2024 American Control Conference (ACC)
July 8-12, 2024. Toronto, Canada

Distributed Conjugate Gradient Method via Conjugate Direction Tracking

Ola Shorinwa[1] and Mac Schwager[2]

***Abstract*— We present a distributed conjugate gradient method for distributed optimization problems, where each agent computes an optimal solution of the problem locally *without* any central computation or coordination, while communicating with its immediate, one-hop neighbors over a communication network. Each agent updates its local problem variable using an estimate of the *average conjugate direction* across the network, computed via a dynamic consensus approach. Our algorithm enables the agents to use *uncoordinated* step-sizes. We prove convergence of the local variable of each agent to the optimal solution of the aggregate optimization problem, without requiring decreasing step-sizes. In addition, we demonstrate the efficacy of our algorithm in distributed state estimation problems, and its robust counterparts, where we show its performance compared to existing distributed first-order optimization methods.**

I. Introduction

A variety of problems in many disciplines can be formulated as distributed optimization problems, where a group of agents seek to compute the optimal estimate, action, or control that minimizes (or maximizes) a specified objective function. Examples of such problems include distributed target tracking [1], [2], pose/state/signal estimation in sensor/robotic networks [3]; machine learning and statistical modeling [4]; process control [5]; and multi-agent planning and control [6], [7]. In these problems, the data is collected and stored locally by each agent, with the additional constraint that no individual agent has access to all the problem data across the network. In many situations, the limited availability of communication and data storage resources, in addition to privacy regulations, preclude the aggregation of the problem data at a central location or node, effectively rendering *centralized optimization* methods infeasible.

Distributed optimization enables each agent to compute an optimal solution via local computation procedures while communicating with its neighbors over a communication network. In essence, via distributed optimization, each agent *collaborates* with its neighbors to compute an optimal solution without access to the aggregate problem data. Some distributed optimization methods require a central coordinator for execution or coordination of some of the update procedures. These methods are often used in machine learning for parallel processing on a cluster of computing nodes, especially in problems involving large datasets. In contrast, in this work, we focus on *fully-distributed* algorithms that do not require a central node for coordination or computation.

We derive a distributed conjugate gradient algorithm, termed DC-Grad, for distributed optimization problems. In our algorithm, each agent utilizes *first-order* information (i.e., gradients) of its local objective function to compute its local conjugate directions for updating its local estimate of the solution of the optimization problem at each iteration and communicates with its one-hop neighbor over a point-to-point communication network. Each agent does not share its local problem data, including its objective function and gradients, with other agents, preserving the *privacy* of the agents. For simplicity of exposition, we limit our analysis to distributed optimization problems with smooth, convex objective functions. We prove convergence of the local problem variables of all agents to the optimal solution of the aggregate optimization problem.

We examine the performance of our distributed algorithm in comparison to notable existing distributed optimization methods in distributed state estimation and robust-state-estimation problems. In both problems, we show that our algorithm converges with the least communication overhead in densely-connected communication networks, with some additional computation overhead in comparison to the best-competing distributed algorithm DIGing-ATC. On sparsely-connected graphs, our algorithm performs similarly to other first-order distributed optimization methods.

II. Related Work

Distributed optimization methods have received significant attention, with many such methods developed from their centralized counterparts. Distributed first-order methods leverage the local gradients (i.e., first-order information) of each agent to iteratively improve the each agent's local estimate of the optimal solution of the optimization problem, bearing similarities with other centralized first-order methods such as the centralized gradient descent. Distributed incremental (sub)gradient methods require a central node that receives the local gradient information from each agent and performs the associated update step [8]. As such, these methods require a hub-spoke communication model — where all the agents are connected to the central node (hub) — or a ring communication model (a cyclic network), which is quite restrictive.

Distributed (sub)gradient methods circumvent this limitation, enabling distributed optimization over arbitrary network topologies. At each iteration, each agent exchanges its local iterates and other auxiliary variables (such as estimates of

*This work was supported in part by NSF NRI awards 1830402 and 1925030 and ONR grant N00014-18-1-2830.

[1]Ola Shorinwa is with the Department of Mechanical Engineering, Stanford University, CA, USA shorinwa@stanford.edu.

[2]Mac Schwager is with the Department of Aeronautics and Astronautics Engineering, Stanford University, CA, USA schwager@stanford.edu.

the average gradient of the joint (global) objective function) with other neighboring agents. In distributed (sub)gradient methods, each agent recursively updates its local estimate using its local (sub)gradient and *mixes* its estimates with the estimates of its neighbors via a convex combination, where the *mixing* step is achieved via average consensus [9] or the push-sum technique [10]. Generally, distributed (sub)gradient methods require a diminishing step-size for convergence to the optimal solution in convex problems [11], which typically slows down convergence. With a constant step-size, these methods converge to a neighborhood of the optimal solution.

Distributed gradient-tracking methods eliminate the need for diminishing step-sizes [12]–[16] by enabling each agent to utilize an estimate of the *average gradient* of the objective function in updating its local estimate of the optimal solution. Distributed gradient-tracking methods provide faster convergence guarantees with constant step-sizes. The alternating direction method of multipliers (ADMM) has also been applied to distributed optimization problems [17], [18]. However, the update procedures in ADMM-based algorithms generally incur greater computational complexity, often requiring higher-order information of the local objective functions.

The conjugate gradient (CG) method was originally developed for computing the solution of a linear system of equations (i.e., $Ax = b$), where the matrix $A \in \mathbb{R}^{n\times n}$ is square, symmetric, and positive-definite [19], [20]. More generally, the method applies to strongly-convex quadratic programming problems, where the conjugate gradient method is guaranteed to compute the optimal solution in at most n iterations, in the absence of roundoff errors. The conjugate gradient method has been extended to nonlinear optimization problems (which includes non-quadratic problems) [21], [22]. In general, the conjugate gradient method provides faster convergence compared to gradient descent methods [23]. Variants of the conjugate gradient method for parallel execution on multiple computing nodes (processors) have been developed [24]–[27]. These methods decompose the data matrix associated with the linear system of equations into individual components assigned to each processor, enabling parallelization of the matrix-vector operations arising in the conjugate gradient method, which constitute the major computational bottleneck in the CG method. However, these methods are only amenable to problems with hub-spoke communication models or all-to-all communication models. Some other distributed CG methods eliminate the need for a hub-spoke communication model [28], but, however, require a ring communication model, which does not support parallel execution of the update procedures, ultimately degrading the computational speed of the algorithm. The distributed variant [29] allows for more general communication networks. Nonetheless, these CG methods are limited to solving a linear system of equations and do not consider a more general optimization problem. Few distributed CG methods for nonlinear optimization problems exist. The work in [30] derives a distributed CG method for online optimization problems where mixing of information is achieved using the average consensus scheme. Like distributed (sub)gradient methods, this algorithm requires a diminishing step-size for convergence to the optimal solution, converging to a neighborhood of the optimal solution if a constant step-size is used.

In this paper, we derive a distributed conjugate gradient method for a more general class of optimization problems, including problems with nonlinear objective functions, and prove convergence of the algorithm to the optimal solution in convex problems with Lipschitz-continuous gradients. Moreover, we note that, in our algorithm, each agent can use uncoordinated constant step-sizes.

