

GradConsensus: Linearly Convergent Algorithm for Reducing Disagreement in Multi-Agent Optimization

Vivek Khatana¹, Student Member, IEEE, Govind Saraswat², Senior Member, IEEE, Sourav Patel³, and Murti V. Salapaka⁴, Fellow, IEEE

Abstract—In this article, we propose a new approach, *optimize then agree* for minimizing a sum of convex functions, over a directed graph. The *optimize then agree* approach decouples the optimization step and the consensus step in a multi-agent distributed optimization framework. The key motivation for *optimize then agree* is to guarantee that the disagreement between the agents' estimates at every iteration of the distributed optimization algorithm remains under any *a priori* specified tolerance; existing algorithms do not provide such a guarantee which is required in many practical scenarios. In the proposed algorithm, each agent utilizes its locally available function information along with a finite-time approximate consensus protocol to move toward the optimal solution. We establish a global R-linear rate of convergence if the aggregate function is strongly convex and Lipschitz differentiable. We show that under the relaxed assumption of convex and Lipschitz differentiable functions, a Q-linear rate is achieved until reaching a neighborhood of the optimal, that can be managed using the tolerance value specified on the finite-time approximate consensus protocol; no existing method in the literature has such strong convergence guarantees for not necessarily strongly convex functions. The communication overhead for these improved guarantees of our algorithm is within a log k , (at the k th iteration of the algorithm) factor of the traditional algorithms. Further, we numerically demonstrate the efficacy of the proposed algorithm in solving a distributed logistic regression problem.

Index Terms—Directed graphs, distributed control, distributed optimization, finite-time consensus, linear convergence, multi-agent networks.

I. INTRODUCTION

IN THIS article, we consider a group of n agents connected as a directed graph, with the objective of solving the following distributed optimization problem:

$$\underset{x \in \mathbb{R}^p}{\text{minimize}} \quad f(x) = \sum_{i=1}^n f_i(x), \quad (1)$$

Manuscript received 20 June 2022; revised 16 May 2023; accepted 27 September 2023. Date of publication 6 October 2023; date of current version 8 January 2024. This work was supported by the Advanced Research Projects Agency-Energy OPEN through the project titled “Rapidly Viable Sustained Grid” under Grant DE-AR0001016. Recommended for acceptance by Dr. Jie Gao. (Corresponding author: Vivek Khatana.)

Vivek Khatana and Murti V. Salapaka are with the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455 USA (e-mail: khata010@umn.edu; murtis@umn.edu).

Govind Saraswat is with the Enphase Energy, Austin, TX 78758 USA (e-mail: govind.saraswat@gmail.com).

Sourav Patel is with the Google, Inc., Mountain View, CA 94043 USA (e-mail: patel292@umn.edu).

Digital Object Identifier 10.1109/TNSE.2023.3321757

where, $x \in \mathbb{R}^p$ is a global optimization variable, and each function $f_i : \mathbb{R}^p \rightarrow \mathbb{R}$ is a convex function known only to agent i . Due to the underlying directed interconnection structure, agents can only send (receive) information to (from) their neighboring nodes connected via a unidirectional link in the directed communication structure.

An initial study of the distributed optimization problem was presented in the seminal works [1], [2]. Since then numerous algorithms to minimize the sum of convex functions in a distributed manner are devised [3], [4], [5]. Most of the existing first-order methods for solving (1) include the Distributed Gradient Descent (DGD) method [3], [6] and its modifications [7], [8], [9], [10], [11]. The authors in [8] proposed an exact first-order method that converges to the optimal solution while using a fixed step-size in the gradient updates. The works in [10], [11] have developed algorithms based on proximal-gradient to tackle (1). However, most existing works including, [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], impose restrictive conditions such as the requirement of doubly-stochastic weight matrices and need of balanced undirected graphs. The first work to propose a distributed optimization method for directed graphs appeared in [12]. Subsequently, the authors in [13] incorporated the push-sum protocol [14] with the earlier DGD method [3] to propose the subgradient-push algorithm for time-varying directed graphs. Recent works [15] and [16] proposed distributed optimization schemes for directed networks that make use of both row and column stochastic matrices in one iteration of the algorithm.

While there is proliferating literature on developing distributed optimization methods for (1), the current methods in the literature suffer from the mismatch between the agents' estimates at any finite-time termination of the algorithm. Consider, an equivalent reformulation of problem (1):

$$\underset{x^1, \dots, x^n}{\text{minimize}} \quad \sum_{i=1}^n f_i(x^i) \\ \text{subject to } x^i = x^j, \text{ for all } i, j \in \{1, 2, \dots, n\}, \quad (2)$$

where, $x^i, x^j \in \mathbb{R}^p$ are local estimates (of the optimal solution) of agent i and j respectively and the equality constraints enforce agreement between the local estimates of the agents. The design

of existing algorithms that focus on finding the solution to problem (2) emphasize solving the unconstrained problem and reaching an agreement between the agents' estimates only asymptotically. In particular, at the intermediate iterations of the algorithm, the agents' estimates often have a considerable mismatch and hence, do not provide a practically feasible solution (where a specification is present to have control over the largest mismatch between the agent estimates) to problem (2). The disagreement of agents' estimates (also referred to as consensus constraint violation) can be problematic for practical applications where obtaining a feasible solution with guarantees in finite time is crucial for adequate performance and stability of the system. Some illustrative examples of real-world applications where agreement on a solution during the iterations of the distributed optimization algorithm is imperative include:

A. Rendezvous of Multi-Robot Systems: Consider n mobile robots that have to rendezvous at a common position and a common time. The i th robot has a hard constraint characterized by $u_i(p, t) \leq 0$ for a position p and time t . This constraint, for example, can impose restrictions such as not being able to reach position p by time t . The i th robot also poses a cost $f_i(p, t)$ to the system planner for reaching position p at time t . The related optimization problem is:

$$\begin{aligned} & \underset{(p_1, t_1), \dots, (p_n, t_n)}{\text{minimize}} \quad \sum_{i=1}^n f_i(p_i, t_i) \\ & \text{subject to} \quad p_i = p_j, t_i = t_j, \forall i, j \in \{1, 2, \dots, n\} \\ & \quad u_i(p_i, t_i) \leq 0, \forall i \in \{1, 2, \dots, n\} \end{aligned} \quad (3)$$

The optimal solution to the above optimization can be realized; however, the optimal with the current know-how [17], [18] can be reached only asymptotically. It is possible that the robots cannot wait beyond a certain time and need to start taking action before a pre-specified time. In such scenarios, it is possible that algorithms that do not satisfy the common meeting time and position remain practically infeasible for the operation when stopped before convergence to the optimal is close (which may be hard to assess). The framework proposed in this article ensures that the goal of a common time and position remains feasible at every iteration of the solution procedure. Thus, even when away from optimality, the constraints are practically met.

B. Decentralized Deep Learning: Consider, a distributed empirical risk minimization problem with n nodes considered in [19], [20]. In this problem, the task is to obtain the best estimate x of a parameter of interest, from data. The estimation is to be carried out by n distributed agents communicating over a given network. Agent i has a data set D_i assigned to it. Agent i has a cost function to quantify the quality of fit when the parameter x is used to fit a data z in D_i given by $f_i(x, z)$ (for example, one such function is $\|x - z\|^2$). All agents have to agree on the estimate of the parameter at the end of the learning process. The associated optimization problem (also see [19], [20]) is

$$\begin{aligned} & \underset{x_1, \dots, x_n}{\text{minimize}} \quad \sum_{i=1}^n \sum_{z \in D_i} f_i(x_i, z) \\ & \text{subject to} \quad x_i = x_j, \forall i, j \in \{1, 2, \dots, n\} \end{aligned} \quad (4)$$

In the context of problem (4), it is important to balance the individual model parameter optimality (via local optimization by the i th agent) and the global constraint of consensus between the model parameter estimates of all the agents during the model parameter learning procedure (typically the neural network training procedure) to recover an estimate of the true model from the distributed datasets [19], [20]. Here, a framework to improve the inter-iteration behavior of the estimates to agree is important (see [21]). Our article meets the gap in the know-how of assuring that the estimates practically agree at every iteration of the solution procedure.

More applications requiring finite-iteration feasibility that the authors are aware of arise in power systems, where the reactive power control, economic dispatch, and optimal power flow problems are modeled as distributed optimization problems [22], [23].

In this article, we introduce a new framework for designing distributed optimization algorithms to solve problem (2). Our approach is motivated towards reducing the inaccuracy of the consensus step in the existing distributed optimization algorithms in the literature. In particular, we propose an *optimize then agree* framework to decouple the gradient-descent step from the consensus step, that are used in tandem in most existing distributed-optimization algorithms (see [3], [5], [8], [9], [10], [16] for example), to reduce the mismatch between the estimates maintained by the agents during every iteration of the algorithm. Based on the *optimize then agree* framework we develop a novel first-order distributed optimization method, termed as Gradient-Consensus (abbreviated as **GradConsensus**) algorithm. Each iteration of the **GradConsensus** algorithm comprises two steps: a local gradient-descent step at each agent followed by a finite-time approximate consensus protocol. The finite-time approximate consensus protocol is designed such that after the consensus step the estimates of all agents are ε -close to each other, where ε is any desired level of accuracy.

We remark here that a few works [24], [25], [26], [27] have explored the idea of utilizing multiple communication steps; we direct the reader to [24], [25], [26], [27] for the motivation of the multiple communication steps. The protocols in [24], [25], [26], [27] depend on a symmetric double stochastic weight matrix, the synthesis of which requires global information of the network making them unsuitable to be implemented in directed networks. The scheme proposed in this article is amenable to directed networks and does not rely on doubly stochastic matrices; thus the synthesis of the algorithm and its implementation does not require centralized information on the communication network. Further, the works in [24], [25], [26], [27] do not take the consensus constraint satisfaction into consideration. On the contrary, we focus on guaranteed control of the mismatch between the agents' estimates to a pre-specified value. The number of communication steps performed during each iteration gets fixed by the tolerance of the finite-time approximate consensus protocol. Utilizing the finite-time approximate consensus protocol (leading to better control of agents' estimates mismatch) instead of arbitrarily increasing the number of communication steps has advantages both in terms of the lesser total communication cost and better-controlled performance of the optimization

algorithm (see numerical results in Section V). Furthermore, we rigorously establish stronger convergence rate estimates with a novel Q-linear rate of convergence for general convex functions (Theorem 4.2). The main contributions of this article are as follows:

1. We focus on minimizing the disagreement between the estimates of all agents in the existing distributed optimization algorithms. Due to the mismatch between the agent estimates, at any finite iteration, the solution obtained by the existing algorithms is not a feasible solution to problem (2). For applications where getting a feasible solution is critical during every time-instant of operation (a few examples are presented above), the existing algorithms do not provide a viable solution.

To address this issue, we present a novel *optimize then agree* framework with the motivation of decoupling the consensus and optimization steps used in tandem in existing distributed optimization schemes. Our framework provides a new perspective on solving distributed optimization problems and presents guidelines for the development of new algorithms. Based on this framework we develop a first-order distributed algorithm termed as **GradConsensus** to solve distributed optimization problems.

2. The proposed **GradConsensus** algorithm has several nice features:

- 1) It is suitable for directed graphs allowing for non-symmetric communication topologies. This feature considerably widens the applicability as in most application scenarios, agents do not have the same range of communication.
- 2) It utilizes *only* the knowledge of the out-degree of each agent in the graph and is based on a column stochastic matrix which is amenable for distributed synthesis (we use the same notion of distributed synthesis as detailed in [28], Appendix A) where centralized information of the network connectivity structure is not required.
- 3) To the best of the authors' knowledge, **GradConsensus** algorithm has the best convergence rate under the assumption of individual functions f_i being not necessarily strongly convex and Lipschitz differentiable. We show that (Theorem 4.2) the **GradConsensus** algorithm converges at a novel Q-linear rate to a small neighborhood of the optimal solution. Most existing analysis of the first-order gradient-based distributed optimization algorithms under these assumptions have a sub-linear rate of convergence. The improved guarantees accrue with an overhead of within $\log k$ communication steps of the optimal $O(k)$ bound by the k th iterate.

The **GradConsensus** algorithm has a global R-linear rate (Theorem 4.3) of convergence to the optimal solution when the aggregate function f is strongly-convex and Lipschitz differentiable.

- 4) Empirical results corroborate that the proposed **GradConsensus** algorithm performs very well in practice in terms of the number of gradient computations and the total run-time required.

