its information to its neighbors at discrete times \( T_0, T_1, T_2, \ldots \). We index nodes’ information states and any other information at time \( T_t \) by \( t \). Hence, we use \( w_{ji}^t = w_{ji}[t] \in \mathbb{R} \) to denote the information state of node \( v_j \) at time \( T_t \). Note that \( w_{ji}^t \) denotes (is equivalent to) \( w_{ji}[t] \).

Each node updates its information state \( w_{ji}^t \) by combining the available information received by its neighbors \( w_{ji}^t \) \((v_j \in \mathcal{N}_-^j)\) using the positive weights \( p_{ji} \), that capture the weight of the information inflow from node \( v_j \) to node \( v_j \) at time \( t \). In this work, we assume that each node \( v_j \) can choose its self-weight and the weights on its out-going links \( \mathcal{N}_+^j \) only. Hence, in its general form, each node updates its information state according to the following relation:

\[
w_{ji}^{t+1} = p_{ji}w_{ji}^t + \sum_{v_i \in \mathcal{N}_-^j} p_{ji}w_{ji}^t, \quad k \geq 0,
\]

where \( w_{ji}^0 \in \mathbb{R} \) is the initial state of node \( v_j \). If we let \( w_{ji} = (w_{j1}^t w_{j2}^t \ldots w_{jn}^t) \) and \( P = [p_{ji}] \in \mathbb{R}^{n \times n} \), then (1) can be written in matrix form as

\[
w_{ji}^{t+1} = P w_{ji}^t,
\]

where \( w_{ji}^0 = (w_{j1}^0 w_{j2}^0 \ldots w_{jn}^0) \vdash w_{ji}^0 \). We say that the nodes asymptotically reach average consensus if

\[
\lim_{t \to \infty} w_{ji}^t = \frac{\sum_{v_i \in \mathcal{V}} w_{ji}^0}{n}, \quad \forall v_j \in \mathcal{V}.
\]

The necessary and sufficient conditions for (2) to reach average consensus are the following: (a) \( P \) has a simple eigenvalue at one with left eigenvector \( 1^T \) and right eigenvector \( 1 \), and (b) all other eigenvalues of \( P \) have magnitude less than 1. If \( P \geq 0 \) (as in our case), the necessary and sufficient condition is that \( P \) is a primitive doubly stochastic matrix. In an undirected graph, assume each node knows \( n \) (or an upper bound \( n' \)) and the graph is connected, each node \( v_j \) can distributively choose the weights on its outgoing links to be \( \frac{1}{n'} \) and set its diagonal to be \( 1 - \frac{D_j^+}{n'} \) (where \( D_j^+ = D_j^+ \vdash D_j \)), so that the resulting weight matrix \( P \) is primitive doubly stochastic. However, in a digraph, this choice does not necessarily lead to a doubly stochastic weight matrix.

C. Ratio Consensus

In [17], an algorithm is suggested that solves the average consensus problem in a directed graph in which each node \( v_j \) distributively sets the weights on its self-link and outgoing-links to be \( \frac{1}{1+D_j} \), so that the resulting weight matrix \( P \) is column stochastic, but not necessarily row stochastic. Average consensus is reached by using this weight matrix to run two iterations with appropriately chosen initial conditions. The algorithm is stated below for a specific choice of weights on each link that assumes that each node knows its out-degree; note, however, that the algorithm works for any set of weights that adhere to the graph structure and form a primitive column stochastic weight matrix.

**Proposition 1 ([17]):** Consider a strongly connected digraph \( \mathcal{G}(\mathcal{V}, \mathcal{E}) \). Let \( y_j^t \) and \( x_j^t \) (for all \( v_j \in \mathcal{V} \) and \( t = \)
0, 1, 2, . . .) be the result of the iterations
\[ y_j^{t+1} = p_{jj} y_j^t + \sum_{v_i \in N_j^+} \rho_{ji} y_i^t, \quad (3a) \]
\[ x_j^{t+1} = p_{jj} x_j^t + \sum_{v_i \in N_j^+} \rho_{ji} x_i^t, \quad (3b) \]
where \( \rho_{ij} = \frac{1}{1+D_{ji}} \) for \( v_i \in N_j^+ \cup \{ v_j \} \) (zeros otherwise), and the initial conditions are \( y_j^0 = y_0 \) and \( x_j^0 = 1 \). Then, the solution to the average consensus problem can be asymptotically obtained as \( \lim_{t \to \infty} \mu_j^t = \frac{\sum_{v_i \in V} y_i^t}{|V|} \), \( \forall v_j \in V \), where \( \mu_j^t = \frac{y_j^t}{x_j^t} \).

**Remark 1:** Proposition [1] proposes a decentralised algorithm with which the exact average is asymptotically reached, even if the directed graph is not balanced.