III. Notation and Preliminaries

In this paper, we denote the gradient of a function f by ∇f and g, interchangeably. We denote the all-ones vector as $\mathbf{1}_n \in \mathbb{R}^n$. We represent the inner-product of two matrices $A \in \mathbb{R}^{m\times n}$ and $B \in R^{m\times n}$ as $\langle A, B\rangle = \operatorname{trace}\left(A^{\mathsf{T}}B\right)$. We denote the standard scalar-vector product, matrix-vector product, and matrix-matrix product (composition) as $A \cdot B$, depending on the mathematical context. For a given matrix $A \in \mathbb{R}^{m\times n}$, we denote its spectral norm as $\rho(A) = \|A\|_2$. Further, we denote its Frobenius norm by $\|A\|_F$. Likewise, we define the mean of a matrix $B \in \mathbb{R}^{N\times n}$, computed across its rows, as $\overline{B} = \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^{\mathsf{T}}B \in \mathbb{R}^{N\times n}$, where each row of $\overline{B}$ is the same. In addition, we define the consensus violation between the matrix $B \in \mathbb{R}^{N\times n}$ and its mean $\overline{B} \in \mathbb{R}^{N\times n}$ as $\tilde{B} = B - \overline{B}$. We denote the domain of a function f as $\operatorname{dom}(f)$, the non-negative orthant as $\mathbb{R}_+$, and the strictly-positive orthant as $\mathbb{R}_{++}$.

We introduce the following definitions that will be relevant to our discussion.

Definition 1 (Conjugacy)**.** *Two vectors $a, b \in \mathbb{R}^n$ are conjugate with respect to a symmetric positive-definite matrix $C \in \mathbb{R}^{n\times n}$ if:*

$$a^{\mathsf{T}}Cb = \langle a, Cb\rangle = \langle Ca, b\rangle = 0. \tag{1}$$

Definition 2 (Convex Function)**.** *A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if for all $x, y \in \operatorname{dom}(f)$ and all $\zeta \in [0, 1]$:*

$$f(\zeta x + (1-\zeta)y) \le \zeta f(x) + (1-\zeta)f(y), \tag{2}$$

and the domain of f, $\operatorname{dom}(f) \subseteq \mathbb{R}^n$, is convex.

Definition 3 (Smoothness)**.** *A function $f : \mathbb{R}^n \to \mathbb{R}$ is L-smooth if it is continuously differentiable over its domain and its gradients are L-Lipschitz continuous, i.e.:*

$$\|\nabla f(x) - \nabla f(y)\|_2 \le L\|x - y\|_2, \ \forall x, y \in \operatorname{dom}(f), \tag{3}$$

where $L \in \mathbb{R}_{++}$ is the Lipschitz constant.

Definition 4 (Coercive Function)**.** *A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is coercive if $f(x) \to \infty$ as $x \to \infty$, for all $x \in \operatorname{dom}(f)$.*

We represent the agents as nodes in an undirected, connected communication graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \ldots, N\}$ denotes the set of vertices, representing the agents, and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ represents the set of edges. An edge (i, j) exists in $\mathcal{E}$ if agents i and j share a communication link. Moreover, we denote the set of neighbors of agent i

as $\mathcal{N}_i$. We associate a *mixing matrix* $W \in \mathbb{R}^{N\times N}$ with the underlying communication graph. A mixing matrix W is compatible with $\mathcal{G}$ if $w_{ij} = 0$, $\forall j \notin \mathcal{N}_i \cup \{i\}$, $\forall i \in \mathcal{V}$. We denote the *degree* of agent i as $\deg(i) = |\mathcal{N}_i|$, representing the number of neighbors of agent i, and the *adjacency matrix* associated with $\mathcal{G}$ as $\mathcal{A} \in \mathbb{R}^{N\times N}$, where $\mathcal{A}_{ij} = 1$ if and only if $j \in \mathcal{N}_i$, $\forall i \in \mathcal{V}$. In addition, we denote the *graph Laplacian* of $\mathcal{G}$ as $L = \mathrm{diag}(\deg(1), \ldots, \deg(N)) - \mathcal{A}$. In this work, we make the following assumption on the mixing matrix.

Assumption 1. *The mixing matrix W associated with the communication graph G satisfies:*

1) (Double-Stochasticity) $W\mathbf{1} = \mathbf{1}$ *and* $\mathbf{1}^\mathsf{T}W = \mathbf{1}$,

2) (Spectral Property) $\lambda = \rho(W - \frac{\mathbf{1}_N\mathbf{1}_N^\mathsf{T}}{N}) < 1$.

Part 2 of Assumption 1 specifies that the matrix $M = W - \frac{\mathbf{1}_N\mathbf{1}_N^\mathsf{T}}{N}$ has a spectral norm less than one. This assumption is necessary and sufficient for consensus, i.e.,

$$\lim_{k\to\infty} W^k \to \frac{\mathbf{1}_N\mathbf{1}_N^\mathsf{T}}{N}. \tag{4}$$

We note that Assumption 1 is not restrictive, in undirected communication networks. We provide common choices for the mixing matrix W:

1) *Metropolis-Hastings Weights*:

$$w_{ij} = \begin{cases} \frac{1}{\max\{\deg(i),\deg(j)\}+\epsilon}, & \text{if } (i,j) \in \mathcal{E}, \\ 0 & \text{if } (i,j) \notin \mathcal{E} \text{ and } i \neq j, \\ 1 - \sum_{r\in\mathcal{V}} w_{ir} & \text{if } i = j, \end{cases}$$

where $\epsilon \in \mathbb{R}_{++}$ denotes a small positive constant, e.g., $\epsilon = 1$ [9].

2) *Laplacian-based Weights*:

$$W = I - \frac{L}{\tau},$$

where L denotes the Laplacian matrix of $\mathcal{G}$, and $\tau \in \mathbb{R}$ denotes a scaling parameter with $\tau > \frac{1}{2}\lambda_{\max}(L)$. One can choose $\tau = \max_{i\in\mathcal{V}}\{\deg(i)\} + \epsilon$, if computing $\lambda_{\max}(L)$ is infeasible, where $\epsilon \in \mathbb{R}_{++}$ represents a small positive constant [31].

IV. Problem Formulation And The Centralized Conjugate Gradient Method

We consider the distributed optimization problem:

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \ \frac{1}{N}\sum_{i=1}^{N} f_i(x), \tag{5}$$

over N agents, where $f_i : \mathbb{R}^n \to \mathbb{R}$ denotes the local objective function of agent i and $x \in \mathbb{R}$ denotes the optimization variable. The objective function of the optimization problem (5) consists of a sum of N local components, making it *separable*, with each component associated with an agent. We assume that agent i only knows its local objective function f_i and has no knowledge of the objective function of other agents.