The authors have introduced a preliminary version of the **GradConsensus** algorithm developed here in [29]. The current work presents a significant generalization, with the *optimize then agree* framework. Compared to the preliminary conference version here, on the theoretical front we: (i) present convergence guarantees (Theorems 4.1–4.3) for the **GradConsensus** algorithm along with detailed proofs, (ii) provide the explicit analysis (Lemma 3.1, (29) and Remarks 5, 6) to account quantitatively for the number of communication steps required by the proposed algorithm, (iii) establish new theoretical results (Theorems 4.1, 4.2) under milder assumptions to provide explicit performance guarantees for the **GradConsensus** under a broader class of problems. On the empirical side, to substantiate the applicability of the **GradConsensus** algorithm, we provide detailed numerical tests on the distributed logistic regression problem and present a comparison with other algorithms in the literature.

The rest of the article is organized as follows. Section II provides the definitions and notations used in the article. In Section III, we describe the *optimize then agree* framework and the proposed **GradConsensus** algorithm. Along with this, we present a discussion on its design and provide some supporting analysis to be used later. We establish the convergence guarantees for the **GradConsensus** algorithm under two different sets of assumptions in Section IV. In Section V, we present the numerical simulation results in solving a distributed logistic regression problem using the **GradConsensus** algorithm along with a comparison with other algorithms in the literature. Section VI provides the concluding remarks and some future research directions.

II. DEFINITIONS AND NOTATIONS

Definition 1. (Directed Graph): A directed graph $\mathcal{G}$ is a pair $(\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is a set of vertices (or nodes) and $\mathcal{E}$ is a set of edges, which are ordered subsets of two distinct elements of $\mathcal{V}$. If an edge from $j \in \mathcal{V}$ to $i \in \mathcal{V}$ exists then it is denoted as $(i, j) \in \mathcal{E}$.

Definition 2. (Strongly Connected Graph): A directed graph is strongly connected if and only if it has a directed path between each pair of distinct nodes k and ℓ .

Definition 3. (Diameter of a Graph): It is the longest shortest directed path between any two nodes of the directed graph $\mathcal{G}$.

Definition 4. (In-Neighborhood): The set $N_i^{in} = \{j \mid (i, j) \in \mathcal{E}\}$ of in-neighbors of node $i \in \mathcal{V}$ not including the node i , is called the in-neighborhood of node i .

Definition 5. (Column Stochastic Matrix): A $n \times n$ real matrix $\mathbf{P} = [p_{ij}]$ is column-stochastic matrix if $\sum_{i=1}^n p_{ij} = 1$ where, $0 \leq p_{ij} \leq 1$, for $1 \leq i, j \leq n$.

Definition 6. (Lipschitz Differentiability): A differentiable function f is called Lipschitz differentiable with constant L , if there exists $L > 0$ such that:

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \forall x, y \in \text{dom} f.$$

Definition 7. (Strongly Convex Function): A differentiable function f is called strongly convex with parameter σ , if there

exists $\sigma > 0$ such that for all x, y in the domain of f :

$$\langle \nabla f(x) - \nabla f(y), x - y \rangle \geq \sigma \|x - y\|^2, \forall x, y \in \text{dom} f.$$

A. Notations

Each agent $i \in \mathcal{V}$ maintains a local estimate $x^i(k) \in \mathbb{R}^p$ at iteration k of the **GradConsensus** algorithm. Let $\mathcal{X}^* = \{x \in \mathbb{R}^p \mid \sum_{i=1}^n f_i(x) = f^*\}$ denote the set of solutions to (1), with f^* being the optimal objective function value. We use a directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ with n nodes, to model the network interconnection between the n agents. $\mathcal{D}$ denotes an upper bound on the diameter of the graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$. $\|x\|_1$ and $\|x\|$ denote the 1-norm and 2-norm of the vector $x \in \mathbb{R}^p$ respectively. The notation $\lceil \cdot \rceil$ denotes the least integer function or the ceiling function, defined as: given $x \in \mathbb{R}$, $\lceil x \rceil = \min\{m \in \mathbb{Z} \mid m \geq x\}$, where $\mathbb{Z}$ is the set of integers. We use $\mathbf{1}$ to denote the vector of ones.

III. THE OPTIMIZE THEN AGREE FRAMEWORK

In this section, we focus on problem (2) and present the proposed *optimize then agree* framework. We emphasize that the proposed framework can be applied to any existing algorithm in the literature that utilizes an optimization scheme in tandem with a consensus step for state updates. Here we utilize the classical DGD method [3] for an explanation. During the iteration k of DGD every agent i updates its local estimate $x^i(k) \in \mathbb{R}^p$ as follows:

$$x^i(k) = \sum_{j \in N_i^{\text{in}} \cup \{i\}} p_{ij} x^j(k-1) - \alpha \nabla f_i(x^i(k-1)), \quad (5)$$

where, the weights $0 \leq p_{ij} \leq 1$ are such that $\sum_j p_{ij} = \sum_i p_{ij} = 1$. The first term $\sum_{j \in N_i^{\text{in}} \cup \{i\}} p_{ij} x^j(k-1)$ in (5) corresponds to a local consensus step and the second term $-\alpha \nabla f_i(x^i(k-1))$ denotes a local gradient step. Notice that due to a local consensus step the update (5) does not account for the agreement between the estimates over the entire network of agents during iteration k . If update (5) is terminated after some finite number of iterations the solution estimates have a disagreement and does not satisfy the equality constraints in problem (2) and hence the (finite-time) solution generated is not feasible for the original problem. To address this issue we propose the *optimize then agree* framework where the optimization step (gradient-descent in this case) and the consensus scheme are decoupled from each other. The optimize then agree framework enforces the mismatch between the agent estimates to remain below any specified threshold (and hence controls the consensus constraint violation). Here, each agent employs an optimization scheme to steer towards the solution of its local sub-problem and utilizes a consensus protocol to achieve agreement with the local estimates of all the other agents. To describe the *optimize then agree* framework we define, the optimization scheme operator used by agent $i \in \mathcal{V}$ as the mapping $\mathcal{O}_i : \mathbb{R}^p \rightarrow \mathbb{R}^p$:

$$z^i(k) := \mathcal{O}_i(x^i(k-1)). \quad (6)$$

Similarly, define the consensus scheme as the operator mapping $\mathcal{C} : \mathbb{R}^{p \times n} \rightarrow \mathbb{R}^{p \times n}$:

$$x(k) := \mathcal{C}(z(k)), \quad (7)$$

Algorithm 1: Optimize then agree framework.

Repeat for $k = 1, 2, \dots$

/* state updates using operators (6)
and (7) */

$z^i(k) = \mathcal{O}_i(x^i(k-1)), \forall i \in \mathcal{V}$

$x(k) = \mathcal{C}(z(k))$

until a stopping criterion is met

where, $z(k) = [z^1(k) \dots z^n(k)]$, $x(k) = [x^1(k) \dots x^n(k)] \in \mathbb{R}^{p \times n}$, such that $x^1(k) = x^2(k) = \dots = x^n(k)$. Utilizing the above notation the *optimize then agree* framework leads to:

Choosing first-order gradient descent with a constant step-size α as the optimization scheme, Algorithm 1 at any iteration $k+1$ results in the following updates:

$$\begin{aligned} z^i(k+1) &= x^i(k) - \alpha \nabla f_i(x^i(k)), \forall i \in \mathcal{V}, \\ x(k+1) &= \mathcal{C}(z(k+1)), \end{aligned} \quad (8)$$

with the consensus operator implemented in a centralized manner (with the entire vector $z(k+1)$ as the input). However, in order to make Algorithm 1 applicable to distributed networks, we need to distributively realize (8). To this end, one natural choice is to employ a distributed average consensus protocol [14]. Although the average consensus protocols lead to agreement among the agents to the initial average, the result holds only asymptotically which is a potential problem for an efficient implementation of Algorithm 1. To alleviate this, the consensus scheme operator is chosen to be a distributed finite-time approximate consensus protocol, which we call the ε -consensus protocol. The ε -consensus protocol produces a desired level of agreement among the estimates of all agents which is determined by a parameter $\varepsilon > 0$ independent of problem (1) that can be chosen arbitrarily small. In the following section, we first discuss the ε -consensus protocol and then introduce the proposed **GradConsensus** algorithm.

Remark 1: We emphasize that the updates in Algorithm 1 are general and do not depend on a particular choice of the optimization scheme or the consensus algorithm, thus the optimize then agree framework provides a guideline for further exploration of distributed optimization algorithms. The framework of Algorithm 1 can also be seen as an outer optimization loop and an inner consensus loop structure algorithm. Thus, existing schemes [24], [25], [26], [27] are special cases of the optimize then agree framework with a particular choice of optimization scheme and the number of communication steps.

A. ε -consensus Protocol

The ε -consensus protocol is a distributed finite-time terminated average consensus algorithm for agreement on vector-valued states. The protocol presented here is an extension to higher dimensions, of the earlier work [30] by the authors in relation to consensus on scalar values. Here, each agent $i \in \mathcal{V}$ has a vector $z^i \in \mathbb{R}^p$. The objective is to find an approximate estimate of the average $\bar{z} = \frac{1}{n} \sum_{i=1}^n z^i \in \mathbb{R}^p$. To achieve this objective an iterative procedure is devised where each agent

$j \in \mathcal{V}$ maintains a state variable $s^j(k) \in \mathbb{R}^p$ and an auxiliary variable $t_j(k) \in \mathbb{R}$, with the following initialization: $s^j(0) = z^j$, $t_j(0) = 1$, for all $j \in \mathcal{V}$. Every agent $j \in \mathcal{V}$ updates its state and auxiliary variables according to:

$$s^j(k+1) = p_{jj}s^j(k) + \sum_{\ell \in N_j^{\text{in}}} p_{j\ell} s^\ell(k) \quad (9)$$

$$t_j(k+1) = p_{jj}t_j(k) + \sum_{\ell \in N_j^{\text{in}}} p_{j\ell} t_\ell(k) \quad (10)$$

$$r^j(k+1) = \frac{1}{t_j(k+1)} s^j(k+1). \quad (11)$$

We make the following assumption on the graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ and the associated matrix $\mathbf{P} = [p_{ij}]$:

Assumption 1: The directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is strongly-connected. The associated weighted adjacency matrix $\mathbf{P} = [p_{ij}]$ is a column-stochastic matrix. In particular, $0 \leq p_{ij} \leq 1$ and $\sum_{i=1}^n p_{ij} = 1$ for all $j \in \mathcal{V}$.

One choice of weights that satisfy Assumption 1 is the equal neighbor weights rule [31].

The convergence of the state $r^j(k) := \frac{1}{t_j(k)} s^j(k)$, $j \in \mathcal{V}$ evolving according to (9)–(11), to the average $\bar{z} = \frac{1}{n} \sum_{i=1}^n z^i$ has been established rigorously [14], [32]. We state the following result for updates (9)–(11):

Theorem 3.1 ([32]): Let Assumption 1 and the updates (9)–(11) hold. Let the initial conditions of the state and auxiliary variables s^j and t_j be $s^j(0) = z^j$, $t_j(0) = 1$, for all $j \in \mathcal{V}$. Then $\lim_{k \rightarrow \infty} r^j(k) = \frac{1}{n} \sum_{i=1}^n z^i$, for all $j \in \mathcal{V}$.

We now provide a criterion for determining when consensus within a tolerance $\varepsilon > 0$ is reached. Let $\mathbf{r}(k) = [r^1(k) \ r^2(k) \ \dots \ r^n(k)] \in \mathbb{R}^{p \times n}$. At any iteration k , define the maximum $\bar{\mathbf{r}}(k)$ and minimum $\underline{\mathbf{r}}(k)$ state variable of the network over all the agents as:

$$\bar{\mathbf{r}}(k) := \left[\max_{1 \leq j \leq n} \{r_{[1j]}(k)\} \ \dots \ \max_{1 \leq j \leq n} \{r_{[pj]}(k)\} \right] \quad (12)$$

$$\underline{\mathbf{r}}(k) := \left[\min_{1 \leq j \leq n} \{r_{[1j]}(k)\} \ \dots \ \min_{1 \leq j \leq n} \{r_{[pj]}(k)\} \right], \quad (13)$$

where, $\mathbf{r}_{[ij]}(k)$ is i th-row and j th-column entry of $\mathbf{r}(k)$. Let $\{\bar{\mathbf{r}}(u\mathcal{D})\}_{u \geq 1}$ and $\{\underline{\mathbf{r}}(u\mathcal{D})\}_{u \geq 1}$ denote sub-sequences of the sequences $\bar{\mathbf{r}}(k)$ and $\underline{\mathbf{r}}(k)$ respectively, sampled at iterations of the form $u\mathcal{D}$, where, $u = 1, 2, \dots$ and $\mathcal{D}$ is an upper bound on the diameter of the graph $\mathcal{G}$. Under Assumption 1, sub-sequences $\bar{\mathbf{r}}(u\mathcal{D})$ and $\underline{\mathbf{r}}(u\mathcal{D})$ converge to the average of the initial values, $\bar{z} = \frac{1}{n} \sum_{i=1}^n z^i$. Further, the sub-sequences $\bar{\mathbf{r}}(u\mathcal{D})$ and $\underline{\mathbf{r}}(u\mathcal{D})$ are strictly monotonically decreasing and increasing in the index u see [32], Theorem 4.1).