### D. Finite-Time Exact Ratio Consensus (FTEC)

In what follows, we present a distributed protocol proposed in [15], [16] with which each node can compute, based on its own local observations and after a minimal number of steps, the exact average. This protocol is based on the algorithm in Proposition [1] with which every node can compute \( \mu_j^\infty = \lim_{t \to \infty} \mu_j^t \) in a minimum number of steps.

**Definition 1:** (Minimal polynomial of a matrix pair) The minimal polynomial associated with the matrix pair \([P, e_j] \) is denoted by \( q_j(s) = s^{M_j+1} + \sum_{i=0}^{M_j} \alpha_i(s) s^i \). Let \( w_j^t \) be the result of the iterations
\[ W_j(z) = \frac{\sum_{i=0}^{M_j+1} \alpha_i(z) \sum_{\ell=0}^{t-1} w_j^\ell z^{-\ell}}{q_j(z)}, \]
where \( q_j(z) \) is the minimal polynomial of \([P, e_j] \). If the network is strongly connected, \( q_j(z) \) does not have any unstable poles apart from one at 1; we can then define the following polynomial:
\[ p_j(z) = \frac{q_j(z)}{z-1} = \sum_{i=0}^{M_j} \alpha_i(z) z^i. \]

The application of the final value theorem [18], [19] yields:
\[ \phi_y(j) = \lim_{t \to \infty} y_j^t = \lim_{z \to 1} (z-1) Y_j(z) = \frac{y_j^\infty \beta_j}{1 \beta_j}, \quad (7a) \]
\[ \phi_x(j) = \lim_{t \to \infty} x_j^t = \lim_{z \to 1} (z-1) X_j(z) = \frac{x_j^\infty \beta_j}{1 \beta_j}, \quad (7b) \]
where \( y_j^\infty = (y_j^0, y_j^1, \ldots, y_j^{M_j}) \) and \( x_j^\infty = (x_j^0, x_j^1, \ldots, x_j^{M_j}) \) and \( \beta_j \) is the vector of coefficients of the polynomial \( p_j(z) \).

Consider the vectors of \( 2t+1 \) successive discrete-time values at node \( v_j \), given by
\[ y_{2t}^j = (y_j^0, y_j^1, \ldots, y_j^{2t}), \]
\[ x_{2t}^j = (x_j^0, x_j^1, \ldots, x_j^{2t}), \]
for the two iterations \( y_j^t \) and \( x_j^t \) at node \( v_j \) (as given in iterations \( 3a \) and \( 3b \)), respectively. Let us define their associated Hankel matrices:
\[ \Gamma\{y_{2t}^j\} = \begin{bmatrix} y_j^0 & y_j^1 & \cdots & y_j^{2t} \\ y_j^1 & y_j^2 & \cdots & y_j^{2t+1} \\ \vdots & \vdots & \ddots & \vdots \\ y_j^t & y_j^{t+1} & \cdots & y_j^{2t} \end{bmatrix}, \]
\[ \Gamma\{x_{2t}^j\} = \begin{bmatrix} x_j^0 & x_j^1 & \cdots & x_j^{2t} \\ x_j^1 & x_j^2 & \cdots & x_j^{2t+1} \\ \vdots & \vdots & \ddots & \vdots \\ x_j^t & x_j^{t+1} & \cdots & x_j^{2t} \end{bmatrix}. \]

It has been shown in [19] that \( y_j^t \) can be computed as the kernel of the first defective Hankel matrices \( \Gamma\{y_{2t}^j\} \) and \( \Gamma\{x_{2t}^j\} \) for arbitrary initial conditions \( y_0 \) and \( x_0 \) (i.e., \( \beta_j \) can be calculated as the normalized kernel \( \beta_j = \left[ \beta_j^0 \beta_j^1 \ldots \beta_j^{x_{M_j-1}} \right] ^{-1} \) of the first defective Hankel matrix \( \Gamma\{y_{2t}^j\} \), except a set of initial conditions with Lebesgue measure zero.

Next, we provide Theorem [1] in which it is stated that the exact average \( \mu \) can be distributively obtained in finite-time in strongly connected digraphs.

**Theorem 1 ([15]):** Consider a strongly connected graph \( \mathcal{G}(V, E) \). Let \( y_j^t \) and \( x_j^t \) for all \( v_j \in V \) and \( t = 0, 1, 2, \ldots \) be the result of the iterations \( 3a \) and \( 3b \), where \( P = [p_{ij}] \in \mathbb{R}^{n \times n} \) is any set of weights that adhere to the graph structure and form a primitive column stochastic weight matrix. Then, the solution to the average consensus can be distributively obtained in finite-time at each node \( v_j \), by computing
\[ \mu_j = \lim_{t \to \infty} \frac{y_j^t}{x_j^t} = \frac{\phi_y(j)}{\phi_x(j)} = \frac{y_j^\infty \beta_j}{x_j^\infty \beta_j}, \quad (8) \]
where \( \phi_y(j) \) and \( \phi_x(j) \) are given by equations \( 7a \) and \( 7b \), respectively and \( \beta_j \) is the vector of coefficients, as defined in \( 6 \).