We begin with a description of the centralized nonlinear conjugate gradient method, before deriving our method in Section V. The nonlinear conjugate gradient method (a generalization of the conjugate gradient method to optimization problems beyond quadratic programs) represents an iterative first-order optimization algorithm that utilizes the gradient of the objective function to generate iterates from the recurrence:

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} \cdot s^{(k)}, \tag{6}$$

where $x^{(k)} \in \mathbb{R}^n$ denotes the estimate at iteration k, $\alpha^{(k)} \in \mathbb{R}_+$ denotes the step-size at iteration k, and $s^{(k)} \in \mathbb{R}^n$ denotes the conjugate direction at iteration k. In the nonlinear conjugate direction method, the conjugate direction is initialized as the negative gradient of the objective function at the initial estimate, with $s^{(0)} = -g^{(0)}$. Further, the conjugate directions are generated from the recurrence:

$$s^{(k+1)} = -g^{(k+1)} + \beta^{(k)} \cdot s^{(k)}, \tag{7}$$

at iteration k, where $\beta^{(k)} \in \mathbb{R}$ denotes the conjugate gradient update parameter. Different schemes have been developed for updating the conjugate update parameter, such as the *Hestenes-Stiefel Scheme* [19], *Fletcher-Reeves Scheme* [32], and *Polak-Ribière Scheme* [33], [34].

We note that the update schemes are equivalent when f is a strongly-convex quadratic function. Moreover, when f is strongly-convex and quadratic, the search directions $\{s^{(k)}\}_{\forall k}$ are conjugate. As a result, the iterate $x^{(k)}$ converges to the optimal solution in at most n iterations. For non-quadratic problems, the search directions lose conjugacy, and convergence may occur after more than n iterations. In many practical problems, the value of the update parameter β is selected via a hybrid scheme, obtained from a combination of the fundamental update schemes, which include the aforementioned ones. Simple hybrid schemes are also used, e.g., $\beta^{(k)} = \max\{0, \beta_{PR}^{(k)}\}$.

V. Distributed Conjugate Gradient Method

In this section, we derive a distributed optimization algorithm based on the nonlinear conjugate method for (5). We assign a local copy of x to each agent, representing its local estimate of the solution of the optimization problem, with each agent computing its conjugate directions locally. Agent i maintains the variables: $x_i \in \mathbb{R}^n$, $s_i \in \mathbb{R}^n$, along with $\alpha_i \in \mathbb{R}_+$, and $\beta_i \in \mathbb{R}$. In addition, we denote the gradient of f_i at x_i by $g_i(x_i)$.

Before proceeding with the derivation, we introduce the following notation:

$$\boldsymbol{x} = \begin{bmatrix} — & x_1^\mathsf{T} & — \\ & \vdots & \\ — & x_N^\mathsf{T} & — \end{bmatrix}, \ \boldsymbol{s} = \begin{bmatrix} — & s_1^\mathsf{T} & — \\ & \vdots & \\ — & s_N^\mathsf{T} & — \end{bmatrix},$$

$$\boldsymbol{g}(\boldsymbol{x}) = \begin{bmatrix} — & (\nabla f_1(x_1))^\mathsf{T} & — \\ & \vdots & \\ — & (\nabla f_N(x_N))^\mathsf{T} & — \end{bmatrix},$$

$\boldsymbol{\alpha} = \mathrm{diag}(\alpha_1, \ldots, \alpha_N)$, and $\boldsymbol{\beta} = \mathrm{diag}(\beta_1, \ldots, \beta_N)$, where the variables are obtained by stacking the local variables of each agent, with $\boldsymbol{x} \in \mathbb{R}^{N\times n}$, $\boldsymbol{s} \in \mathbb{R}^{N\times n}$, $\boldsymbol{g}(\boldsymbol{x}) \in \mathbb{R}^{N\times n}$, and

$\boldsymbol{\alpha} \in \mathbb{R}^N$. To simplify notation, we denote $\boldsymbol{g}(\boldsymbol{x}^{(k)})$ by $\boldsymbol{g}^k$. In addition, we note that all agents achieve *consensus* when all the rows of $\boldsymbol{x}$ are the same. Moreover, *optimality* is achieved when $\mathbf{1}_N^\mathsf{T} \boldsymbol{g}(\boldsymbol{x}) = \mathbf{0}_n^\mathsf{T}$, i.e., the first-order optimality condition is satisfied. Further, we define the *aggregate* objective function considering the local variables of each agent as:

$$\boldsymbol{f}(\boldsymbol{x}) = \frac{1}{N} \sum_{i=1}^{N} f_i(x_i). \tag{8}$$

To obtain a distributed variant of the centralized conjugate gradient method, one could utilize the average consensus technique to eliminate the need for centralized procedures, yielding the distributed algorithm:

$$\boldsymbol{x}^{(k+1)} = W\boldsymbol{x}^{(k)} + \boldsymbol{\alpha}^{(k)} \cdot \boldsymbol{s}^{(k)}, \tag{9}$$

$$\boldsymbol{s}^{(k+1)} = -\boldsymbol{g}^{(k+1)} + \boldsymbol{\beta}^{(k)} \cdot \boldsymbol{s}^{(k)}, \tag{10}$$

which simplifies to:

$$x_i^{(k+1)} = w_{ii} x_i^{(k)} + \sum_{j \in \mathcal{N}_i} w_{ij} x_j^{(k)} + \alpha_i^{(k)} \cdot s_i^{(k)}, \tag{11}$$

$$s_i^{(k+1)} = -g_i^{(k+1)} + \beta_i^{(k)} \cdot s_i^{(k)}, \tag{12}$$

when expressed with respect to agent i, with initialization $x_i^{(0)} \in \mathbb{R}^n$, $s_i^{(0)} = -\nabla f_i(x_i^{(0)})$, and $\alpha_i^{(0)} \in \mathbb{R}_+$.

One can show that the above distributed algorithm does not converge to the optimal solution with a non-diminishing step-size. Here, we provide a simple proof by contradiction showing that the optimal solution $x^\star$ is not a fixed point of the distributed algorithm (9): Assume that $x^\star$ is a fixed point of the algorithm. With this assumption, the first-two terms on the right-hand side of (11) simplify to $x^\star$. Further, the conjugate update parameter $\beta_i^{(k-1)}$ simplifies to zero, where we define the ratio $\frac{0}{0}$ to be zero if the Fletcher-Reeves Scheme is utilized. However, in general, the local conjugate direction of agent i, denoted by $s_i^{(k)}$, may not be zero, since the critical point of the joint objective function $\frac{1}{N}\sum_{i=1}^{N} f_i$ may not coincide with the critical point of f_i, i.e., $\nabla f_i(x^\star)$ may not be zero. Consequently, the last term in (9) is not zero, in general, and as a result, agent i's iterate $x_i^{(k+1)}$ deviates from $x^\star$, showing that $x^\star$ is not a fixed point of the distributed algorithm given by (11). This property mirrors that of distributed (sub)gradient methods where a diminishing step-size is required for convergence.