To distributively determine sub-sequences $\{\bar{\mathbf{r}}(u\mathcal{D})\}_{u \geq 1}$ and $\{\underline{\mathbf{r}}(u\mathcal{D})\}_{u \geq 1}$ each agent $j \in \mathcal{V}$, maintains two variables $M^j(k)$, $m^j(k) \in \mathbb{R}^p$ at any iteration k that are initialized as

$$M^j(0) = m^j(0) = z^j, \ \forall j \in \mathcal{V}, \quad (14)$$

and updated as:

$$M^j(k+1) = \left[\max_{\ell \in N_j^{\text{in}} \cup \{j\}} M_1^\ell(k) \ \dots \ \max_{\ell \in N_j^{\text{in}} \cup \{j\}} M_p^\ell(k) \right] \quad (15)$$

$$m^j(k+1) = \left[\min_{\ell \in N_j^{\text{in}} \cup \{j\}} m_1^\ell(k) \ \dots \ \min_{\ell \in N_j^{\text{in}} \cup \{j\}} m_p^\ell(k) \right], \quad (16)$$

where, $M_i^\ell(k)$, and $m_i^\ell(k)$ are the i th entries of the vectors $M^\ell(k)$ and $m^\ell(k)$ respectively. It can be rigorously established that (see [33], Proposition 2.3) the variables $M^j(k)$ and $m^j(k)$ for all $j \in \mathcal{V}$, initialized via the rule (14) converge to $\bar{\mathbf{r}}(0)$ and $\underline{\mathbf{r}}(0)$ respectively in finite number of iterations $k_f \leq \mathcal{D}$.

The above discussion leads to a method for finding the sequences $\{\bar{\mathbf{r}}(u\mathcal{D})\}_{u \geq 1}$ and $\{\underline{\mathbf{r}}(u\mathcal{D})\}_{u \geq 1}$ distributedly: re-initializing the updates (15), (16) for all agents $j \in \mathcal{V}$, at every iteration of the form $k = u\mathcal{D}$, $u = 1, 2, \dots$, to the variable $r^j(u\mathcal{D})$ allow the variables M^j and m^j to converge to $\bar{\mathbf{r}}((u-1)\mathcal{D})$ and $\underline{\mathbf{r}}((u-1)\mathcal{D})$ respectively, after a finite number of iterations $k_f \leq \mathcal{D}$. Since, both the sequences $\{\bar{\mathbf{r}}(u\mathcal{D})\}_{u \geq 1}$ and $\{\underline{\mathbf{r}}(u\mathcal{D})\}_{u \geq 1}$ converge to $\bar{z}$, the norm of the difference between the two, i.e., $\|\bar{\mathbf{r}}(u\mathcal{D}) - \underline{\mathbf{r}}(u\mathcal{D})\|$ also converges to zero. Therefore, given $\varepsilon > 0$, the quantity $\|\bar{\mathbf{r}}(u\mathcal{D}) - \underline{\mathbf{r}}(u\mathcal{D})\|$ eventually becomes less than ε . Based on the above observations we propose an algorithm that allows all agents to simultaneously converge to an ε -close estimate of $\bar{z}$ with ε being any arbitrary pre-specified tolerance:

For $u = 1, 2, \dots$, let $\gamma^j(u\mathcal{D}) := M^j(u\mathcal{D}) - m^j(u\mathcal{D})$. Each agent computes the norm $\|\gamma^j(u\mathcal{D})\|$ every $\mathcal{D}$ iterations and compares it with ε . If $\|\gamma^j(u\mathcal{D})\| := \|M^j(u\mathcal{D}) - m^j(u\mathcal{D})\| < \varepsilon$ at any iteration $u\mathcal{D}$, then agent j , knowing its estimate $r^j(u\mathcal{D})$ is ε -close to $\bar{z}$, terminates the consensus protocol updates (9)–(11) at $u\mathcal{D}$. The variable $r^j(u\mathcal{D})$ is the ε -close estimate of $\bar{z}$ available with agent j .

The following Lemma provides the number of communication steps required by the ε -consensus protocol to converge to an ε -close solution.

Lemma 3.1: Given, consensus tolerance $\varepsilon > 0$, under Assumption 1, after $k \geq \left\lceil \frac{\log(1/\varepsilon)}{-\log(\lambda)} + \frac{\log(C\|\mathbf{s}(0)\|_1/\delta)}{-\log(\lambda)} \right\rceil$ iterations of the consensus protocol (9)–(11) we have:

$$\left\| r^i(k) - \frac{1}{n} \sum_{i=1}^n s^i(0) \right\| \leq \varepsilon, \ \forall i \in \mathcal{V},$$

where, $\mathbf{s}(0) := [s^1(0)^\top \ s^2(0)^\top \ \dots \ s^n(0)^\top]^\top \in \mathbb{R}^{n \times p}$ and $C, \delta, \lambda \in (0, 1)$ are constants related to the properties of the matrix $\mathbf{P}$ and graph $\mathcal{G}$.

Proof: Using (9) it is easy to see that, $k \geq 0$,

$$\mathbf{s}(k) = \mathbf{P}^k \mathbf{s}(0), \quad (17)$$

where, $\mathbf{s}(k) := [s^1(k)^\top \ s^2(k)^\top \ \dots \ s^n(k)^\top]^\top \in \mathbb{R}^{n \times p}$. Since, $\mathbf{P}$ is column stochastic, we have,

$$\mathbf{1}^\top \mathbf{s}(k) = \mathbf{1}^\top \mathbf{s}(0). \quad (18)$$

From Corollary 2, [13] there exists stochastic vector $\phi_k \in \mathbb{R}^n$ and constants C_1 and $\lambda \in (0, 1)$ such that for all $i, j = 1, \dots, n$,

$$|\mathbf{P}_{ij}^k - \phi_{kj}| \leq C_1 \lambda^k, \text{ for all } k \geq 0. \quad (19)$$

From (17) and (18) we have,

$$\begin{aligned} (\mathbf{P} - \phi_k \mathbf{1}^\top) \mathbf{s}(k) &= (\mathbf{P}^{k+1} - \phi_k \mathbf{1}^\top) \mathbf{s}(0) \\ \Rightarrow \mathbf{P} \mathbf{s}(k) &= \phi_k \mathbf{1}^\top \mathbf{s}(k) + (\mathbf{P}^{k+1} - \phi_k \mathbf{1}^\top) \mathbf{s}(0) \\ \Rightarrow \mathbf{s}(k+1) &= \phi_k \mathbf{1}^\top \mathbf{s}(k) + (\mathbf{P}^{k+1} - \phi_k \mathbf{1}^\top) \mathbf{s}(0) \end{aligned} \quad (20)$$

Similarly, for (10) we can write,

$$\mathbf{t}(k+1) = \phi_k \mathbf{1}^\top \mathbf{t}(k) + (\mathbf{P}^{k+1} - \phi_k \mathbf{1}^\top) \mathbf{t}(0), \quad (21)$$

where, $\mathbf{t}(k) := [t_1(k)^\top \ t_2(k)^\top \ \dots \ t_n(k)^\top]^\top \in \mathbb{R}^n$. Recall, $\mathbf{t}(0) = \mathbf{1}$. From (20) and (21) we obtain for every $k \geq 1$ and any agent $i \in \mathcal{V}$,

$$r^i(k+1) = \frac{\phi_{ki} \mathbf{1}^\top \mathbf{s}(k) + [(\mathbf{P}^{k+1} - \phi_k \mathbf{1}^\top) \mathbf{s}(0)]_i}{n \phi_{ki} + [(\mathbf{P}^{k+1} - \phi_k \mathbf{1}^\top) \mathbf{1}]_i}.$$

Therefore,

$$r^i(k+1) - \frac{\mathbf{1}^\top \mathbf{s}(k)}{n} = \frac{\phi_{ki} \mathbf{1}^\top \mathbf{s}(k) + [(\mathbf{P}^{k+1} - \phi_k \mathbf{1}^\top) \mathbf{s}(0)]_i}{n \phi_{ki} + [(\mathbf{P}^{k+1} - \phi_k \mathbf{1}^\top) \mathbf{1}]_i} - \frac{\mathbf{1}^\top \mathbf{s}(k)}{n}.$$

By rearranging the fraction, we get

$$\begin{aligned} r^i(k+1) - \frac{\mathbf{1}^\top \mathbf{s}(k)}{n} &= \frac{n[(\mathbf{P}^{k+1} - \phi_k \mathbf{1}^\top) \mathbf{s}(0)]_i - \mathbf{1}^\top \mathbf{s}(k)[(\mathbf{P}^{k+1} - \phi_k \mathbf{1}^\top) \mathbf{1}]_i}{n(n \phi_{ki} + [(\mathbf{P}^{k+1} - \phi_k \mathbf{1}^\top) \mathbf{1}]_i)}. \end{aligned}$$

Let $\delta := \min_k \min_{1 \leq i \leq n} \sum_j [\mathbf{P}^k]_{ij}$. Note, that there exists $\underline{\delta} > 0$ (see [13], [34]) such that $\delta > \underline{\delta}$. Therefore, for all i and $k \geq 0$, we have,

$$\begin{aligned} \left| r^i(k+1) - \frac{\mathbf{1}^\top \mathbf{s}(k)}{n} \right| &\leq \frac{\sqrt{n} \max_j (|\mathbf{P}^{k+1} - \phi_k \mathbf{1}^\top|_{ij}) (\|\mathbf{s}(0)\|_1 + \|\mathbf{1}^\top \mathbf{s}(k)\|)}{\underline{\delta}}. \end{aligned} \quad (22)$$

Using, (19) and (22), with $C = 2\sqrt{n}C_1$ we get,

$$\left| r^i(k) - \frac{\mathbf{1}^\top \mathbf{s}(k-1)}{n} \right| \leq \frac{C \|\mathbf{s}(0)\|_1 \lambda^k}{\delta}.$$

Note, that due to Assumption 1, $\frac{\mathbf{1}^\top \mathbf{s}(k-1)}{n} = \frac{\mathbf{1}^\top \mathbf{s}(0)}{n}$. Therefore, for $k \geq \left\lceil \frac{\log(1/\varepsilon)}{-\log(\lambda)} + \frac{\log(C \|\mathbf{s}(0)\|_1 / \delta)}{-\log(\lambda)} \right\rceil$,

$$\left| r^i(k) - \frac{1}{n} \sum_{l=1}^n s^l(0) \right| \leq \varepsilon, \quad \forall i \in \mathcal{V}.$$

□

B. The **GradConsensus** Algorithm

Here, each agent i maintains two estimates: an optimization variable $x^i(k) \in \mathbb{R}^p$ and a local update variable $z^i(k) \in \mathbb{R}^p$ at any iteration k . Every iteration k proceeds in two steps: first each agent i updates $z^i(k)$ via a gradient descent update based on its own function f_i at $x^i(k-1)$. At the next step, the optimization variable $x^i(k)$ is updated to an estimate which is $\varepsilon(k)$ -close to the average value $\hat{z}(k) := \frac{1}{n} \sum_{i=1}^n z^i(k)$, i.e. $\|x^i(k) - \hat{z}(k)\| \leq \varepsilon(k)$, using the distributed ε -consensus protocol, initialized with $z^i(k)$ as the initial condition for the agent i and tolerance $\varepsilon(k)$. Thus, at each iteration k , every individual agent $i \in \mathcal{V}$ performs the following updates:

$$z^i(k) = x^i(k-1) - \alpha \nabla f_i(x^i(k-1)) \quad (23)$$

$$x^i(k) = r^i(\theta_k), \text{ with} \quad (24)$$

$$\|r^i(\theta_k) - \hat{z}(k)\| < \varepsilon(k), \quad \hat{z}(k) := \frac{1}{n} \sum_{i=1}^n z^i(k), \quad (25)$$

where, θ_k denotes the number of communication steps in the ε -consensus protocol at iteration k of the **GradConsensus** algorithm, and $r^i(\theta_k)$ is the approximate estimate of the average $\hat{z}(k)$ produced by the ε -consensus protocol. We make the following assumptions:

Assumption 2: For problem (1) the optimal objective function value f^* is finite and the optimal solution set $\mathcal{X}^*$ is non-empty.

Assumption 3:

i) For all $i \in \mathcal{V}$, f_i is a proper convex function.

ii) Each f_i is Lipschitz differentiable with constant $L_{f_i} > 0$.