Theorem [1] states that the average consensus in a strongly connected digraph can be computed by the ratio of the final values computed for each of the iterations \( 3a \) with initial condition \( y^0 = y_0 \) and iteration \( 3b \) with initial condition \( x^0 = 1 \). Note that \( x^0 = 1 \) does not belong into the Lebesgue measure zero set of matrix \( P \) as defined in Proposition [1].
E. max-consensus algorithm

The max-consensus algorithm is a simple algorithm for computing the maximum value in a distributed fashion [20]. For any node \( v_j \in \mathcal{V} \), the update rule is as follows:

\[
x_j^{k+1} = \max_{v_i \in \mathcal{N}_j^{-} \cup \{v_j\}} \{x_i^k\}.
\]

(9)

It has been shown (see, e.g., [21], [22]) that this algorithm converges to the maximum value among all nodes in a finite number of steps \( s \), \( s \leq D \).

F. Standard ADMM Algorithm

The standard ADMM algorithm solves the following problem

\[
\min f(x) + g(z),
\]

s.t. \( Ax + Bz = c \),

(10)

for variables \( x \in \mathbb{R}^p, z \in \mathbb{R}^m \) with matrices \( A \in \mathbb{R}^{q \times p}, B \in \mathbb{R}^{r \times m} \) and vector \( c \in \mathbb{R}^q \). Note that \( p \in \mathbb{N} \) and \( m \in \mathbb{N} \) represent the dimensions of prime variables. The augmented Lagrangian is

\[
L_\rho(x, z, \lambda) = f(x) + g(z) + \lambda^T (Ax + Bz - c) + \frac{\rho}{2} \|Ax + Bz - c\|^2,
\]

(11)

where \( \lambda \) is the Lagrange multiplier and \( \rho \) is a positive scalar. In ADMM, the primary variables \( x, z \) and the Lagrange multipliers \( \lambda \) are updated as follows: starting from some initial vector \( [x^0, z^0, \lambda^0]^T \), at each optimization iteration \( k \),

\[
x^{k+1} = \arg\min_x L_\rho(x, z^k, \lambda^k),
\]

(12)

\[
z^{k+1} = \arg\min_z L_\rho(x^{k+1}, z, \lambda^k),
\]

(13)

\[
\lambda^{k+1} = \lambda^k + \rho(Ax^{k+1} + Bz^{k+1} - c).
\]

(14)

The step-size in the Lagrange multiplier update is the same as the augmented Lagrangian function parameter \( \rho \).

III. PROBLEM FORMULATION

In this work, we consider a strongly connected digraph \( \mathcal{G}(\mathcal{V}, \mathcal{E}) \) in which each node \( v_j \in \mathcal{V} \) is endowed with a scalar cost function \( f_i : \mathbb{R}^p \to \mathbb{R} \) assumed to be known to the node only. We assume that each node \( v_j \) knows the number of its out-going links, \( \mathcal{D}_j^+ \), and has access to local information only via its communication with its neighboring nodes, \( \mathcal{N}_j^- \). The only global information available to all the nodes in the network is given in Assumption 1.

**Assumption 1**: Each node \( v_j \in \mathcal{V} \) knows an upper bound on the number of nodes in the network \( n' \) (i.e., \( n' \geq n \)).

While Assumption 1 is limiting, there exist distributed methods for computing the size of the network; see, for example, [23].

The problem is to design a discrete-time coordination algorithm that allows every node \( v_j \) in a digraph to distributively solve the following optimization problem:

\[
\arg\min_{x \in \mathbb{R}^p} \sum_{i=1}^{n} f_i(x),
\]

(15)

where \( x \in \mathbb{R}^p \) is a global optimization variable (or a common decision variable). In order to distributively solve the previous problem and to enjoy the structure ADMM scheme at the same time, a separate decision variable \( x_i \) for node \( v_i \) is introduced and the constraint \( x_i = x_j \) is imposed to guarantee that the node decision variables are equal. In other words, problem (15) is reformulated as

\[
\min \sum_{i=1}^{n} f_i(x_i),
\]

s.t. \( x_i = x_j, \forall i, \forall j, v_j \in \mathcal{V} \).

Define a closed nonempty convex set \( C \) as

\[
C = \left\{ [x_1^T, x_2^T, \ldots, x_n^T]^T \in \mathbb{R}^{np} : x_i = x_j \right\}.
\]

(17)

By denoting \( X := [x_1^T, x_2^T, \ldots, x_n^T]^T \) and making variable \( z \in \mathbb{R}^{np} \) as a copy of vector \( X \), problem (16) becomes

\[
\min \sum_{i=1}^{n} f_i(x_i),
\]

s.t. \( X - z = 0 \).