Further, we note that the last term in (11) is zero if agent i utilizes the average conjugate direction in place of its local conjugate direction. With this modified update procedure, the optimal solution $x^\star$ represents a fixed point of the resulting, albeit non-distributed, algorithm. To address this challenge, we assign an auxiliary variable z to each agent, representing an estimate of the average conjugate direction, which is updated locally using dynamic average consensus [35], yielding the *Distributed Conjugate Gradient Method* (DC-Grad), given by:

$$\boldsymbol{x}^{(k+1)} = W(\boldsymbol{x}^{(k)} + \boldsymbol{\alpha}^{(k)} \cdot \boldsymbol{z}^{(k)}), \tag{13}$$

$$\boldsymbol{s}^{(k+1)} = -\boldsymbol{g}^{(k+1)} + \boldsymbol{\beta}^{(k)} \cdot \boldsymbol{s}^{(k)}, \tag{14}$$

$$\boldsymbol{z}^{(k+1)} = W(\boldsymbol{z}^{(k)} + \boldsymbol{s}^{(k+1)} - \boldsymbol{s}^{(k)}), \tag{15}$$

which is initialized with $x_i^{(0)} \in \mathbb{R}^n$, $s_i^{(0)} = -\nabla f_i(x_i^{(0)})$, $z_i^{(0)} = s_i^{(0)}$, and $\alpha_i^{(0)} \in \mathbb{R}_+$, $\forall i \in \mathcal{V}$. Using dynamic average consensus theory, we can show that the agents reach consensus with $z_i^{(\infty)} = \bar{z}^{(\infty)} = \bar{s}^{(\infty)}$, $\forall i \in \mathcal{V}$. The resulting distributed conjugate gradient method enables each agent to compute the optimal solution of the optimization problem using *uncoordinated*, *non-diminishing* step-sizes.

Considering the update procedures in terms of each agent, at each iteration k, agent i performs the following updates:

$$x_i^{(k+1)} = \sum_{j \in \mathcal{N}_i \cup \{i\}} w_{ij} \left(x_j^{(k)} + \alpha_j^{(k)} \cdot z_j^{(k)} \right), \tag{16}$$

$$s_i^{(k+1)} = -g_i^{(k+1)} + \beta_i^{(k)} \cdot s_i^{(k)}, \tag{17}$$

$$z_i^{(k+1)} = \sum_{j \in \mathcal{N}_i \cup \{i\}} w_{ij} \left(z_j^{(k)} + s_j^{(k+1)} - s_j^{(k)} \right), \tag{18}$$

where agent i communicates:

$$u_i^{(k)} = x_i^{(k)} + \alpha_i^{(k)} \cdot z_i^{(k)}, \tag{19}$$

$$v_i^{(k)} = z_i^{(k)} + s_i^{(k+1)} - s_i^{(k)}, \tag{20}$$

with its neighbors. We summarize the distributed conjugate gradient algorithm in Algorithm 1.

Algorithm 1: Distributed Conjugate Gradient Method (DC-Grad)

Initialization:
$x_i^{(0)} \in \mathbb{R}^n$, $s_i^{(0)} = -\nabla f_i(x_i^{(0)})$, $z_i^{(0)} = s_i^{(0)}$, and $\alpha_i^{(0)} \in \mathbb{R}_+$, $\forall i \in \mathcal{V}$.
do in parallel $\forall i \in \mathcal{V}$
- $x_i^{(k+1)} \leftarrow$ Procedure (16)
- $s_i^{(k+1)} \leftarrow$ Procedure (17)
- $z_i^{(k+1)} \leftarrow$ Procedure (18)
- $k \leftarrow k+1$

while *not converged or stopping criterion is not met*;

We present some assumptions that will be relevant in analyzing the convergence properties of our algorithm.

Assumption 2. *The local objective function of each agent, f_i, is closed, proper, and convex. Moreover, f_i is L_i-Lipschitz-smooth, with Lipschitz-continuous gradients.*

Remark 1. *From Assumption 2, we note that the aggregate objective function $\boldsymbol{f}$ is closed, convex, proper, and Lipschitz-continuous with:*

$$\|\nabla \boldsymbol{f}(x) - \nabla \boldsymbol{f}(y)\|_2 \leq L \|x - y\|_2, \ \forall x, y \in \mathbb{R}^n, \tag{21}$$

where $L = \max_{i \in \mathcal{V}}\{L_i\}$.

Assumption 3. *The local objective function of each agent f_i is coercive.*

Assumption 4. *The optimization problem* (5) *has a non-empty feasible set, and further, an optimal solution $x^\star$ exists for the optimization problem.*

The aforementioned assumptions are standard in convergence analysis of distributed optimization algorithms.

VI. Convergence Analysis

We analyze the convergence properties of our distributed algorithm. Before proceeding with the analysis, we consider the following sequence:

$$\overline{\boldsymbol{x}}^{(k+1)} = \overline{\boldsymbol{x}}^{(k)} + \overline{\boldsymbol{\alpha}^{(k)} \cdot \boldsymbol{z}^{(k)}}, \tag{22}$$

$$\overline{\boldsymbol{s}}^{(k+1)} = -\overline{\boldsymbol{g}}^{(k+1)} + \overline{\boldsymbol{\beta}^{(k)} \cdot \boldsymbol{s}^{(k)}}, \tag{23}$$

$$\overline{\boldsymbol{z}}^{(k+1)} = \overline{\boldsymbol{z}}^{(k)} + \overline{\boldsymbol{s}}^{(k+1)} - \overline{\boldsymbol{s}}^{(k)}, \tag{24}$$

derived from the mean of the local iterates of each agent, where we have utilized the assumption that W is column-stochastic. From (24) , we note that $\overline{z}^{(k)} = \overline{s}^{(k)}$, $\forall k$, given that $\overline{z}^{(0)} = \overline{s}^{(0)}$.

Theorem 1 (Agreement)**.** *Given the recurrence* (13)*,* (14)*, and* (15)*, the local iterates of agent i, $\left(x_i^{(k)}, s_i^{(k)}, z_i^{(k)}\right)$, converge to the mean, $\forall i \in \mathcal{V}$, i.e., each agent reaches agreement with all other agents, for sufficiently large k. In particular:*

$$\lim_{k\to\infty} \left\|\tilde{\boldsymbol{s}}^{(k)}\right\|_2 = 0, \ \lim_{k\to\infty} \left\|\tilde{\boldsymbol{x}}^{(k)}\right\|_2 = 0, \ \lim_{k\to\infty} \left\|\tilde{\boldsymbol{z}}^{(k)}\right\|_2 = 0. \tag{25}$$

Further, the local iterates of each agent converge to a limit point, as $k \to \infty$, with:

$$\lim_{k\to\infty} \left\|\overline{\boldsymbol{\alpha}^{(k)} \cdot \boldsymbol{z}^{(k)}}\right\|_2 = \lim_{k\to\infty} \left\|\overline{\boldsymbol{\beta}^{(k)} \cdot \boldsymbol{s}^{(k)}}\right\|_2 = 0. \tag{26}$$

Moreover, the norm of the mean of the agents' local iterates tracking the average conjugate direction converges to zero, with the norm of the average gradient evaluated at the local iterate of each agent also converging to zero, for sufficiently large k. Specifically, the following holds:

$$\lim_{k\to\infty} \left\|\overline{\boldsymbol{s}}^{(k)}\right\|_2 = 0, \ \lim_{k\to\infty} \left\|\overline{\boldsymbol{z}}^{(k)}\right\|_2 = 0, \ \lim_{k\to\infty} \left\|\overline{\boldsymbol{g}}^{(k)}\right\|_2 = 0. \tag{27}$$

Proof. We refer readers to [36] for the proof. □

Further, we note that $\left\|\nabla f(x^{(\infty)})\right\|_2 = 0$ [36]. Hence, the limit point of the distributed algorithm represents a critical point of the optimization problem (5).