Proposition 3.1: Let Assumptions 1–3 hold. Let $\alpha \leq 1/L_f$, where, $L_f := \sum_{i=1}^n L_{f_i}$. Assume, that the set of minimizing solutions of the function f_i is non-empty and bounded for all $i \in \mathcal{V}$. Then, there exists constant $\bar{R} < \infty$ such that $\|z^i(k)\| \leq \bar{R}$ for all k and $i = 1, \dots, n$.

Proof: Under Assumption 3, for any $x \in \mathbb{R}^p$ at any k ,

$$\begin{aligned} f_i(x) &\leq f_i(x^i(k-1)) + \langle \nabla f_i(x^i(k-1)), x - x^i(k-1) \rangle \\ &\quad + \frac{1}{2\alpha} \|x - x^i(k-1)\|^2 \end{aligned}$$

Substituting, $x = x^i(k-1) - \alpha \nabla f_i(x^i(k-1))$ we get,

$$f_i(z^i(k)) \leq f_i(x^i(k-1)), \text{ for all } k. \quad (26)$$

Without loss of generality assume, $x^i(0) = 0$, $f_i(0) < \infty$ and $\|\nabla f_i(0)\| \leq h_i < \infty$, for all $i \in \mathcal{V}$. Under Assumption 3,

$$f_i(x^i(k)) \leq f_i(0) + h_i \|x^i(k)\| + \frac{1}{2\alpha} \|x^i(k)\|^2, \quad \forall k. \quad (27)$$

From (24) and (25), for all k ,

$$\|x^i(k)\| \leq \|\hat{z}(k)\| + \varepsilon(k). \quad (28)$$

Note that all the level sets of f_i are bounded since the set of minimizing solutions of f_i is non-empty and bounded (See [35], Proposition B.9). Next, we prove that the $\|z^i(k)\|$ is bounded for $i = 1, \dots, n$ and for all k . We use an induction argument. Assume, there exists, R_i^k such that $\|z^i(k)\| \leq R_i^k \Rightarrow \|\hat{z}(k)\| \leq R_i^k$. From, (28), $\|x^i(k)\| \leq R_i^k + \varepsilon(k) \leq R_i^k + \bar{\varepsilon}$, where, $\bar{\varepsilon} := \sup_k \varepsilon(k)$. From (27), $f_i(x^i(k)) \leq f_i(0) + h_i(R_i^k + \bar{\varepsilon}) + \frac{1}{2\alpha} \|R_i^k + \bar{\varepsilon}\|^2 < \infty$. Thus, from (26) and the fact that all the level sets of f_i are bounded $\|z^i(k+1)\| \leq R_i^{k+1}$, where, R_i^{k+1} is diameter of the $(f_i(0) + h_i(R_i^k + \bar{\varepsilon}) + \frac{1}{2\alpha} \|R_i^k + \bar{\varepsilon}\|^2)$ -level set of f_i . Note, that $f_i(x^i(0)) = f_i(0) < \infty \Rightarrow \|z^i(1)\| \leq R_i^1$. Thus, the induction hypothesis can be applied recursively. Hence, induction holds. Therefore,

$$\|z^i(k)\| \leq \bar{R}, \text{ for } i = 1, \dots, n, \text{ and for all } k,$$

where, $\bar{R} := \max_{1 \leq i \leq n, k} R_i^k < \infty$. □

Remark 2: Proposition 3.1 assumes that the functions f_i have bounded minimizing solutions. Note this is not a restriction as in most applications the gradient descent minimization step is performed over a large enough compact set. In that case, f_i being proper convex functions have bounded minimizing solutions. Most often in practical applications, the solution that minimizes the objective function f_i in problem (1) is bounded.

Since, the **GradConsensus** algorithm utilizes the ε -consensus protocol at every iteration k with the initialization

Algorithm 2: GradConsensus Algorithm.**Input:**Consensus tolerances $\{\varepsilon(k)\}_{k \geq 0}$ and step-size α **Initialize:**Each agent $j \in \mathcal{V}$ initializes its estimates as

$$x^j(0) = z^j(0) = 0.$$

Repeat for $k = 1, 2, \dots$ **for** $j = 1, 2, 3, \dots, n$, (**In parallel**) **do**

gradient descent iteration:

$$z^j(k) = x^j(k-1) - \alpha \nabla f_j(x^j(k-1))$$

consensus iterations:

$$x^j(k) \leftarrow \varepsilon(k)\text{-consensus}(z^j(k), j \in \mathcal{V})$$

end**until** a stopping criterion is met

$s^i(0) = z^i(k)$, $t_i(0) = 1, \forall i \in \mathcal{V}$, the number of communication rounds per iteration can be determined using Lemma 3.1 and Proposition 3.1. In particular, the number of communication rounds at iteration k of the **GradConsensus** algorithm is given by,

$$\theta_k := \left\lceil \frac{\log(1/\varepsilon(k))}{-\log(\lambda)} + \frac{\log(nC\bar{R}/\delta)}{-\log(\lambda)} \right\rceil, \quad (29)$$

where, C, δ, λ and $\bar{R}$ are constants defined in Lemma 3.1 and Proposition 3.1 respectively. The **GradConsensus** algorithm is summarized in Algorithm 2.

Remark 3: Note that **GradConsensus** algorithm (updates (9)–(11)) utilizes a column stochastic matrix which allows for a distributed synthesis of the protocol. This feature gives an advantage over existing schemes in the literature [3], [8], [24], [36] that utilize a doubly-stochastic weight matrix and cannot achieve tractable distributed synthesis in directed networks.

IV. CONVERGENCE ANALYSIS

We analyze the convergence of the **GradConsensus** algorithm under two scenarios: (i) when the aggregate function f is convex and Lipschitz differentiable and, (ii) when f is strongly convex and Lipschitz differentiable. We provide estimates of the rate of convergence of the iterates generated by Algorithm 2 to the optimal solution(s) of problem (1) and the communication steps performed by Algorithm 2 at each iteration. Let the average of the optimization variables at iteration k be denoted as: $\hat{x}(k) := \frac{1}{n} \sum_{i=1}^n x^i(k)$. Denote the gradient of the function f evaluated at individual optimization variables of all the agents and at the average $\hat{x}(k)$ as:

$$g(k) := \sum_{i=1}^n \nabla f_i(x^i(k)), \text{ and } \hat{g}(k) := \sum_{i=1}^n \nabla f_i(\hat{x}(k)) \quad (30)$$

respectively. A consequence of the ε -consensus protocol is that the difference between $g(k)$ and $\hat{g}(k)$ is bounded at every iteration k of the **GradConsensus** algorithm.

Lemma 4.1: Under Assumptions 1 and 3, at k th iteration of Algorithm 2 with θ_k (defined in (29)) communication rounds per iteration of the consensus protocol,

$$\|g(k) - \hat{g}(k)\| \leq 2nL_h\varepsilon(k),$$

where, $L_h = \max_{1 \leq i \leq n} L_{f_i}$ and $\varepsilon(k)$ is the consensus tolerance at iteration k .

Proof: Under Assumption 3, note that,

$$\begin{aligned} \|g(k) - \hat{g}(k)\| &= \left\| \sum_{i=1}^n \nabla f_i(x^i(k)) - \sum_{i=1}^n \nabla f_i(\hat{x}(k)) \right\| \\ &\leq L_h \sum_{i=1}^n \|x^i(k) - \hat{x}(k) + \hat{x}(k) - \hat{z}(k)\| \\ &\leq L_h \sum_{i=1}^n \|x^i(k) - \hat{z}(k)\| + \|\hat{z}(k) - \hat{x}(k)\| \\ &\leq 2nL_h\varepsilon(k), \end{aligned}$$

where, in the last step follows from (24) and (25). $\square$

From update (23),

$$\begin{aligned} \frac{1}{n} \sum_{i=1}^n z^i(k) &= \frac{1}{n} \sum_{i=1}^n [x^i(k-1) - \alpha \nabla f_i(x^i(k-1))] \\ &= \hat{x}(k-1) - \frac{\alpha}{n} \sum_{i=1}^n \nabla f_i(x^i(k-1)). \end{aligned}$$

Therefore, we have,

$$\begin{aligned} \hat{z}(k) &= \hat{x}(k-1) - \hat{\alpha}g(k-1) + \hat{x}(k) - \hat{x}(k) \Rightarrow \\ \hat{x}(k) &= \hat{x}(k-1) - \hat{\alpha}g(k-1) + v(k-1), \end{aligned} \quad (31)$$

with $v(k-1) := \hat{x}(k-1) - \hat{z}(k-1)$, $\|v(k-1)\| \leq \varepsilon(k-1)$ and $\hat{\alpha} = \frac{\alpha}{n}$. In the centralized setting the information about the gradient of the function f , i.e. $\nabla f(x) = \sum_{i=1}^n \nabla f_i(x) = \hat{g}(x)$ is known to the central server. Here, an iteration of the (centralized) gradient descent is of the form: $\tilde{x} = x - \alpha \hat{g}(x)$, where, $\tilde{x}$ denotes the updated estimate of the optimal solution. Note that due to Lemma 4.1 the update (31) can be viewed as an inexact centralized gradient descent update performed at the average of all the agents' estimates for the function f . In particular,

$$\begin{aligned} \hat{x}(k+1) &= \hat{x}(k) - \hat{\alpha}g(k) + u(k), \text{ with} \\ u(k) &:= \hat{\alpha}(\hat{g}(k) - g(k)) + v(k), \|u(k)\| \leq (2L_h\alpha + 1)\varepsilon(k). \end{aligned}$$

Therefore, the **GradConsensus** algorithm updates are equivalent to (approximate) centralized gradient descent updates applied to the average optimization variable $\hat{x}(k)$ at each iteration k . We define the solution residual $\hat{e}(k)$ and objective value residual $\hat{r}(k)$ at iteration k as:

For the optimal objective function value $-\infty < f^* < \infty$ and the optimal solution set, $\mathcal{X}^*$

$$\hat{e}(k) := \hat{x}(k) - x^*, \text{ where, } x^* \in \mathcal{X}^*, \text{ and,} \quad (32)$$

$$\hat{r}(k) := f(\hat{x}(k)) - f^* = \sum_{i=1}^n f_i(\hat{x}(k)) - f^*. \quad (33)$$

Under Assumption 3 since all f_i are Lipschitz differentiable with parameter L_{f_i} , f also is Lipschitz differentiable with the constant $L_f = \sum_{i=1}^n L_{f_i}$. The following Lemma is a property

of Lipschitz differentiable convex functions that is a standard result in convex analysis.

Lemma 4.2 ([37], Theorem 2.1.5): Under Assumption 3, for all $x, y \in \mathbb{R}^p$,

$$\langle \nabla f(y) - \nabla f(x), y - x \rangle \geq \frac{1}{L_f} \|\nabla f(y) - \nabla f(x)\|^2.$$

The next result establishes that the **GradConsensus** algorithm converges to an optimal solution of the problem (1) whenever the function f is convex but not necessarily strongly convex and Lipschitz differentiable.

Theorem 4.1 (Convergence to an optimal solution): Let $\{x^i(k)\}_{k \geq 1}$ denote the sequence of optimization variables generated by Algorithm 2. Let the sequence of tolerances $\varepsilon(k)$ satisfy: $\varepsilon(k) \in (0, 1), \forall k, \sum_{s=0}^{\infty} \varepsilon^2(s) < \infty$. Under Assumptions 1–3, $x^i(k)$ for all $i \in \mathcal{V}$ converges to a solution x^∞ of (1), i.e., $f(x^\infty) = f^*$ and $\nabla f(x^\infty) = 0$.