(18)

Then, take \( g \) as the indicator function of set \( C \), and define \( g(z) \) as

\[
g(z) = \begin{cases} 0, & \text{if } z \in C, \\ \infty, & \text{otherwise.} \end{cases}
\]

(19)

Finally, problem (18) is transformed to

\[
\min \sum_{i=1}^{n} f_i(x_i) + g(z),
\]

s.t. \( X - z = 0 \).

(20)

For notational convenience, denote \( F(X) := \sum_{i=1}^{n} f_i(x_i) \). Thus, denote the Lagrangian function as

\[
L(X, z, \lambda) = F(X) + g(z) + \lambda^T (X - z),
\]

(21)

where \( \lambda \in \mathbb{R}^{np} \) is the Lagrange multiplier associated with the constraint \( X - z = 0 \). Then, the following standard assumptions are required for the optimization problem.

**Assumption 2**: Each cost function \( f_i : \mathbb{R}^p \to \mathbb{R} \cup \{+\infty\} \) is closed, proper and convex.

**Assumption 3**: The Lagrangian \( L(X, z, \lambda) \) has a saddle point, i.e., there exists a solution \((X^*, z^*, \lambda^*)\), for which

\[
L(X^*, z^*, \lambda^*) \leq L(X, z^*, \lambda^*) \leq L(X, z^*, \lambda^*),
\]

(22)

holds for all \( X \in \mathbb{R}^{np}, z \in \mathbb{R}^{np} \) and \( \lambda \in \mathbb{R}^{np} \).

Assumption 2 allows \( f_i \) to be non-differentiable [2]. By Assumptions 2 and based on the definition of \( g(z) \) in (19), \( L(X, z, \lambda^*) \) is convex in \((X, z)\) and \((X^*, z^*)\) is a solution to problem (20) [2, 12].

This step is quite standard in distributed optimization.
The finite-time consensus algorithm is terminated after $2n'$ iterations in the first step of the ADMM optimization ($n'$ is an upper bound of $n$ known to all nodes). During the first optimization step, each node $v_j$ computes $M_j$. During the second step, they run a max-consensus algorithm (which converges in $s$ ($s \leq D$) iterations) and they determine $M_{\text{max}}$. The second optimization step is terminated after $n'$ iterations. Thereafter, each optimization step is changed to $t_{\text{max}} := M_{\text{max}} + 1$ iterations.

IV. D-ADMM-FTERC ALGORITHM

At iteration $k$, the corresponding augmented Lagrangian of optimization problem (20) is written as

$$L_k(x_i, z_i, \lambda_i)$$

$$= \sum_{i=1}^{n} \left( f_i(x_i) + g(z_i) + \lambda_i^0(x_i - z_i) + \frac{\rho}{2} \| x_i - z_i \|^2 \right) + g(z_i),$$

where $z_i \in \mathbb{R}^p$ is the $i$-th element of vector $z$. By ignoring terms which are independent of the minimization variables (i.e., $x_i, z$), for each node $v_i$, the standard ADMM updates (12)-(14) change to the following format:

$$x_i^{k+1} = \arg\min_{x_i} f_i(x_i) + \lambda_i^k(x_i - z_i) + \frac{\rho}{2} \| x_i - z_i \|^2,$$

$$z_i^{k+1} = \arg\min_{z_i} g(z) + \lambda_i^k(x_i^{k+1} - z_i) + \frac{\rho}{2} \| x_i^{k+1} - z_i \|^2,$$

$$\lambda_i^{k+1} = \lambda_i^k + \rho (x_i^{k+1} - z_i^{k+1}),$$

where the last term in (25) comes from the identity $2a^T b + b^2 = (a + b)^2 - a^2$ with $a = \lambda_i^k / \rho$ and $b = x_i^{k+1} - z_i$.

Update (24) for $x_i^{k+1}$ can be solved by a classical method, e.g., the proximity operator [2, Section 4]. Update (26) for the dual variable $\lambda_i^{k+1}$ can be implemented trivially by node $v_i$. Note that both updates can be done independently by node $v_i$. Since $g$ is the indicator function of the closed nonempty convex set $C$, update (25) for $z_i^{k+1}$ becomes