Theorem 2 (Convergence of the Objective Value)**.** *The value of the objective function $\boldsymbol{f}$ evaluated at the mean of the local iterates of all agents converges to the optimal objective value. Moreover, the value of $\boldsymbol{f}$ evaluated at the agents' local iterates converges to the optimal objective value, for sufficiently large k. Particularly:*

$$\lim_{k\to\infty} \boldsymbol{f}(\boldsymbol{x}^{(k)}) = \lim_{k\to\infty} \boldsymbol{f}(\overline{\boldsymbol{x}}^{(k)}) = f^\star, \tag{28}$$

Proof. We provide the proof in [36]. □

VII. Simulations

In this section, we examine the performance of our distributed conjugate gradient method (DC-Grad) in comparison to other existing distributed optimization algorithms, namely: DIGing-ATC [14], C-ADMM [17], AB/Push-Pull [16], and ABm [13], which utilizes *momentum* acceleration to achieve faster convergence. We note that AB/Push-Pull reduces to DIGing-CTA when the matrices A and B are selected to be doubly-stochastic [14]. We assess the convergence rate of our algorithm across a range of communication networks, with varying degrees of connectivity, described by the connectivity ratio $\kappa = \frac{2|\mathcal{E}|}{N(N-1)}$. We consider a *state estimation problem*, formulated as a least-squares optimization problem, in addition to its robust variant derived with the Huber loss function. In each problem, we utilize *Metropolis-Hastings* weights for the mixing matrix W. Since Metropolis-Hastings weights yield doubly-stochastic (DS) mixing matrices, we use the terms ABm and ABm-DS interchangeably. We compute the convergence error of the local iterate of each agent to the optimal solution, in terms of the *relative-squared error* (RSE) given by: $\text{RSE} = \frac{\|x_i - x^\star\|_2}{\|x^\star\|_2}$, where x_i denotes the local iterate of agent i and $x^\star$ denotes the optimal solution, computed from the aggregate optimization problem. We set the threshold for convergence at $1e^{-13}$.

For a good comparison of the computation and communication overhead incurred by each method, we selected a convergence threshold that could be attained by all methods. In our simulation study, we note that DIGing-ATC and DC-Grad yield higher-accuracy solutions compared to the other methods, with AB/Push-Pull yielding solutions with the least accuracy. We utilize the *golden-section search* to select an optimal step-size for our distributed conjugate gradient method, DIGing-ATC, AB/Push-Pull, and ABm. Likewise, we select an optimal value for the penalty parameter ρ in C-ADMM using golden-section search. Further, we assume that each scalar component in the agents' iterates is represented using the double-precision floating-point representation format.

A. Distributed State Estimation

In the state estimation problem, we seek to compute an estimate of a parameter (*state*) given a set of observations (*measurements*). In many situations (e.g., in robotics, process control, and finance), the observations are collected by a network of sensors, resulting in decentralization of the problem data, giving rise to the distributed state estimation problem. Here, we consider the distributed state estimation problem over a network of N agents, where the agents estimate the state $x \in \mathbb{R}^n$, representing the parameter of interest, such as the location of a target. Each agent makes noisy observations of the state, given by the model: $y_i = C_i x + w_i$, where $y_i \in \mathbb{R}^{m_i}$ denotes the observations of agent i, $C_i \in \mathbb{R}^{m_i \times n}$ denotes the observation (measurement) matrix, and w_i denotes random noise. We can formulate the state estimation problem as a least-squares optimization problem, given by:

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \ \frac{1}{N}\sum_{i=1}^{N} \|C_i x - y_i\|_2^2 . \tag{29}$$

We determine the number of local observations for each agent randomly by sampling from the uniform distribution over the closed interval $[5, 30]$. We randomly generate the problem data: C_i and y_i, $\forall i \in \mathcal{V}$, with $N = 50$ and $n = 10$. We examine the convergence rate of the distributed

optimization algorithms over randomly-generated connected communication graphs. We update the conjugate gradient parameter β using a modified *Fletcher-Reeves Scheme*.

In Table I, we present the mean and standard deviation of the cumulative computation time per agent, in seconds, required for convergence by each distributed algorithm, over 20 randomly-generated problems for each communication network. We utilize a closed-form solution for the primal update procedure arising in C-ADMM, making it competitive with other distributed optimization methods in terms of computation time. From Table I, we note that DIGing-ATC requires the shortest computation time, closely followed by DC-Grad, on densely-connected communication graphs, i.e., on graphs with κ close to one, where we note that DC-Grad requires an update procedure for β, increasing its computation time. However, on more sparsely-connected communication graphs, C-ADMM requires the shortest computation time.

TABLE I

THE MEAN AND STANDARD DEVIATION OF THE CUMULATIVE COMPUTATION TIME (IN SECONDS) PER AGENT IN THE DISTRIBUTED STATE ESTIMATION PROBLEM.

Algorithm	$\kappa = 0.48$	$\kappa = 0.80$	$\kappa = 0.97$	$\kappa = 1.00$
AB/Push-Pull [16]	$8.95e^{-4} \pm 1.60e^{-4}$	$9.42e^{-4} \pm 1.44e^{-4}$	$1.03e^{-3} \pm 1.59e^{-4}$	$1.06e^{-3} \pm 1.43e^{-4}$
ABm-DS [13]	$5.54e^{-4} \pm 2.23e^{-5}$	$3.19e^{-4} \pm 3.86e^{-5}$	$2.85e^{-4} \pm 4.71e^{-5}$	$2.91e^{-4} \pm 4.10e^{-5}$
C-ADMM [17]	$\mathbf{1.71e^{-4} \pm 1.87e^{-5}}$	$\mathbf{1.34e^{-4} \pm 1.20e^{-5}}$	$1.18e^{-4} \pm 6.84e^{-6}$	$1.17e^{-4} \pm 7.87e^{-6}$
DIGing-ATC [14]	$5.98e^{-4} \pm 3.08e^{-5}$	$2.12e^{-4} \pm 1.61e^{-5}$	$\mathbf{6.79e^{-5} \pm 9.10e^{-6}}$	$\mathbf{3.87e^{-5} \pm 2.75e^{-6}}$
DC-Grad (ours)	$7.94e^{-4} \pm 4.39e^{-5}$	$2.85e^{-4} \pm 2.04e^{-5}$	$9.10e^{-5} \pm 9.32e^{-6}$	$4.47e^{-5} \pm 2.08e^{-6}$

Moreover, we provide the mean and standard deviation of the cumulative size of messages exchanged per agent, in Megabytes (MB), for each distributed algorithm in Table II. We note that C-ADMM requires agents to communicate fewer variables by a factor of 2, compared to AB/Push-Pull, ABm, DIGing-ATC , and DC-Grad. Table II shows that DC-Grad incurs the least communication overhead for convergence on more-densely-connected graphs, closely followed by DIGing-ATC. This finding reveals that DC-Grad requires fewer iterations for convergence on these graphs, compared to the other algorithms. On more-sparsely-connected graphs, C-ADMM incurs the least communication overhead.