Proof: Under Assumption 3, at any iteration k ,

$$\begin{aligned} f(\hat{x}(k+1)) &\leq f(\hat{x}(k)) + \langle \hat{g}(k), \hat{x}(k+1) - \hat{x}(k) \rangle \\ &\quad + \frac{L_f}{2} \|\hat{x}(k+1) - \hat{x}(k)\|^2 \end{aligned}$$

Subtracting, f^* from both sides, and using (31),

$$\begin{aligned} \hat{r}(k+1) &\leq \hat{r}(k) + \langle \hat{g}(k), \hat{x}(k+1) - \hat{x}(k) \rangle \\ &\quad + \frac{L_f}{2} \|\hat{x}(k+1) - \hat{x}(k)\|^2 \\ &= \hat{r}(k) - \hat{\alpha} \langle \hat{g}(k), g(k) \rangle + \frac{\hat{\alpha}^2 L_f}{2} \|g(k)\|^2 \\ &\quad - \hat{\alpha} L_f \langle g(k), v(k) \rangle + \langle \hat{g}(k), v(k) \rangle + \frac{L_f}{2} \|v(k)\|^2 \\ &= \hat{r}(k) - \hat{\alpha} \langle \hat{g}(k), g(k) + \hat{g}(k) - \hat{g}(k) \rangle \\ &\quad + \frac{\hat{\alpha}^2 L_f}{2} \|g(k) + \hat{g}(k) - \hat{g}(k)\|^2 + \frac{L_f}{2} \|v(k)\|^2 \\ &\quad - \hat{\alpha} L_f \langle g(k) + \hat{g}(k) - \hat{g}(k), v(k) \rangle + \langle \hat{g}(k), v(k) \rangle \\ &= \hat{r}(k) - \hat{\alpha} \langle \hat{g}(k), \hat{g}(k) \rangle - \hat{\alpha} \langle \hat{g}(k), g(k) - \hat{g}(k) \rangle \\ &\quad + \frac{\hat{\alpha}^2 L_f}{2} \|g(k) - \hat{g}(k)\|^2 + \frac{\hat{\alpha}^2 L_f}{2} \|\hat{g}(k)\|^2 \\ &\quad + \hat{\alpha}^2 L_f \langle g(k) - \hat{g}(k), \hat{g}(k) \rangle \\ &\quad + (1 - \hat{\alpha} L_f) \langle \hat{g}(k), v(k) \rangle \\ &\quad + \hat{\alpha} L_f \langle \hat{g}(k) - g(k), v(k) \rangle + \frac{L_f}{2} \|v(k)\|^2 \\ &= \hat{r}(k) - \hat{\alpha} \langle \hat{g}(k), \hat{g}(k) \rangle + \frac{\hat{\alpha}^2 L_f}{2} \|g(k) - \hat{g}(k)\|^2 \\ &\quad + \frac{L_f}{2} \|v(k)\|^2 + \frac{\hat{\alpha}^2 L_f}{2} \|\hat{g}(k)\|^2 \\ &\quad + (\hat{\alpha} - \hat{\alpha}^2 L_f) \langle \hat{g}(k), \hat{g}(k) - g(k) \rangle \\ &\quad + (1 - \hat{\alpha} L_f) \langle \hat{g}(k), v(k) \rangle + \hat{\alpha} L_f \langle \hat{g}(k) \\ &\quad - g(k), v(k) \rangle. \end{aligned}$$

Using the $\pm 2\langle a, b \rangle \leq \|a\|^2 + \|b\|^2$, $a, b \in \mathbb{R}^p$ inequality on the term $\langle \hat{g}(k), \hat{g}(k) - g(k) \rangle$ we get,

$$\begin{aligned} \hat{r}(k+1) &\leq \hat{r}(k) + \frac{L_f}{2} \|v(k)\|^2 + \frac{\hat{\alpha}^2 L_f}{2} \|g(k) - \hat{g}(k)\|^2 \\ &\quad + (1 - \hat{\alpha} L_f) \langle \hat{g}(k), v(k) \rangle \end{aligned}$$

$$\begin{aligned} &+ \hat{\alpha} L_f \langle \hat{g}(k) - g(k), v(k) \rangle \\ &+ \frac{(\hat{\alpha} - \hat{\alpha}^2 L_f)}{2} \|\hat{g}(k)\|^2 + \frac{(\hat{\alpha} - \hat{\alpha}^2 L_f)}{2} \|\hat{g}(k) - g(k)\|^2 \\ &+ \frac{\hat{\alpha}^2 L_f}{2} \|\hat{g}(k)\|^2 - \hat{\alpha} \|\hat{g}(k)\|^2 \\ &= \hat{r}(k) + \frac{L_f}{2} \|v(k)\|^2 + \frac{\hat{\alpha}^2 L_f}{2} \|g(k) - \hat{g}(k)\|^2 \\ &\quad + (1 - \hat{\alpha} L_f) \langle \hat{g}(k), v(k) \rangle + \hat{\alpha} L_f \langle \hat{g}(k) \\ &\quad - g(k), v(k) \rangle - \frac{\hat{\alpha}}{2} \|\hat{g}(k)\|^2 + \frac{(\hat{\alpha} - \hat{\alpha}^2 L_f)}{2} \\ &\quad \times \|\hat{g}(k) - g(k)\|^2 \end{aligned}$$

Application of the $\pm 2\langle a, b \rangle \leq \zeta \|a\|^2 + \frac{1}{\zeta} \|b\|^2$, $a, b \in \mathbb{R}^p$ with $\zeta = \frac{2}{\hat{\alpha}}$ on the term $\langle \hat{g}(k), v(k) \rangle$ leads to

$$\begin{aligned} \hat{r}(k+1) &\leq \hat{r}(k) + \frac{L_f}{2} \|v(k)\|^2 + \frac{\hat{\alpha}^2 L_f}{2} \|g(k) - \hat{g}(k)\|^2 \\ &\quad + \frac{(\hat{\alpha} - \hat{\alpha}^2 L_f)}{2} \|\hat{g}(k) - g(k)\|^2 + \hat{\alpha} L_f \langle \hat{g}(k) \\ &\quad - g(k), v(k) \rangle - \frac{\hat{\alpha}}{2} \|\hat{g}(k)\|^2 + (1 - \hat{\alpha} L_f) \left(\frac{\hat{\alpha}}{4} \right) \\ &\quad \times \|\hat{g}(k)\|^2 + (1 - \hat{\alpha} L_f) \frac{1}{\hat{\alpha}} \|v(k)\|^2 \\ &\leq \hat{r}(k) + \frac{L_f}{2} \|v(k)\|^2 + \frac{\hat{\alpha}^2 L_f}{2} \|g(k) - \hat{g}(k)\|^2 \\ &\quad + \frac{(1 - \hat{\alpha} L_f)}{\hat{\alpha}} \|v(k)\|^2 + \frac{(\hat{\alpha} - \hat{\alpha}^2 L_f)}{2} \\ &\quad \times \|\hat{g}(k) - g(k)\|^2 + \hat{\alpha} L_f \langle \hat{g}(k) - g(k), v(k) \rangle \\ &\quad - \frac{\hat{\alpha}}{4} \|\hat{g}(k)\|^2 \end{aligned}$$

Further, using Cauchy-Schwarz inequality and dropping the negative terms,

$$\begin{aligned} \hat{r}(k+1) &\leq \hat{r}(k) - \frac{\hat{\alpha}}{4} \|\hat{g}(k)\|^2 + \frac{1}{\hat{\alpha}} \|v(k)\|^2 \\ &\quad + \frac{\hat{\alpha}}{2} \|g(k) - \hat{g}(k)\|^2 + \hat{\alpha} L_f \|\hat{g}(k) \\ &\quad - g(k)\| \|v(k)\| \end{aligned}$$

Using, Lemma 4.1, we get,

$$\hat{r}(k+1) \leq \hat{r}(k) - \frac{\hat{\alpha}}{4} \|\hat{g}(k)\|^2 + a_0 \varepsilon^2(k) \quad (34)$$

$$\leq \hat{r}(k) + a_0 \varepsilon^2(k). \quad (35)$$

where, $a_0 := (\frac{1}{\hat{\alpha}} + 2nL_h\hat{\alpha}(nL_h + L_f))$. Applying (35) recursively, we get,

$$\hat{r}(k) \leq \hat{r}(0) + a_0 \sum_{s=0}^{\infty} \varepsilon^2(s) < \infty. \quad (36)$$

Therefore, there exists sub-sequence $\hat{r}(k_l)$ such that $\lim_{l \rightarrow \infty} \hat{r}(k_l) < \infty$. From (34) it follows, $\lim_{l \rightarrow \infty} \hat{g}(k_l) = \lim_{l \rightarrow \infty} \nabla f(\hat{x}(k_l)) = 0$. From (31) and Lemma 4.1, $\|\hat{x}(k+1) - \hat{x}(k)\| \leq \hat{\alpha} \|\hat{g}(k)\| + (2nL_h\hat{\alpha} + 1)\varepsilon(k)$. Thus, taking the limit over appropriate sequences on both sides gives $\|\hat{x}(k_l+1) - \hat{x}(k_l)\| \rightarrow 0$. Thus, $\hat{x}(k_l) \rightarrow \hat{x}^\infty$. Note that for all $i \in \mathcal{V}$, $\|x^i(k) - \hat{x}(k)\| \leq \varepsilon(k)$. Therefore, there exists x^∞ such that $\lim_{l \rightarrow \infty} x^i(k_l) = x^\infty$ for all $i = 1, \dots, n$. Now, for any $i \in \mathcal{V}$, consider $\|\nabla f(\hat{x}(k)) -$

$\|\nabla f(x^i(k))\| \leq L_f \|\hat{x}(k) - x^i(k)\|$. Taking limits over appropriate sequences, $\|\nabla f(\hat{x}(k_l)) - \nabla f(x^i(k_l))\| \rightarrow 0$. Thus, $\nabla f(x^\infty) = \lim_{l \rightarrow \infty} \nabla f(x^i(k_l)) = 0$. Further, since, f is convex,

$$f(x^*) \geq f(x^i(k)) + \langle \nabla f(x^i(k)), x^* - x^i(k) \rangle.$$

Note that under the assumptions here $x^i(k)$ is bounded. Taking limit over appropriate sequences on both sides gives, $f^* = f(x^*) \geq f(x^\infty)$. Thus, $f(x^\infty) = f^*$. To complete the proof, we show that $x^i(k)$ has a unique limit point. Let x_1^∞ and x_2^∞ be any two limit points of $x^i(k)$. From (36) and the fact that f is convex we have, $\|x^i(k) - x_j^\infty\|$ is bounded for all $i \in \mathcal{V}, j = 1, 2$. Thus, $\lim_{k \rightarrow \infty} \|x^i(k) - x_j^\infty\|^2 = \psi_j < \infty$ exists for all $i \in \mathcal{V}, j = 1, 2$. Now consider,

$$\begin{aligned} & \|x^i(k) - x_1^\infty\|^2 - \|x^i(k) - x_2^\infty\|^2 \\ &= \|x_1^\infty\|^2 - \|x_2^\infty\|^2 - 2(x_1^\infty - x_2^\infty)^\top x^i(k) \end{aligned}$$

Taking limit both sides for each limit point we get,

$$\begin{aligned} \psi_1 - \psi_2 &= \|x_1^\infty\|^2 - \|x_2^\infty\|^2 - 2(x_1^\infty - x_2^\infty)^\top x_1^\infty, \\ &= -\|x_1^\infty - x_2^\infty\|^2 \text{ and,} \\ \psi_1 - \psi_2 &= \|x_1^\infty\|^2 - \|x_2^\infty\|^2 - 2(x_1^\infty - x_2^\infty)^\top x_2^\infty, \\ &= \|x_1^\infty - x_2^\infty\|^2. \end{aligned}$$

Therefore, $\|x_1^\infty - x_2^\infty\| = 0$. Hence, x^∞ is unique. $\square$

We now provide two estimates for the convergence rate for the **GradConsensus** algorithm.

A. Q-Linear Rate of Convergence

Here, the convergence rate estimate of the **GradConsensus** algorithm is derived when the function f is convex but not necessarily strongly convex and Lipschitz differentiable. Under these assumptions, Theorem 4.2 establishes a Q-linear rate of convergence to a small neighborhood of an optimal solution of (1).

Theorem 4.2 (Q-linear rate of convergence): Let Assumptions 1–3 hold. Let $\hat{\alpha} \leq \frac{2}{L_f}$, $\varepsilon(k) = \frac{\varepsilon_0}{k^{1+\eta}}$, where, ε_0, η are positive constants. Denote, the Riemann zeta function by $\zeta(\cdot)$ and let $\mathbf{e} := \|\hat{x}(0) - x^*\| + (2\alpha L_h + 1)\varepsilon_0\zeta(1+\eta)$. Then, while $\hat{r}(k) > 2\mathbf{e}\varepsilon_0\sqrt{\frac{2}{\hat{\alpha}}(\frac{1}{\hat{\alpha}} + 2nL_h\hat{\alpha}(nL_h + L_f))}$, $\hat{r}(k)$ decreases at a Q-linear rate. In particular, there exists $\beta \in (0, 1)$, such that for all $k \geq 0$, $\hat{r}(k+1) \leq \beta\hat{r}(k)$.