$$z_i^{k+1} = \Pi_C (x_i^{k+1} + \lambda_i^k / \rho),$$

where $\Pi_C$ denotes the projection (in the Euclidean norm) onto $C$. Intuitively, from (25) and the definition of indicator function $g(z)$ in (19), one can see that the elements of $z$ (i.e., $z_1, z_2, \ldots, z_n$) should go into $C$ in finite time. If not, one will have $g(z) = \infty$, it is known as the definition of $g(z)$, i.e., $g(z) = \infty$ implies $z \notin C$ and update (25) which includes $g(z)$ will never be finished. Then, from the definition of $C$ in (17), one can see that $z$ going into $C$ means $z_1 = z_2 = \ldots = z_n$, which is in the mathematical format of consensus. Therefore, if each node $v_i \in \mathcal{V}$ can have $z_i$ reach $\frac{1}{n} \sum_{i=1}^{n} z_i(0)$ in a finite number of steps, with $z_i(0) = x_i^{k+1} + \lambda_i^k / \rho$, then the update can be completed. Hence, update (25) reduces to a finite-time consensus problem. For this reason, we adopt the FTERC algorithm for digraphs, introduced in Section II-D.

Algorithm 1 D-ADMM-FTERC

1: **Input:** $G(\mathcal{V}, \mathcal{E})$, $\rho > 0$, $n'$ (upper bound on $n$), $k_{\text{max}}$ (ADMM maximum number of iterations)
2: **Data:** Node $v_i \in \mathcal{V}$ sets $x_i^{0}, z_i^{0}, \lambda_i^{0}$ randomly, and $k = 0$
3: **Node** $v_i \in \mathcal{V}$ does the following:
4: **while** $k \leq k_{\text{max}}$ **do**
5: **Compute** $x_i^{k+1}$ using Eq. (24)
6: **if** $k = 0$ **then**
7: **Compute** $z_i^{1}$ via FTERC which runs for $2n'$ steps, and determine $M_i$ and $\beta_i$
8: **else if** $k = 1$ **then**
9: **Run** max-consensus and ratio consensus (3): determine $M_{\text{max}}$ (via max-consensus), compute $z_i^2$ with the same $\beta_i$, and terminate iterations after $n'$ steps
10: **else**
11: **Compute** $z_i^{k+1}$ via ratio consensus (3) with the same $\beta_i$ (runs for $t_{\text{max}}$)
12: **end if**
13: **Compute** $\lambda_i^{k+1}$ using Eq. (26)
14: **if** ADMM stopping criterion is satisfied **then**
15: **Stop** D-ADMM-FTERC
16: **end if**
17: **k** $\leftarrow$ $k + 1$
18: **end while**

For the optimization, we assume that all nodes are aware of an upper bound of the size of the network $n'$ (i.e., $n' \geq n$ and $n'$ is known to all nodes), the augmented Lagrangian function parameter $\rho$, and the ADMM maximum optimization step $k_{\text{max}}$ ($k_{\text{max}}$ is decided based on task time requirement or computer computing capability). The optimization consists of the following steps:

1) At the first optimization step, node $v_i \in \mathcal{V}$ computes $x_i^1$ using (24), computes $z_i^{1}$ via FTERC which runs for $2n'$ iterations. By that time, it is guaranteed that each node has computed their final value $x_i^1$, which requires computing $\beta_i$ and as a consequence $M_i$ is determined. Then, using $x_i^1$ and $z_i^1$, it computes $\lambda_i^1$ using (26).

2) At the second optimization step, node $v_i \in \mathcal{V}$, computes $x_i^2$ using (24), runs ratio consensus (3) for $n'$ iterations and computes $z_i^2$ with the same $\beta_i$, computed at the first optimization step (i.e., there is no need to compute the defective Hankel matrices again). At the same time it runs a max-consensus algorithm with initial condition $x_i^0 = M_i + 1$. Note that $M_i + 1 < n$ is not (as $x_i^0 = M_i + 1$ and $n' < n$). Hence, at this step node $v_i$, not only computes $z_i^2$, but also the maximum number of iterations needed $t_{\text{max}} := M_{\text{max}} + 1$ by each node $v_i \in \mathcal{V}$ for every optimization step $k$ to compute their $z_i^{k+1}$. Again, using $x_i^2$ and $z_i^2$, it computes $\lambda_i^2$ using (26).

3) At each optimization step thereafter, node $v_i \in \mathcal{V}$ computes $x_i^{k+1}$ using (24), $z_i^{k+1}$ via ratio consensus (3) with
The ADMM algorithm terminates once the stopping criterion is satisfied or the maximum number of optimization steps, $k_{\text{max}}$ is reached.

The algorithm guarantees that the number of iterations needed at every optimization step $k$, $k \geq 2$ is the minimum (see properties of FTERC) and that the solution at every step is the exact optimal. Fig. 1 shows the number of iterations needed at every optimization step.

We now formally describe our algorithm, herein called Algorithm 7, in which the nodes distributively solve optimization problem (20).

**Remark 2:** For FTERC algorithm, the exact finite-time iteration number $t_{\text{max}}$ can be determined, as shown in Algorithm 1. On the contrary, for finite-time ε consensus in [14], the iteration number cannot be decided by theory and also varies depending on the value of $\epsilon$.