TABLE II

THE MEAN AND STANDARD DEVIATION OF THE CUMULATIVE SIZE OF MESSAGES EXCHANGED BY EACH AGENT (IN MB) IN THE DISTRIBUTED STATE-ESTIMATION PROBLEM.

Algorithm	$\kappa = 0.48$	$\kappa = 0.80$	$\kappa = 0.97$	$\kappa = 1.00$
AB/Push-Pull [16]	$6.06e^{-2} \pm 1.18e^{-2}$	$6.52e^{-2} \pm 1.10e^{-2}$	$6.97e^{-2} \pm 1.23e^{-2}$	$7.20e^{-2} \pm 1.13e^{-2}$
ABm-DS [13]	$3.64e^{-2} \pm 1.31e^{-3}$	$2.13e^{-2} \pm 2.21e^{-3}$	$1.85e^{-2} \pm 3.17e^{-3}$	$1.90e^{-2} \pm 2.78e^{-3}$
C-ADMM [17]	$\mathbf{1.15e^{-2} \pm 9.80e^{-4}}$	$\mathbf{9.24e^{-3} \pm 8.17e^{-4}}$	$7.98e^{-3} \pm 4.78e^{-4}$	$8.03e^{-3} \pm 5.00e^{-4}$
DIGing-ATC [14]	$4.68e^{-2} \pm 1.27e^{-3}$	$1.69e^{-2} \pm 1.11e^{-3}$	$5.16e^{-3} \pm 2.19e^{-4}$	$3.00e^{-3} \pm 2.20e^{-4}$
DC-Grad (ours)	$4.63e^{-2} \pm 1.70e^{-3}$	$1.70e^{-2} \pm 1.10e^{-3}$	$\mathbf{5.16e^{-3} \pm 2.00e^{-4}}$	$\mathbf{2.58e^{-3} \pm 1.00e^{-4}}$

In Figure 1, we show the convergence error of the agents' iterates, per iteration, on a fully-connected communication network. Figure 1 highlights that DC-Grad requires the least number of iterations for convergence, closely followed by DIGing-ATC. In addition, ABm and C-ADMM converge at relatively the same rate. Similarly, we show the convergence error of the iterates of each agent on a randomly-generated connected communication graph with $\kappa = 0.48$ in Figure 2. We note that C-ADMM converges the fastest in Figure 2. In addition, we note that the convergence plot of DIGing-ATC overlays that of DC-Grad, with both algorithms exhibiting a similar performance.

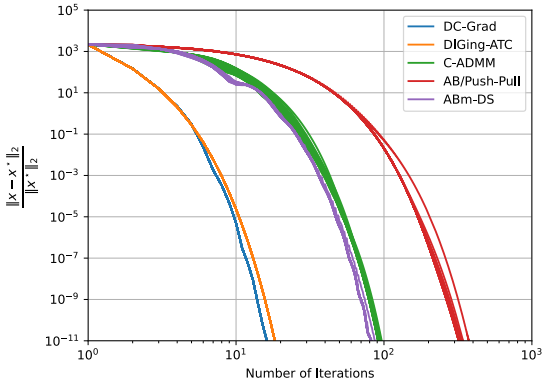

Fig. 1. Convergence error of all agents per iteration in the distributed state estimation problem on a fully-connected communication graph. DC-Grad converges the fastest, closely followed by DIGing-ATC.

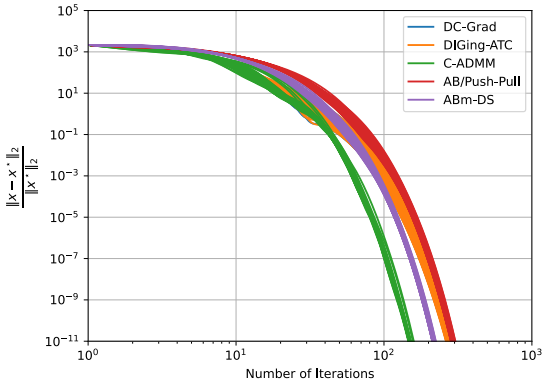

Fig. 2. Convergence error of all agents per iteration in the distributed state estimation problem on a randomly-generated connected communication graph with $\kappa = 0.48$. C-ADMM attains the fastest convergence rate. The convergence plot of DIGing-ATC *overlays* that of DC-Grad, with both algorithms converging at the same rate.

B. Distributed Robust Least-Squares

We consider the robust least-squares formulation of the state estimation problem, presented in Section VII-A. We replace the ℓ_2^2-loss function in (29) with the Huber loss function, given by:

$$f_{\text{hub},\xi}(u) = \begin{cases} \frac{1}{2}u^2, & \text{if } |u| \leq \xi \ (\ell_2^2\text{-zone}), \\ \xi(|u| - \frac{1}{2}\xi), & \text{otherwise } (\ell_1\text{-zone}). \end{cases} \tag{30}$$

We note the Huber loss function is less sensitive to outliers, since the penalty function $f_{\text{hub},\xi}$ grows linearly for large values of u. The corresponding robust least-squares optimization problem is given by:

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \ \frac{1}{N} \sum_{i=1}^{N} f_{\text{hub},\xi}(C_i x - y_i). \tag{31}$$

We assume each agent has a single observation, i.e., $m_i = 1,\ \forall i \in \mathcal{V}$ and assess the convergence rate of the distributed algorithms on randomly-generated connected communication graphs, with $N = 50$ and $n = 10$. We randomly initialize x_i such that the x_i lies in the ℓ_1-zone, $\forall i \in \mathcal{V}$. Further, we randomly generate the problem data such that the optimal solution $x^\star$ lies in the ℓ_2^2-zone. We set the maximum number of iterations to 3000. We note that a closed-form solution does not exist for the primal update procedure for C-ADMM in this problem. Consequently, we do not include C-ADMM in this study, noting that solving the primal update procedure with iterative solvers would negatively impact the computation time of C-ADMM, effectively limiting its competitiveness. Further, we update the conjugate gradient parameter β using a modified *Polak-Ribière Scheme*.

We provide the mean computation time per agent, in seconds, required for convergence of each algorithm, along with the standard deviation in Table III, over 20 randomly-generated problems for each communication network. From Table III, we note that ABm requires the shortest computation time for convergence on more-sparsely-connected communication graphs. However, on more-densely-connected communication graphs, DIGing-ATC achieves the shortest computation time, followed by DC-Grad.

TABLE III

THE MEAN AND STANDARD DEVIATION OF THE CUMULATIVE COMPUTATION TIME (IN SECONDS) PER AGENT IN THE DISTRIBUTED ROBUST-STATE-ESTIMATION PROBLEM.