Proof: We show that the solution residual $\|\hat{e}(k)\|$ is bounded. Consider,

$$\begin{aligned} \|\hat{x}(k) - \hat{\alpha}\hat{g}(k) - x^*\|^2 &= \|\hat{x}(k) - x^*\|^2 + \hat{\alpha}^2\|\hat{g}(k)\|^2 \\ &\quad - 2\hat{\alpha}\langle \hat{g}(k), \hat{x}(k) - x^* \rangle \\ &\leq \|\hat{x}(k) - x^*\|^2 + \hat{\alpha}(\hat{\alpha} - \frac{2}{L_f})\|\hat{g}(k)\|^2. \end{aligned}$$

Therefore,

$$\|\hat{x}(k) - \hat{\alpha}\hat{g}(k) - x^*\| \leq \|\hat{x}(k) - x^*\|, \quad (37)$$

where, we used Lemma 4.2 and the facts $\hat{g}(x^*) = 0$, $\hat{\alpha} \leq 2/L_f$. Using (31), Lemma 4.1 and (37),

$$\begin{aligned} \|\hat{x}(k+1) - x^*\| &= \|\hat{x}(k) - \hat{\alpha}\hat{g}(k) - x^* + v(k)\| \\ &= \|\hat{x}(k) - \hat{\alpha}\hat{g}(k) + \hat{\alpha}\hat{g}(k) \\ &\quad - \hat{\alpha}\hat{g}(k) - x^* + v(k)\| \\ &\leq \|\hat{x}(k) - \hat{\alpha}\hat{g}(k) - x^*\| + \hat{\alpha}\|g(k) \\ &\quad - \hat{g}(k)\| + \varepsilon(k) \\ &\leq \|\hat{x}(k) - x^*\| + \frac{(2\alpha L_h + 1)\varepsilon_0}{k^{1+\eta}} \\ &\leq \|\hat{x}(0) - x^*\| + (2\alpha L_h + 1)\varepsilon_0 \sum_{s=0}^k \frac{1}{k^{1+\eta}} \\ &\leq [\|\hat{x}(0) - x^*\| + (2\alpha L_h + 1)\varepsilon_0\zeta(1+\eta)] =: \mathbf{e}, \end{aligned} \quad (38)$$

where, $\zeta(\cdot)$ is the Riemann zeta function. Note that, $\hat{r}(k) = f(\hat{x}(k)) - f^* \leq \langle \hat{g}(k), \hat{x}(k) - x^* \rangle = \langle \hat{g}(k), \hat{e}(k) \rangle$. Using (38) we obtain,

$$\|\hat{g}(k)\| \geq \|\hat{g}(k)\| \frac{\|\hat{x}(k) - x^*\|}{\mathbf{e}} \geq \frac{\langle \hat{g}(k), \hat{x}(k) - x^* \rangle}{\mathbf{e}} \geq \frac{\hat{r}(k)}{\mathbf{e}}.$$

Thus, from (34), we have,

$$\hat{r}(k+1) \leq \hat{r}(k) - \frac{\hat{\alpha}}{4\mathbf{e}^2}\hat{r}^2(k) + a_0\varepsilon^2(k).$$

Therefore, while $\hat{r}(k) > 2\mathbf{e}\sqrt{\frac{2a_0}{\hat{\alpha}}}\varepsilon_0$, we have,

$$\begin{aligned} \hat{r}(k+1) &\leq \hat{r}(k) - \frac{\hat{\alpha}}{8\mathbf{e}^2}\hat{r}^2(k) \\ &\leq \hat{r}(k) - \frac{\hat{\alpha}}{8\mathbf{e}^2}\hat{r}(k)2\mathbf{e}\varepsilon_0L_h \\ &= (1 - \frac{\alpha}{4\mathbf{e}}L_h\varepsilon_0)\hat{r}(k) = \beta\hat{r}(k). \end{aligned}$$

Note, $\alpha L_h\varepsilon_0 < 4\mathbf{e} \Rightarrow \beta \in (0, 1)$. $\square$

Remark 4: From Theorem 4.1, there exists $\bar{B} < \infty$ such that $\|x^i(k)\| \leq \bar{B}$ for all $i \in \mathcal{V}$ and $k \geq 0$. Therefore, under Assumption 3, $\|\nabla f_i(x^i(k))\| \leq (L_h\bar{B} + \max_{1 \leq i \leq n} \|\nabla f_i(x^i(0))\|) =: h_m < \infty$ for all $i \in \mathcal{V}$. Thus,

$$\begin{aligned} f(x^i(k)) - f^* &\leq \hat{r}(k) + \langle \nabla f(x^i(k)), x^i(k) - \hat{x}(k) \rangle \\ &\leq \hat{r}(k) + \sum_{j=1}^n \|\nabla f(x^i(k))\| \|x^i(k) - \hat{x}(k)\| \\ &\leq \hat{r}(k) + h_m n \varepsilon(k) = \hat{r}(k) + h_m n \frac{\varepsilon_0}{k^{1+\eta}}. \end{aligned}$$

Hence, using the result of Theorem 4.2 we conclude that $f(x^i(k)) - f^*$, similar to $\hat{r}(k)$ decreases geometrically, with respect to the outer gradient descent iterations of the Algorithm 2, until reaching a small $O(\varepsilon_0)$ neighborhood. The parameter ε_0 is a user-specified algorithm parameter that can be chosen appropriately to get solutions close to the optimal solution.

Remark 5: For the result in Theorem 4.2, the communication rounds per iteration k for Algorithm 2 are given by $\left\lceil \frac{\log(k^{1+\eta}/\varepsilon_0)}{-\log(\lambda)} + \frac{\log(nC\bar{R}/\delta)}{-\log(\lambda)} \right\rceil = O\left(\log\left(k/(\varepsilon_0)^{\frac{1}{\eta+1}}\right)\right)$, where C, δ, λ and $\bar{R}$ are constants defined in Lemma 3.1 and Proposition 3.1 respectively. The Algorithm 2 with $\varepsilon_0 = 1$ (in the worst case) utilizes $O(k \log k)$ number of communication steps

along with k gradient descent iterations. This communication complexity is within a factor of $\log k$ of the optimal lower bound $O(k)$. The additional communication steps provide improved convergence guarantees for the **GradConsensus** algorithm over the existing algorithms in the literature. In particular, the Q-linear rate of convergence in Theorem 4.2 is stronger than the sub-linear convergence rate present in the literature [3], [5], [8] under Assumption 3. Moreover, empirically it is seen that the total number of gradient descent (computation) steps is smaller with our algorithm. Thus it is well suited to situations where computations are expensive.

B. R-Linear Rate of Convergence

In this subsection, we make an additional assumption:

Assumption 4: $f = \sum_{i=1}^n f_i$ is strongly convex with parameter $\sigma_f > 0$.

Note that Assumption 4 does not require all f_i 's to be necessarily strongly convex. However, when this is the case, Assumption 4 holds naturally. We show in Theorem 4.3 that under Assumption 4 the solution estimates of every agent converge exactly to the optimal solution of problem (1) at an R-linear rate. We utilize the following standard result in the proof of Theorem 4.3.

Lemma 4.3 ([37], Theorem 2.1.12): Under assumptions 3 and 4 for all $x, y \in \mathbb{R}^p$,

$$\begin{aligned} \langle x - y, \nabla f(x) - \nabla f(y) \rangle &\geq \frac{\sigma_f L_f}{\sigma_f + L_f} \|x - y\|^2 \\ &\quad + \frac{1}{\sigma_f + L_f} \|\nabla f(x) - \nabla f(y)\|^2. \end{aligned}$$

Theorem 4.3: Let Assumptions 1–4 hold. Let $\hat{\alpha} \leq \frac{2}{(\sigma_f + L_f)}$. Define, $\rho := \sqrt{1 - 2\hat{\alpha}\sigma_f L_f / (\sigma_f + L_f)}$. Let $\varepsilon(k) \leq \mu^k$, where, $\mu \in [\rho, 1)$. Then the agent estimates converge at an R-linear rate to the optimal solution,

$$\|x^i(k) - x^*\| \leq \Gamma \mu^k, \text{ for all } i \in \mathcal{V},$$

where, $\Gamma := \|\hat{x}(0) - x^*\| + (2\alpha L_h + 1) \frac{\mu}{\mu - \rho} + 1$.

Proof: Consider,

$$\begin{aligned} \|\hat{x}(k) - \hat{\alpha}\hat{g}(k) - x^*\|^2 &= \|\hat{x}(k) - x^*\|^2 + \hat{\alpha}^2 \|\hat{g}(k)\|^2 \\ &\quad - 2\hat{\alpha} \langle \hat{g}(k), \hat{x}(k) - x^* \rangle \\ &\leq \left(1 - \frac{2\hat{\alpha}\sigma_f L_f}{\sigma_f + L_f}\right) \|\hat{x}(k) - x^*\|^2 \\ &\quad + \hat{\alpha} \left(\hat{\alpha} - \frac{2}{\sigma_f + L_f}\right) \|\hat{g}(k)\|^2 \\ &\leq \rho^2 \|\hat{x}(k) - x^*\|^2, \end{aligned} \quad (39)$$

where, we used Lemma 4.3 and the facts $\hat{g}(x^*) = 0, \hat{\alpha} \leq \frac{2}{(\sigma_f + L_f)}$. From (31),

$$\begin{aligned} \|\hat{x}(k+1) - x^*\| &= \|\hat{x}(k) - \hat{\alpha}g(k) - x^* + v(k)\| \\ &\leq \|\hat{x}(k) - \hat{\alpha}\hat{g}(k) - x^*\| + \hat{\alpha} \|\hat{g}(k) - g(k)\| + \varepsilon(k). \end{aligned} \quad (40)$$

Combining (39), (40) and the result of Lemma 4.1 we get,

$$\begin{aligned} \|\hat{x}(k+1) - x^*\| &\leq \rho \|\hat{x}(k) - x^*\| + (2\alpha L_h + 1) \varepsilon(k) \\ &= \rho \|\hat{x}(k) - x^*\| + (2\alpha L_h + 1) \mu^k. \end{aligned}$$

Applying the above inequality recursively we get,

$$\|\hat{x}(k) - x^*\| \leq \rho^k \|\hat{x}(0) - x^*\| + (2\alpha L_h + 1) \sum_{s=0}^{k-1} \rho^s \mu^{k-s}.$$

Using the triangle inequality and the above relation we get,

$$\begin{aligned} \|x^i(k) - x^*\| &\leq \|\hat{x}(k) - x^*\| + \|x^i(k) - \hat{x}(k)\| \\ &\leq \rho^k \|\hat{x}(0) - x^*\| + (2\alpha L_h + 1) \sum_{s=0}^{k-1} \rho^s \mu^{k-s} + \mu^k \\ &\leq \rho^k \|\hat{x}(0) - x^*\| + (2\alpha L_h + 1) \mu^k \sum_{s=0}^{k-1} \left(\frac{\rho}{\mu}\right)^s + \mu^k \\ &\leq \rho^k \|\hat{x}(0) - x^*\| + (2\alpha L_h + 1) \mu^k \frac{\mu}{\mu - \rho} + \mu^k \\ &\leq \left[\|\hat{x}(0) - x^*\| + (2\alpha L_h + 1) \frac{\mu}{\mu - \rho} + 1 \right] \mu^k = \Gamma \mu^k. \end{aligned}$$

□

Remark 6: For the result in Theorem 4.3, the communication rounds per iteration k for Algorithm 2 are given by $\left\lceil \frac{k \log(\mu)}{-\log(\lambda)} + \frac{\log(nC\bar{R}/\delta)}{-\log(\lambda)} \right\rceil = O(k)$, where C, δ, λ and $\bar{R}$ are constants defined in Lemma 3.1 and Proposition 3.1 respectively. Algorithm 2 (in the worst case) utilizes $O(k^2)$ number of communication steps along with k gradient descent iterations for a global R-linear rate of convergence to the exact optimal solution. The per node work complexity (sum of total number of gradient computations and total number of communication steps) to achieve an accuracy of $\tau > 0$, i.e., $\|x^i(k) - x^*\| \leq \tau$ is given by: $O(\log(1/\tau))$ computation steps + $O(\log(1/\tau)^2)$ communication steps.

From Theorems 4.2 and 4.3, the consensus parameter $\varepsilon(k)$ can be chosen suitably to get near-optimal solutions. Theorem 4.3 shows that by controlling the disagreement between the agents (and hence the infeasibility of the algorithm iterates) to a small value using an appropriately chosen $\varepsilon(k)$, results in strong convergence guarantees (global R-linear rate of convergence) for the **GradConsensus** algorithm. However, there exists a trade-off as tight regulation of the disagreement between agents' estimates would require performing more total number of communication steps ($O(k^2)$ in the worst case) until any outer gradient descent iteration k , compared to $O(k \log k)$ (in the worst case) that gives a Q-linear rate (Theorem 4.2).

V. NUMERICAL SIMULATIONS AND RESULTS

We consider a network of 1000 agents with the interconnection topology generated using the Erdos-Rényi model [38] with connectivity probability 0.005. The interconnection topology is shown in Fig. 1. The diameter of the 1000 node graph is 10. In the simulation study, we choose the parameter $\mathcal{D}$ in the **GradConsensus** algorithm to be 11.