**V. CONVERGENCE ANALYSIS**

In this section, the $O(\frac{1}{\epsilon^2})$ convergence rate of our proposed D-ADMM-FTER algorithm will be presented. The analysis is inspired by [12] and [14]. Authors in [12] analyzed the D-ADMM based on the undirected graph. The D-ADMM for the directed graph is proposed in [14] based on a finite-time “approximate” consensus method, which means the resulted solution to problem (20) will not be optimal, but close to the optimal solution $(\bar{X}^*, \bar{z}^*)$. By using the FTERC method presented in the previous section, we will prove our D-ADMM-FTER algorithm for the digraph being able to get the optimal solution $(\bar{X}^*, \bar{z}^*)$. The following relationship holds for any iteration $k$ as

$$0 \leq L(\overline{X}^k, z^k, \lambda^k) - L(\bar{X}^*, \bar{z}^*, \lambda^*) \leq \frac{1}{k} \left( \frac{1}{2\rho} ||\lambda^k - \lambda^0||^2 + \frac{\rho}{2} ||\bar{X}^* - z^0||^2 \right).$$  \hspace{1cm} (27)

**Proof:** From the second inequality of the saddle point of Lagrangian function (22), the first inequality in (27) can be proved directly.

We now prove the second inequality in (27). For each node $i$, since $x_{i}^{k+1}$ minimizes $L_{\rho}(x, z^k, \lambda^k)$ in (24), by the optimal condition, we have

$$(x - x_{i}^{k+1})^T [h_i(x_{i}^{k+1}) + \lambda_{i}^{k+1} + \rho(z_{i}^{k+1} - z^k)] \geq 0,$$  \hspace{1cm} (28)

where $h_i(x_{i}^{k+1})$ is the sub-gradient of $f_i$ at $x_{i}^{k+1}$. By integrating $x_{i}^{k+1} = (\lambda_{i}^{k+1} - \lambda_{i}^{k})/\rho + z_{i}^{k+1}$ from (26) into the above inequality, we have

$$(x - x_{i}^{k+1})^T [h_i(x_{i}^{k+1}) + \lambda_{i}^{k+1} + \rho(z_{i}^{k+1} - z^k)] \geq 0.$$  \hspace{1cm} (29)

The compact mathematical format of the above $n$ inequalities can be written as

$$(X - X^{k+1})^T [\bar{h}(X^{k+1}) + \lambda^{k+1} + \rho(z^{k+1} - z^k)] \geq 0,$$  \hspace{1cm} (30)

where $\bar{h}(X^{k+1}) = [h_1(x_1^{k+1}), \ldots, h_n(x_n^{k+1})]^T$. Since $z^{k+1}$ minimizes $L_{\rho}(x^{k+1}, z, \lambda^k)$ in (25), similarly, for all $z \in C$,

$$(z - z^{k+1})^T [\bar{g}(z^{k+1}) - \lambda^k - \rho(X^{k+1} - z^{k+1})] \geq 0,$$  \hspace{1cm} (31)

where $\bar{g}(z^{k+1})$ is the sub-gradient of $g$ at $z^{k+1}$. As both $F$ and $g$ are convex, by utilizing the sub-gradient inequality,

$$F(X^{k+1}) - F(X) + g(z^{k+1}) - g(z) \leq - (X - X^{k+1})^T \bar{h}(X^{k+1}) - (z - z^{k+1})^T \bar{g}(z^{k+1}) \leq \lambda^{k+1} F(X - X^{k+1}) + \rho(X - X^{k+1})^T (z^{k+1} - z),$$  \hspace{1cm} (32)

where the last inequality comes from (30) and (31). Due to feasibility of the optimal solution $(\bar{X}^*, \bar{z}^*)$, we obtain $X^* - z^* = 0$. By setting $X = X^*$, $z = z^*$, (32) becomes

$$F(X^{k+1}) - F(X^*) + g(z^{k+1}) - g(z^*) \leq \lambda^{k+1} (z^{k+1} - X^{k+1}) + \rho(X^* - X^{k+1})^T (z^{k+1} - z),$$  \hspace{1cm} (33)

Adding $\lambda^{k+1} (X^{k+1} - z^{k+1})$ to both sides of (33), we have

$$F(X^{k+1}) - F(X^*) + g(z^{k+1}) - g(z^*) + \lambda^{k+1} (X^{k+1} - z^{k+1}) \leq (\lambda^* - \lambda^{k+1})^T (X^{k+1} - z^{k+1}) + \rho(X^* - X^{k+1})^T (z^{k+1} - z) \leq \frac{1}{\rho} (\lambda^* - \lambda^{k+1})^T (X^{k+1} - z^{k+1}) + (X^* - X^{k+1})^T (z^{k+1} - z),$$  \hspace{1cm} (34)

where the last equality is calculated from (26). Recall an equality law (proof in Appendix) that

$$(a_1 - a_2)^T(a_3 - a_4) = \frac{1}{2} [||a_1 - a_4||^2 - ||a_1 - a_3||^2] \geq \frac{1}{2} [||a_2 - a_3||^2 - ||a_2 - a_4||^2], \forall a_1, a_2, a_3, a_4 \in \mathbb{R}^p.$$  \hspace{1cm} (35)