Algorithm	$\kappa = 0.42$	$\kappa = 0.74$	$\kappa = 0.96$	$\kappa = 1.00$
AB/Push-Pull [16]	$7.55e^{-3} \pm 1.89e^{-3}$	$7.76e^{-3} \pm 1.99e^{-3}$	$8.16e^{-3} \pm 2.04e^{-3}$	$8.63e^{-3} \pm 2.02e^{-3}$
ABm-DS [13]	$8.96e^{-4} \pm 1.66e^{-4}$	$9.06e^{-4} \pm 2.64e^{-4}$	$9.47e^{-4} \pm 2.96e^{-4}$	$9.69e^{-4} \pm 2.75e^{-4}$
DIGing-ATC [14]	$\mathbf{7.43e^{-4} \pm 6.89e^{-5}}$	$\mathbf{4.46e^{-4} \pm 9.93e^{-5}}$	$\mathbf{1.87e^{-4} \pm 4.33e^{-5}}$	$\mathbf{1.71e^{-4} \pm 4.51e^{-5}}$
DC-Grad (ours)	$1.28e^{-3} \pm 1.15e^{-4}$	$7.65e^{-4} \pm 1.75e^{-4}$	$3.05e^{-4} \pm 7.27e^{-5}$	$2.85e^{-4} \pm 6.69e^{-5}$

In Table IV, we show the mean and standard deviation of the cumulative size of messages exchanged by each agent (in MB), in each distributed algorithm. Generally, on more-sparsely-connected graphs, ABm converges the fastest, in terms of the number of iterations, and as a result, incurs the least communication overhead, closely followed by DIGing-ATC and DC-Grad. On the other hand, on more-densely-connected communication graphs, DC-Grad incurs the least communication overhead.

TABLE IV

THE MEAN AND STANDARD DEVIATION OF THE CUMULATIVE SIZE OF MESSAGES EXCHANGED BY EACH AGENT (IN MB) IN THE DISTRIBUTED ROBUST-STATE-ESTIMATION PROBLEM.

Algorithm	$\kappa = 0.42$	$\kappa = 0.74$	$\kappa = 0.96$	$\kappa = 1.00$
AB/Push-Pull [16]	$8.86e^{-1} \pm 2.26e^{-1}$	$9.02e^{-1} \pm 2.30e^{-1}$	$9.80e^{-1} \pm 2.48e^{-1}$	$1.01e^{0} \pm 2.387e^{-1}$
ABm-DS [13]	$1.02e^{-1} \pm 1.85e^{-2}$	$1.02e^{-1} \pm 2.99e^{-2}$	$1.09e^{-1} \pm 3.45e^{-2}$	$1.08e^{-1} \pm 3.14e^{-2}$
DIGing-ATC [14]	$\mathbf{1.14e^{-1} \pm 1.05e^{-2}}$	$\mathbf{6.65e^{-2} \pm 1.48e^{-2}}$	$2.81e^{-2} \pm 6.74e^{-3}$	$2.52e^{-2} \pm 6.70e^{-3}$
DC-Grad (ours)	$1.14e^{-1} \pm 1.06e^{-2}$	$6.65e^{-2} \pm 1.55e^{-2}$	$\mathbf{2.71e^{-2} \pm 6.69e^{-3}}$	$\mathbf{2.47e^{-2} \pm 5.95e^{-3}}$

We show the convergence error of each agent's iterate x_i, per iteration, on a fully-connected communication network in Figure 3. We note that DC-Grad converges within the fewest number of iterations, closely followed by DIGing-ATC. In addition, AB/Push-Pull requires the greatest number of iterations for convergence. We note that AB/Push-Pull utilizes the *combine-then-adapt* update scheme, which results in slower convergence, generally [14]. Moreover, the objective function in (30) is not strongly-convex over its entire domain, particularly in the ℓ_1-zone. In addition, gradient-tracking methods, in general, require (*restricted*) strong convexity for linear convergence. As a result, all the algorithms exhibit sublinear convergence initially, since all the algorithms are initialized with x_i in the ℓ_1-zone, $\forall i \in \mathcal{V}$. The algorithms exhibit linear convergence when the iterates enter the ℓ_2^2-zone, as depicted in Figure 3. In addition, Figure 4 shows the convergence error of each agent's iterates on a randomly-generated communication network with $\kappa = 0.42$. On these graphs, ABm requires the least number of iterations for convergence. We note that the convergence plot for DIGing-ATC overlays that of DC-Grad, with both algorithms exhibiting relatively the same performance.

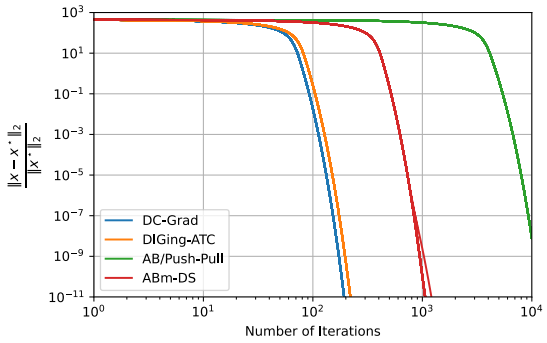

Fig. 3. Convergence error of all agents per iteration in the distributed robust-state-estimation problem on a fully-connected communication network. DC-Grad attains the fastest convergence rate, while AB/Push-Pull attains the slowest convergence rate.

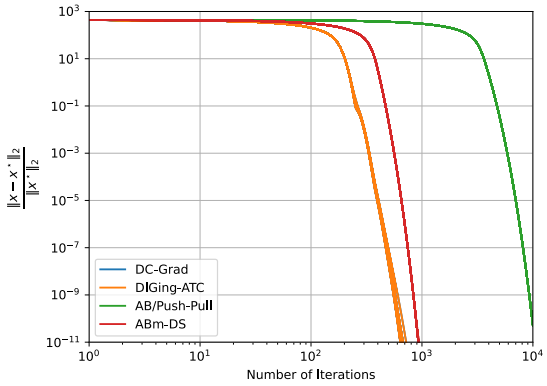

Fig. 4. Convergence error of all agents per iteration in the distributed robust-state-estimation problem on a randomly-generated connected communication network with $\kappa = 0.42$. The convergence plot of DIGing-ATC *overlays* that of DC-Grad, with both methods converging faster than the other methods in this trial, although, in general, ABm converges marginally faster on more-sparsely-connected graphs.

VIII. CONCLUSION

We introduce DC-Grad, a distributed conjugate gradient method, where each agent communicates with its immediate

neighbors to compute an optimal solution of a distributed optimization problem. Our algorithm utilizes only first-order information of the optimization problem, without requiring second-order information. Through simulations, we show that our algorithm requires the least communication overhead for convergence on densely-connected communication graphs, in general, at the expense of a slightly increased computation overhead in comparison to the best-competing algorithm. In addition, on sparsely-connected communication graphs, our algorithms performs similarly to other first-order distributed algorithms. Preliminary convergence analysis of our algorithm suggests that our algorithm converges linearly. In future work, we seek to characterize the convergence rate of our method. Further, in our simulation studies, our algorithm exhibits a notably similar performance with DIGing-ATC. We intend to examine this similarity in future work.