The weight matrix $\mathbf{P}$ is chosen using the equal neighbor model [31]. To demonstrate the performance, we solve the distributed logistic regression problem,

$$\underset{x \in \mathbb{R}^p}{\text{minimize}} \sum_{i=1}^{1000} \frac{1}{n_i} \sum_{j=1}^{n_i} \log(1 + \exp(-(A_{ij}x)y_{ij})), \quad (41)$$



Fig. 1. 1000-node test network topology.

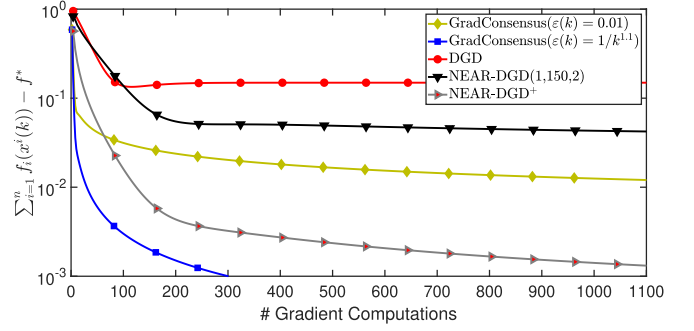
where, each agent $i \in \mathcal{V}$ has its training data $(A_{ij}, y_{ij}) \in \mathbb{R}^p \times \{-1, +1\}$, $j = 1, 2, \dots, n_i$, with feature variables A_{ij} and binary outcomes y_{ij} . Here, $(p = 35, n_i = 1500)$. For our simulations, we generate an artificial data set of feature vectors A_{ij} with outcome $y_{ij} = 1$ from a standard normal distribution, and with output $y_{ij} = -1$ from another normal distribution with mean 0 and standard deviation 0.707. We compare the performance of the proposed **GradConsensus** algorithm with the well-known Distributed Gradient Descent (DGD) [3] method, and the NEAR-DGD methods presented in [27] that utilize multiple communication steps at each iteration. Two versions of the **GradConsensus**, with $\varepsilon(k) = 0.01$, and $\varepsilon(k) = 1/k^{1.1}$, $\forall k$ are used for comparison. Two variants of the NEAR-DGD algorithm are implemented: (i) NEAR-DGD(1,150,2) where we start with 1 round of communication and the number of communication rounds are doubled after every 150 algorithm iterations, (ii) NEAR-DGD⁺ where the number of communication steps is increased linearly with respect to algorithm iteration (at algorithm iteration k the number of communication steps performed is k).

Let $x^1(k), x^2(k), \dots, x^n(k)$ denote the algorithm estimates produced by the corresponding algorithm and f^* denote the optimal value of problem (41) solved via a centralized solver. We provide a performance comparison between the **GradConsensus** and the baseline methods (DGD and NEAR-DGD) by plotting the objective value residual $\sum_{i=1}^n f_i(x^i(k)) - f^*$ with respect to (i) the number gradient computations performed by the methods, (ii) total communication steps required by the methods, and (iii) the CPU run-time taken by the methods. We also provide a practical estimate of communication complexity by plotting the number of communication steps required at each algorithm iteration (denoted as the counter k in the above sections) for all the methods.

All the numerical examples in this section are implemented in MATLAB and run on a desktop computer with 16 GB RAM and an Intel Core i7 processor running at 1.90 GHz. The parameters

TABLE I
ALGORITHM PARAMETERS

Algorithm	Parameters
GradConsensus	$\varepsilon(k) = 0.01, 1/k^{1.1}$, step-size $\alpha = 0.02$
DGD [3]	step-size $\alpha = 0.02$
NEAR-DGD(1,150,2) [27]	step-size $\alpha = 0.02$
NEAR-DGD ⁺ [27]	step-size $\alpha = 0.02$

Fig. 2. Objective value residual $\sum_{i=1}^n f_i(x^i(k)) - f^*$ against the number of gradient computations.

used in the simulations for all the algorithms are reported in Table I.

In choosing the step-sizes of the algorithms in Table I we followed the approach of hand-optimizing the hyper-parameters that produce a good performance for these algorithms while maintaining the stability of the algorithmic estimates for the logistic regression problem. We choose the same step-size of 0.02 for all the compared algorithms.

A. Comparison With Respect to Gradient Computations

Fig. 2 provides a comparison with respect to the number of gradient computations. The **GradConsensus** algorithm with $\varepsilon(k) = 1/k^{1.1}$ gives the best performance among the compared algorithms converging to an objective value residual less than 10^{-3} after around 300 gradient computations. The NEAR-DGD⁺ method has the second-best performance, requiring more than 1100 iterations to achieve an objective value residual of 10^{-3} . Observe that **GradConsensus** with $\varepsilon(k) = 0.01$ performs better than the NEAR-DGD(1,150,2) method with objective value residual closer to 10^{-2} after 1100 gradient computations. The total number of gradient computation steps required by the proposed **GradConsensus** algorithm with $\varepsilon(k) = 1/k^{1.1}$ is lesser compared to the other algorithms. This shows an advantage of using the **GradConsensus** algorithm in applications where the computation of the gradients is cost-intensive.

B. Communication Complexity During Algorithm Iterations

A comparison between the communication complexity of the variants of the proposed **GradConsensus** algorithm and the baselines (DGD and NEAR-DGD methods) is presented in Fig. 3. The theoretical worst-case bounds on the communication steps for the **GradConsensus** algorithm is given in (29). The

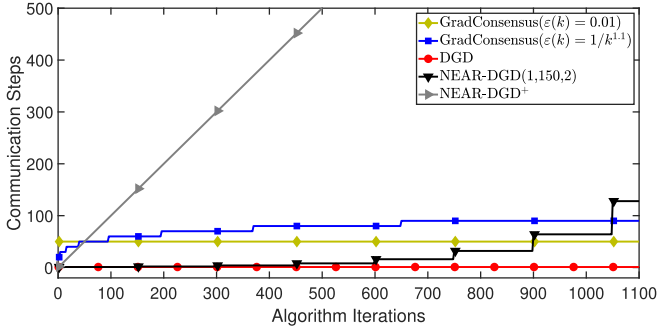


Fig. 3. Communication steps required per iteration of the algorithm.

communication steps required at each iteration by the algorithms while solving the distributed logistic regression problem (41) are presented in Fig. 3.

The DGD method has the minimum number of communication steps required at each iteration with only one round of communication per algorithm iteration. The NEAR-DGD⁺ algorithm requires the number of communication steps to linearly increase with the number of iterations which leads to the worst total communication burden among the compared methods. The two variants of the **GradConsensus** algorithm and the NEAR-DGD(1,150,2) method incur a communication load that lies between the communication load of DGD and NEAR-DGD⁺ algorithms. The **GradConsensus** algorithm with $\varepsilon(k) = 0.01$ takes almost a constant number (50 to 52) of communication steps at each iteration throughout the simulation run. The **GradConsensus** algorithm with $\varepsilon(k) = 1/k^{1.1}$ has a near logarithmic increase in the number of communication steps taken at each iteration. The NEAR-DGD(1,150,2) method starts with one communication step load but has a geometric increase after every 150 algorithm iteration. The number of communication steps required by the NEAR-DGD(1,150,2) method per algorithm iteration becomes more than the two variants of the **GradConsensus** algorithm after 1050 algorithm iterations, implying worse communication complexity per iteration (than the **GradConsensus** algorithm variants) as the algorithm progresses.

C. Comparison With Respect to Communication Steps

Fig. 4 gives a comparison of the objective value residuals with respect to the total communication steps taken by the algorithms. One can see that the **GradConsensus** algorithm starts slow due to the communication steps required by the ε -consensus protocol to maintain a feasible solution to the consensus constraint. Thus, for smaller total communication steps (less than 100) the **GradConsensus** algorithm doesn't have the best performance. In this region (total communication steps less than 100), the NEAR-DGD and DGD methods perform well. However, maintaining consensus constraint feasibility in the **GradConsensus** algorithm pays off as the algorithm progresses. For a higher total number of communication steps, the **GradConsensus** with $\varepsilon(k) = 1/k^{1.1}$ has the best performance with the objective value residual value less than 10^{-2} after 900 communication steps. The NEAR-DGD⁺ method has a

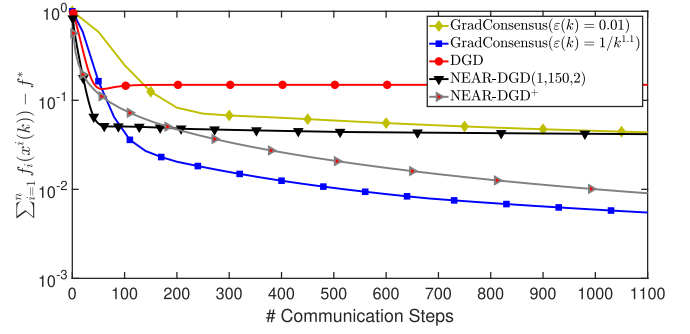


Fig. 4. Objective value residual $\sum_{i=1}^n f_i(x^i(k)) - f^*$ against total communication steps.

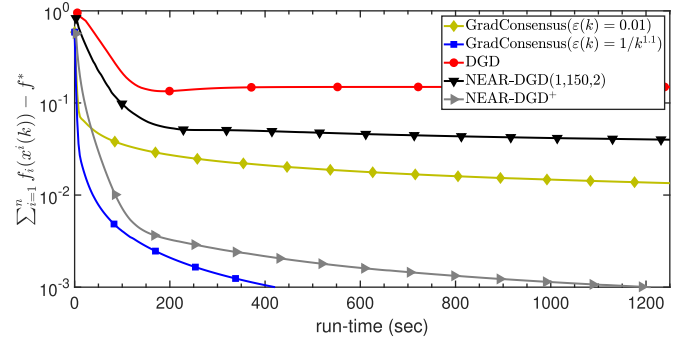


Fig. 5. Comparison of the objective value residual $\sum_{i=1}^n f_i(x^i(k)) - f^*$ with respect to total run-time.

higher communication requirement (a linear increase in the communication steps compared to the logarithmic increase for the **GradConsensus** algorithm with $\varepsilon(k) = 1/k^{1.1}$) which deteriorates its eventual performance. NEAR-DGD(1,150,2) method on the other hand has a lesser communication requirement which leads to its initial good performance. However, the objective value residual plateaus and the **GradConsensus** algorithm with $\varepsilon(k) = 0.01$ (with consensus constraint feasibility) having started behind the NEAR-DGD(1,150,2) catches up with the objective value residual less than 10^{-1} around 1000 total communication steps. Thus, utilizing the ε -consensus protocol that uses an appropriate number of communication steps (while maintaining consensus constraint feasibility) based on the network structure rather than an ad-hoc increase in the communication steps leads to better performance of the **GradConsensus** algorithm.

D. Comparison With Respect to run Time

Next, we present a performance comparison between the proposed **GradConsensus** algorithm and other methods regarding the total run time required. For this simulation, during each iteration of the algorithms, a communication delay is introduced uniformly at random taking values between 0 and 1 ms (maximum delay of 1 ms). Fig. 5 presents the objective value residual with respect to the total run-time (in seconds). The ε -consensus protocol incurs additional communication steps, however, the information mixing steps (9)–(10) and (15)–(16)

have a low computational footprint and do not significantly increase the overall run time of the **GradConsensus** algorithm. The objective value residual plots in Fig. 5 demonstrate this. The **GradConsensus** algorithm with $\varepsilon(k) = 1/k^{1.1}$ has the best decrease in the objective value residual with a value of 10^{-3} . The **GradConsensus** with $\varepsilon(k) = 0.01$ also has good performance.

E. Discussion on Scalability of the **GradConsensus**

We remark that the proposed **GradConsensus** algorithm is distributedly synthesized and implemented. Each agent i only needs to know its neighborhood information. This is pivotal since the algorithm doesn't need to be set up centrally, and thus when the number of nodes in the network increases agents only need to adjust their local weights in the algorithm. Further, the information mixing over networks depends on the diameter of the directed graph which is a measure of connectivity in a graph. **GradConsensus** utilizes an upper bound $\mathcal{D}$ on the diameter. We emphasize that in many modern applications $\mathcal{D} \ll n$, the diameter of the graph is of a much smaller order than the number of nodes n in the graph [39]. Even though a thorough mathematical investigation of the relationship between a graph's size and its diameter is well outside the scope of our article, as a heuristic, let us mention for example that in an Erdos-Rényi graph $G(n, p)$ (with n being the number of nodes and p being the probability of a link between two nodes), where p is an order larger than $\frac{\log n}{n}$, then the graph is connected almost surely, and its diameter is of the order $\log(n)/\log(np)$ (see [40]). In short, in large graphs, when connections between the nodes are random, the diameter of the graph is of a much smaller order than the size of the graph, even under sparse connectivity. Thus, the **GradConsensus** algorithm: (i) requiring only local neighborhood information by the agents for setup and implementation, and (ii) dependence on the graph diameter (or an upper bound $\mathcal{D}$) rather than the number of nodes, scales well with the increase in the number of nodes.