Then, by using equality (35), (34) changes to

$$F(X^{k+1}) - F(X^*) + g(z^{k+1}) - g(z^*) + \lambda^{k+1} (X^{k+1} - z^{k+1}) \leq \frac{1}{2\rho} [||\lambda^* - \lambda^{k+1}||^2 - ||\lambda^* - \lambda^{k+1}||^2 + ||X^{k+1} - z^{k+1}||^2 - ||X^* - z^{k+1}||^2] \leq \frac{1}{2\rho} [||\lambda^* - \lambda^{k+1}||^2 - ||\lambda^* - \lambda^{k+1}||^2] + \frac{\rho}{2} [||X^* - z^{k+1}||^2 - ||X^* - z^{k+1}||^2] \leq \frac{1}{2\rho} [||\lambda^* - \lambda^{k+1}||^2 - ||\lambda^* - \lambda^{k+1}||^2] + \frac{\rho}{2} [||X^* - z^{k+1}||^2 - ||X^* - z^{k+1}||^2].$$  \hspace{1cm} (36)

where the last inequality comes from using (26) and dropping the negative term $-\frac{\rho}{2} ||X^* - z^{k+1}||^2$. Now, by using $s := k$, we change (36) to another format as

$$F(X^{s+1}) - F(X^*) + g(z^{s+1}) - g(z^*) + \lambda^{s+1} (X^{s+1} - z^{s+1}) \leq \frac{1}{2\rho} [||\lambda^* - \lambda^{s+1}||^2 - ||\lambda^* - \lambda^{s+1}||^2] + \frac{\rho}{2} [||X^* - z^{s+1}||^2 - ||X^* - z^{s+1}||^2],$$  \hspace{1cm} (37)
which holds true for all \( s \). By summing (37) over \( s = 0, 1, \ldots, k - 1 \) and after telescoping calculation, we have
\[
\sum_{s=0}^{k-1} F(X^{s+1}) + \sum_{s=0}^{k-1} g(z^{s+1}) + \lambda^T \sum_{s=0}^{k-1} (X^{s+1} - z^{s+1}) - kF(X^*) - kg(z^*) \\
\leq \frac{1}{2\rho} (\|\lambda^* - \lambda^0\|^2 + \frac{\rho}{2} \|X^* - z^0\|^2).
\]

Due to the convexity of both \( F \) and \( g \), we get \( kF(X^k) \leq \sum_{s=0}^{k-1} F(X^{s+1}) \) and \( kg(z^k) \leq \sum_{s=0}^{k-1} g(z^{s+1}) \). Thus, utilizing the definition of \( X^k \), \( z^k \) and dropping the negative terms, we have
\[
kF(X^k) - kF(X^*) + kg(z^k) - kg(z^*) + \lambda^T(k\tilde{X}^k - k\tilde{z}^k) \\
\leq \frac{1}{2\rho} \|\lambda^* - \lambda^0\|^2 + \frac{\rho}{2} \|\lambda^* - \lambda^k\|^2.
\]

Based on \( X^* - z^* = 0 \), (39) combined with the definition of Lagrangian function [cf. (21)] prove (27).

Remark 3: The convergence proof here is basically different from the ones in [12] and [14] as the investigated problems are different. Specifically, in [12], the proposed D-ADMM can be only applied to nodes with undirected graphs as the constraint \( X = 0 \) is needed to minimize the objective function (6), where the matrix \( A \) is related to the communication graph structure which must be undirected. To apply D-ADMM for digraphs, authors in [14] proposed a different constraint which is \( \|x_i - x_j\| \leq \varepsilon, \varepsilon > 0 \) with the value of \( \varepsilon \) predefined. Note that the above constraint will inevitably lead to a sub-optimal solution, which is close to the optimal but not the optimal as we can see from the comparisons in Section VI. In this paper, we propose the constraint \( x_i = x_j \) to guarantee the solution is optimal by using the FTERC algorithm.

Remark 4: Theorem 2 demonstrates D-ADMM-FTERC solves problem (29) with a rate of \( O(1/k) \). From Theorem 1 and Algorithm 1 and Fig. 1, one can see after the first 2 ADMM iterations, node \( v_i \) using FTERC can finish \( z^{k+1} \) update (25) in \( t_{\text{max}} \) ratio consensus steps with \( \beta_i \) (calculated once in the first ADMM iteration). This is very time-efficient compared to the algorithm in [14], as we will see in Fig. 4 and Table I.