REFERENCES

[1] S.-S. Park, Y. Min, J.-S. Ha, D.-H. Cho, and H.-L. Choi, "A distributed admm approach to non-myopic path planning for multi-target tracking," *IEEE Access*, vol. 7, pp. 163 589–163 603, 2019.

[2] O. Shorinwa, J. Yu, T. Halsted, A. Koufos, and M. Schwager, "Distributed multi-target tracking for autonomous vehicle fleets," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 3495–3501.

[3] I. Necoara, V. Nedelcu, and I. Dumitrache, "Parallel and distributed optimization methods for estimation and control in networks," *Journal of Process Control*, vol. 21, no. 5, pp. 756–766, 2011.

[4] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine learning and systems*, vol. 2, pp. 429–450, 2020.

[5] Y. Wang, S. Wang, and L. Wu, "Distributed optimization approaches for emerging power systems operation: A review," *Electric Power Systems Research*, vol. 144, pp. 127–135, 2017.

[6] W. Tang and P. Daoutidis, "Distributed nonlinear model predictive control through accelerated parallel admm," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 1406–1411.

[7] O. Shorinwa and M. Schwager, "Distributed model predictive control via separable optimization in multi-agent networks," *IEEE Transactions on Automatic Control*, 2023.

[8] A. Nedić, D. P. Bertsekas, and V. S. Borkar, "Distributed asynchronous incremental subgradient methods," *Studies in Computational Mathematics*, vol. 8, no. C, pp. 381–407, 2001.

[9] L. Xiao, S. Boyd, and S.-J. Kim, "Distributed average consensus with least-mean-square deviation," *Journal of parallel and distributed computing*, vol. 67, no. 1, pp. 33–46, 2007.

[10] F. Bénézit, V. Blondel, P. Thiran, J. Tsitsiklis, and M. Vetterli, "Weighted gossip: Distributed averaging using non-doubly stochastic matrices," in *2010 IEEE International Symposium on Information Theory*. IEEE, 2010, pp. 1753–1757.

[11] K. Yuan, Q. Ling, and W. Yin, "On the convergence of decentralized gradient descent," *SIAM Journal on Optimization*, vol. 26, no. 3, pp. 1835–1854, 2016.

[12] W. Shi, Q. Ling, G. Wu, and W. Yin, "EXTRA: An exact first-order algorithm for decentralized consensus optimization," *SIAM Journal on Optimization*, vol. 25, no. 2, pp. 944–966, 2015.

[13] R. Xin and U. A. Khan, "Distributed heavy-ball: A generalization and acceleration of first-order methods with gradient tracking," *IEEE Transactions on Automatic Control*, vol. 65, no. 6, pp. 2627–2633, 2019.

[14] A. Nedic, A. Olshevsky, and W. Shi, "Achieving geometric convergence for distributed optimization over time-varying graphs," *SIAM Journal on Optimization*, vol. 27, no. 4, pp. 2597–2633, 2017.

[15] Q. Lü, X. Liao, H. Li, and T. Huang, "A Nesterov-like gradient tracking algorithm for distributed optimization over directed networks," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2020.

[16] R. Xin, S. Pu, A. Nedić, and U. A. Khan, "A general framework for decentralized optimization with first-order methods," *Proceedings of the IEEE*, vol. 108, no. 11, pp. 1869–1889, 2020.

[17] G. Mateos, J. A. Bazerque, and G. B. Giannakis, "Distributed sparse linear regression," *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5262–5276, 2010.

[18] F. Farina, A. Garulli, A. Giannitrapani, and G. Notarstefano, "A distributed asynchronous method of multipliers for constrained nonconvex optimization," *Automatica*, vol. 103, pp. 243–253, 2019.

[19] M. R. Hestenes, E. Stiefel *et al.*, "Methods of conjugate gradients for solving linear systems," *Journal of research of the National Bureau of Standards*, vol. 49, no. 6, pp. 409–436, 1952.

[20] W. W. Hager and H. Zhang, "A survey of nonlinear conjugate gradient methods," *Pacific journal of Optimization*, vol. 2, no. 1, pp. 35–58, 2006.

[21] Y.-H. Dai and Y. Yuan, "A nonlinear conjugate gradient method with a strong global convergence property," *SIAM Journal on optimization*, vol. 10, no. 1, pp. 177–182, 1999.

[22] G. Yuan, T. Li, and W. Hu, "A conjugate gradient algorithm for large-scale nonlinear equations and image restoration problems," *Applied numerical mathematics*, vol. 147, pp. 129–141, 2020.

[23] G. Yuan, Z. Wei, and Y. Yang, "The global convergence of the polak–ribière–polyak conjugate gradient algorithm under inexact line search for nonconvex functions," *Journal of Computational and Applied Mathematics*, vol. 362, pp. 262–275, 2019.

[24] L. Ismail and R. Barua, "Implementation and performance evaluation of a distributed conjugate gradient method in a cloud computing environment," *Software: Practice and Experience*, vol. 43, no. 3, pp. 281–304, 2013.

[25] P. Lanucara and S. Rovida, "Conjugate-gradient algorithms: An mpi open-mp implementation on distributed shared memory systems," in *First European Workshop on OpenMP*, 1999, pp. 76–78.

[26] R. Helfenstein and J. Koko, "Parallel preconditioned conjugate gradient algorithm on GPU," *Journal of Computational and Applied Mathematics*, vol. 236, no. 15, pp. 3584–3590, 2012.

[27] A. Engelmann and T. Faulwasser, "Essentially decentralized conjugate gradients," *arXiv preprint arXiv:2102.12311*, 2021.

[28] S. Xu, R. C. De Lamare, and H. V. Poor, "Distributed estimation over sensor networks based on distributed conjugate gradient strategies," *IET Signal Processing*, vol. 10, no. 3, pp. 291–301, 2016.

[29] H. Ping, Y. Wang, and D. Li, "Dcg: Distributed conjugate gradient for efficient linear equations solving," *arXiv preprint arXiv:2107.13814*, 2021.

[30] C. Xu, J. Zhu, Y. Shang, and Q. Wu, "A distributed conjugate gradient online learning method over networks," *Complexity*, vol. 2020, pp. 1–13, 2020.

[31] A. H. Sayed, "Diffusion adaptation over networks," in *Academic Press Library in Signal Processing*. Elsevier, 2014, vol. 3, pp. 323–453.

[32] R. Fletcher and C. M. Reeves, "Function minimization by conjugate gradients," *The computer journal*, vol. 7, no. 2, pp. 149–154, 1964.

[33] E. Polak and G. Ribiere, "Note sur la convergence de directions conjugeés. rev. francaise informat," *Recherche Opertionelle, 3e année*, vol. 16, pp. 35–43, 1969.

[34] B. T. Polyak, "The conjugate gradient method in extremal problems," *USSR Computational Mathematics and Mathematical Physics*, vol. 9, no. 4, pp. 94–112, 1969.

[35] M. Zhu and S. Martínez, "Discrete-time dynamic average consensus," *Automatica*, vol. 46, no. 2, pp. 322–329, 2010.

[36] O. Shorinwa and M. Schwager, "Distributed conjugate gradient method via conjugate direction tracking," *arXiv preprint arXiv:2309.12235*, 2023.