Summarizing, the simulation results indicate the **GradConsensus** algorithm has a better performance compared to the baselines with respect to the number of gradient computations and the run time and remains acceptable with respect to total communication steps. Further, **GradConsensus** is scalable due to distributed synthesis and local computation and communication.

VI. CONCLUSION

In this article, we consider the problem of distributedly minimizing the sum of n convex functions over a directed multi-agent network. We introduce the *optimize then agree* framework where the optimization step and the consensus step are decoupled to improve the convergence properties of the distributed optimization algorithms by reducing the mismatch between the solution estimates of the agents. We developed a novel **GradConsensus** algorithm where each agent performs a gradient-descent update for the optimization step and utilizes a finite-time ε -consensus protocol to achieve $\varepsilon(k)$ -close agreement between the agent estimates at each iteration k of the

algorithm. We established explicit convergence rate estimates for the **GradConsensus** algorithm under two different sets of assumptions on the aggregate objective function f . In particular, we showed that the iterates generated by the **GradConsensus** algorithm converges to the (neighborhood of the optimal solution) optimal solution at an $R(Q)$ -linear rate of convergence under these assumptions. In numerical simulations, we applied the **GradConsensus** to solve the distributed logistic regression problem. The results indicate the suitability of the proposed **GradConsensus** algorithm in solving distributed optimization problems. The presented *optimize then agree* framework can be applied to existing algorithms in the literature. It provides guidelines for the development of newer algorithms to solve the distributed optimization problem. Extension of the *optimize then agree* framework and the proposed **GradConsensus** algorithm to time-varying directed topologies with communication delays is a future direction of research of the current work.

REFERENCES

- [1] J. N. Tsitsiklis, "Problems in decentralized decision making and computation," Cambridge Lab for Inf. and Decis. Syst., Massachusetts Inst. of Technol., Cambridge, MA, USA, Tech. Rep. ADA150025, 1984.
- [2] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, vol. 23. Englewood Cliffs, NJ, USA: Prentice-Hall, 1989.
- [3] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Autom. Control*, vol. 54, no. 1, pp. 48–61, Jan. 2009.
- [4] I. Lobel and A. Ozdaglar, "Distributed subgradient methods for convex optimization over random networks," *IEEE Trans. Autom. Control*, vol. 56, no. 6, pp. 1291–1306, Jun. 2011.
- [5] J. C. Duchi, A. Agarwal, and M. J. Wainwright, "Dual averaging for distributed optimization: Convergence analysis and network scaling," *IEEE Trans. Autom. Control*, vol. 57, no. 3, pp. 592–606, Mar. 2012.
- [6] K. Yuan, Q. Ling, and W. Yin, "On the convergence of decentralized gradient descent," *SIAM J. Optim.*, vol. 26, no. 3, pp. 1835–1854, 2016.
- [7] A. Nedić and A. Olshevsky, "Stochastic gradient-push for strongly convex functions on time-varying directed graphs," *IEEE Trans. Autom. Control*, vol. 61, no. 12, pp. 3936–3947, Dec. 2016.
- [8] W. Shi, Q. Ling, G. Wu, and W. Yin, "Extra: An exact first-order algorithm for decentralized consensus optimization," *SIAM J. Optim.*, vol. 25, no. 2, pp. 944–966, 2015.
- [9] A. Olshevsky, "Linear time average consensus and distributed optimization on fixed graphs," *SIAM J. Control Optim.*, vol. 55, no. 6, pp. 3990–4014, 2017.
- [10] W. Shi, Q. Ling, G. Wu, and W. Yin, "A proximal gradient algorithm for decentralized composite optimization," *IEEE Trans. Signal Process.*, vol. 63, no. 22, pp. 6013–6023, Nov. 2015.
- [11] Z. Li, W. Shi, and M. Yan, "A decentralized proximal-gradient method with network independent step-sizes and separated convergence rates," *IEEE Trans. Signal Process.*, vol. 67, no. 17, pp. 4494–4506, Sep. 2019.
- [12] K. I. Tsianos, S. Lawlor, and M. G. Rabbat, "Push-sum distributed dual averaging for convex optimization," in *Proc. IEEE Conf. Decis. Control*, 2012, pp. 5453–5458.
- [13] A. Nedić and A. Olshevsky, "Distributed optimization over time-varying directed graphs," *IEEE Trans. Autom. Control*, vol. 60, no. 3, pp. 601–615, Mar. 2015.
- [14] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *Proc. IEEE 44th Annu. Symp. Found. Comput. Sci.*, 2003, pp. 482–491.
- [15] R. Xin and U. A. Khan, "A linear algorithm for optimization over directed graphs with geometric convergence," *IEEE Contr. Syst. Lett.*, vol. 2, no. 3, pp. 315–320, Jul. 2018.
- [16] S. Pu, W. Shi, J. Xu, and A. Nedic, "Push-pull gradient methods for distributed optimization in networks," *IEEE Trans. Autom. Control*, vol. 66, no. 1, pp. 1–16, Jan. 2020.

- [17] Z. Feng and G. Hu, "A distributed constrained optimization approach for spatiotemporal connectivity-preserving rendezvous of multi-robot systems," in *Proc. IEEE Conf. Decis. Control*, 2018, pp. 987–992.
- [18] J. A. Fax and R. M. Murray, "Information flow and cooperative control of vehicle formations," *IFAC Proc. Volumes*, vol. 35, no. 1, pp. 115–120, 2002.
- [19] Z. Jiang, A. Balu, C. Hegde, and S. Sarkar, "Collaborative deep learning in fixed topology networks," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 5904–5914.
- [20] Y. Esfandiari, K. Nagasubramanian, F. Fotouhi, P. S. Schnable, B. Ganapathysubramanian, and S. Sarkar, "Distributed deep learning for persistent monitoring of agricultural fields," in *Proc. NeurIPS AI Sci. Workshop*, 2021, pp. 1–6.
- [21] Z. Jiang, A. Balu, C. Hegde, and S. Sarkar, "On consensus-optimality trade-offs in collaborative deep learning," *Front. Artif. Intell.*, vol. 4, 2021, Art. no. 573731.
- [22] J. W. Simpson-Porco, Q. Shafiee, F. Dörfler, J. C. Vasquez, J. M. Guerrero, and F. Bullo, "Secondary frequency and voltage control of islanded microgrids via distributed averaging," *IEEE Trans. Ind. Electron.*, vol. 62, no. 11, pp. 7025–7038, Nov. 2015.
- [23] D. K. Molzahn et al., "A survey of distributed optimization and control algorithms for electric power systems," *IEEE Trans. Smart Grid*, vol. 8, no. 6, pp. 2941–2962, Nov. 2017.
- [24] D. Jakovetić, J. Xavier, and J. M. Moura, "Fast distributed gradient methods," *IEEE Trans. Autom. Control*, vol. 59, no. 5, pp. 1131–1146, May 2014.
- [25] A. I.-A. Chen, "Fast distributed first-order methods," Ph.D. dissertation, Massachusetts Inst. of Technol., Cambridge, MA, USA, 2012.
- [26] B. Johansson, T. Keviczky, M. Johansson, and K. H. Johansson, "Subgradient methods and consensus algorithms for solving convex optimization problems," in *Proc. IEEE Conf. Decis. Control*, 2008, pp. 4185–4190.
- [27] A. S. Berahas, R. Bollapragada, N. S. Keshkar, and E. Wei, "Balancing communication and computation in distributed optimization," *IEEE Trans. Autom. Control*, vol. 64, no. 8, pp. 3141–3155, Aug. 2019.
- [28] V. Khatana and M. V. Salapaka, "DC-DistADMM: ADMM algorithm for constrained optimization over directed graphs," *IEEE Trans. Autom. Control*, vol. 68, no. 9, pp. 5365–5380, Sep. 2023.
- [29] V. Khatana, G. Saraswat, S. Patel, and M. V. Salapaka, "Gradient-consensus method for distributed optimization in directed multi-agent networks," in *Proc. IEEE Amer. Control Conf.*, 2020, pp. 4689–4694.
- [30] M. Prakash, S. Talukdar, S. Attree, V. Yadav, and M. V. Salapaka, "Distributed stopping criterion for consensus in the presence of delays," *IEEE Trans. Control Netw. Syst.*, vol. 7, no. 1, pp. 85–95, Mar. 2020.
- [31] A. Olshevsky and J. N. Tsitsiklis, "Convergence speed in distributed consensus and averaging," *SIAM J. Control Optim.*, vol. 48, no. 1, pp. 33–55, 2009.
- [32] J. Melbourne, G. Saraswat, V. Khatana, S. Patel, and M. V. Salapaka, "On the geometry of consensus algorithms with application to distributed termination in higher dimension," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 2951–2956, 2020.
- [33] V. Yadav and M. V. Salapaka, "Distributed protocol for determining when averaging consensus is reached," in *Proc. 45th Annu. Allerton Conf.*, 2007, pp. 715–720.
- [34] J. Wolfowitz, "Products of indecomposable, aperiodic, stochastic matrices," *Proc. Amer. Math. Soc.*, vol. 14, no. 5, pp. 733–737, 1963. [Online]. Available: <http://www.jstor.org/stable/2034984>
- [35] D. P. Bertsekas, "Nonlinear programming," *J. Oper. Res. Soc.*, vol. 48, no. 3, pp. 334–334, 1997.
- [36] G. Qu and N. Li, "Accelerated distributed Nesterov gradient descent," *IEEE Trans. Autom. Control*, vol. 65, no. 6, pp. 2566–2581, Jun. 2020.
- [37] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, vol. 87. Berlin, Germany: Springer, 2013.
- [38] P. Erdős and A. Rényi, "On the evolution of random graphs," *Pub. Math. Inst. Hung. Acad. Sci.*, vol. 5, no. 1, pp. 17–60, 1960.
- [39] P. Crescenzi, R. Grossi, C. Imbrenda, L. Lanzi, and A. Marino, "Finding the diameter in real-world graphs: Experimentally turning a lower bound into an upper bound," in *Proc. 18th Annu. Eur. Symp. Algorithms*, 2010, pp. 302–313.
- [40] F. Chung and L. Lu, "The diameter of sparse random graphs," *Adv. Appl. Math.*, vol. 26, no. 4, pp. 257–279, 2001.



Vivek Khatana (Student Member, IEEE) received the bachelor's degree in electrical engineering from the Indian Institute of Technology, Roorkee, India, in 2018. He is currently a Ph.D. candidate with the Department of Electrical Engineering, the University of Minnesota, Minneapolis, MN, USA. His research interests include distributed optimization, distributed and stochastic control, networks of dynamical systems, data-driven system identification, and control of power systems.



Govind Saraswat received the B.Tech degree in electrical engineering from the Indian Institute of Technology, Delhi, India, in 2007 and the Ph.D. degree in electrical engineering from the University of Minnesota, Twin Cities, Minneapolis, MN, USA, in 2014. He is a part of the Sensing and Predictive Analytics Group with the National Renewable Energy Laboratory, Golden, CO (NREL), where he is on data-driven technology for energy systems planning and operation. His research interests include power system modeling and analysis, measurement-based operation and control, machine learning, and optimization.



Sourav Patel received the B.Tech. degree in instrumentation and control engineering from the National Institute of Technology, Jalandhar, India, in 2011 and the M.S. degree in electrical engineering and Ph.D. degree with a focus on power engineering from the University of Minnesota, Twin Cities, Minneapolis, MN, USA, in 2018 and 2021, respectively. He is currently a Control System Engineer with Data Center, Google, Mountain View, CA, U.S. In 2011, he was with the National Thermal Power Corporation, Ltd., Kaniha, India, as a Control and Instrumentation Engineer.



Murti V. Salapaka (Fellow, IEEE) received the B.Tech. degree in mechanical engineering from the Indian Institute of Technology, Madras, India, in 1991 and the M.S. and Ph.D. degrees in mechanical engineering from the University of California at Santa Barbara, Santa Barbara, CA, USA, in 1993 and 1997, respectively. He was a Faculty Member with the Electrical and Computer Engineering Department, Iowa State University, Ames, IA, USA, from 1997 to 2007. He is currently the Director of Graduate Studies and the Vincentine Hermes Luh Chair Professor with the Electrical and Computer Engineering Department, University of Minnesota, Minneapolis, MN, USA. His research interests include control and network science, nanoscience, and single-molecule physics. Dr. Salapaka was the recipient of the NSF CAREER Award and the ISU Young Engineering Faculty Research Award for the years 1998 and 2001.