VI. EXAMPLES

The distributed least square problem is considered as
\[
\arg\min_{x \in \mathbb{R}^p} f(x) = \frac{1}{n} \sum_{i=1}^{n} \|A_i x - b_i\|^2,
\]
where \( A_i \in \mathbb{R}^{n \times p} \) is only known to node \( v_i \), \( b_i \in \mathbb{R}^q \) is the measured data and \( x \in \mathbb{R}^p \) is the common decision variable that needs to be optimized. For the automatic generation of large number of different matrices \( A_i \), we choose \( q = p \) to have the square \( A_i \). All elements of \( A_i \) and \( b_i \) are set from independent and identically distributed (i.i.d.) samples of standard normal distribution \( \mathcal{N}(0, 1) \). To better demonstrate the difference between our proposed Algorithm 1 and the algorithm in [14], we set \( p = 3 \).

Example 1. First, we choose \( n = 6 \) to have 6 nodes having a strongly connected digraph. Fig. 2 shows that the solution of D-ADMM-FTERC is always smaller than the one of D-DistADMM in [14] [14] no matter whether we have ADMM stopping condition or not, which verifies Remark 3. Note the value of \( \epsilon = 0.01 \) is already very small and \( \epsilon \) cannot be zero. If it is the case, the finite-time \( \epsilon \) consensus technique in D-DistADMM will become normal ratio consensus technique in Section II-C making the \( z^{k+1} \) update in (25) converges in infinite time as it asks \( z_i = z_j \) (from \( \|z_i - z_j\| \leq \varepsilon \)). Fig. 3 gives more details about the comparisons.

Example 2. Now, we choose \( n = 70, 100 \) and 700, respectively, with a random strongly connected digraph. Here, we choose \( n' = n + 1 \) from the FERTC structure in Fig. 1 which is verified by Fig 4. In addition, Fig 4 demonstrates that after the first 2 ADMM iterations, FTERC iteration number in each ADMM iteration is much smaller than finite-time \( \epsilon \) consensus in [14]. Note that during the first two ADMM optimization steps, finite-time \( \epsilon \) consensus has less iteration numbers. This is related to the node number.
Furthermore, Table I describes the running time comparison on an Intel Core i5 processor at 2.6 GHz with Matlab R2020b for $\epsilon = 0.01, k_{\text{max}} = 200$ for this example. One can see D-ADMM-FTERC is time-efficient than D-DistADMM, and is much more time-efficient for large scale systems.

![Fig. 4. Ratio consensus iteration number comparison: $n = 70, 100, 700$.](image)

### Table I

<table>
<thead>
<tr>
<th>Method</th>
<th>$n=70$</th>
<th>$n=100$</th>
<th>$n=700$</th>
</tr>
</thead>
<tbody>
<tr>
<td>D-ADMM-FTERC</td>
<td>1.4122s</td>
<td>2.1967s</td>
<td>30.8675s</td>
</tr>
<tr>
<td>D-DistADMM</td>
<td>22.3361s</td>
<td>78.7964s</td>
<td>308.675s</td>
</tr>
</tbody>
</table>

### VII. CONCLUSIONS AND FUTURE DIRECTIONS

#### A. Conclusions

A distributed alternating direction method of multipliers using finite-time exact ratio consensus (D-ADMM-FTERC) algorithm is proposed to solve the multi-node convex optimization problem under digraphs. Compared to other state-of-art distributed ADMM algorithms, D-ADMM-FTERC can not only apply to digraphs and reach the optimal solution, but also is very time-efficient, especially for large-scale systems.

#### B. Future Directions

This work assumes that the nodes are aware of an upper bound of the size of the network, which in most applications consisting of static networks (e.g., in data centers) this is readily available. However, in more dynamical networks (e.g., sensor networks) this becomes a limitation. We readily available. However, in more dynamical networks consisting of static networks (e.g., in data centers) this is readily available. However, in more dynamical networks (e.g., sensor networks) this becomes a limitation. We

Furthermore, it would be interesting to study the asynchronous implementation of our proposed method.

#### APPENDIX

$$a_1 - a_4 = a_1 - a_4 - 2a_1 a_4 + a_3 a_4 + a_2 a_4 - a_1 a_4$$

$$= 2a_1^2 a_4 + a_1^2 a_4 - 2a_1 a_4 + 2a_1 a_4 - a_3 a_4$$

$$= a_1^2 (a_3 - a_4) - a_2^2 (a_3 - a_4).$$

Hence, Eq. (35) holds.

### REFERENCES


### Appendix

$$\mathbf{R}$$

$$\mathbf{X}$$

$$\mathbf{Y}$$

$$\mathbf{Z}$$

$$\mathbf{A}$$

$$\mathbf{B}$$