Distributed Optimisation With Communication Delays

Shuubham Ojha, Ketan Rajawat

Abstract—This paper discusses distributed optimization over a directed graph. We begin with some well known algorithms which achieve consensus among agents including FROST [1], which possesses the quickest convergence to the optimum. It is a well known fact FROST has a linear convergence. However FROST works only over fixed topology of underlying network. Moreover the updates proposed therein require perfectly synchronized communication among nodes. Hence communication delays among nodes, which are inevitable in a realistic scenario, preclude the possibility of implementing FROST in real time. In this paper we introduce a co-operative control strategy which makes convergence to optimum robust to communication delays.

Index Terms—Distributed optimization, (Fast row-stochastic optimization with uncoordinated steps) FROST algorithm, multi-agent systems, convergence rate, time varying delay, co-operative control, distributed estimation.

I. INTRODUCTION

Distributed optimization has already been a subject of research for its numerous advantage over a centralized network. A decentralized network involves independent optimization of local functions at each node unlike a centralized optimization which helps in parallel processing of data at the same time as well as in privacy preservation because each node is aware of its local function only.

In a classical consensus problem, the agent acquires the information from its neighbouring nodes and subsequently updates its own value. The distributed FROST algorithm [1] converge to the optimizer of the sum of local functions under certain graph topologies (for a smooth and fixed functions).

The presence of adversarial nodes along with non-adversarial ones botches up the attainment of global optimum. This scenario can be seen as a cyber-physical attack in a distributed network and a scale of vulnerability must be measured for secure data transmission. [2] gives a certain guarantee that the regular nodes can attain consensus under malicious behaviour though they are still far away from the global optimum of the objective function. In this paper we shall focus on adversary free networks.

In a later section we go through the FROST algorithm [1] for row-stochastic weights which can be applied for both directed and undirected graphs and the algorithm converges linearly to the global optimum for a strongly convex and smooth function. Most of the existing algorithms were based on column-stochastic weights but such algorithms require the nodes to know their out neighbours which is not the case for broadcast model of communication.

Our simulations suggest that FROST does not guarantee convergence to the global optimum in presence communication delays as well as abrupt changes to topology. In [3] the authors introduce a control law which dynamically estimates the first left eigenvector of the network adjacency matrix for changing topologies and communication delay between nodes. This estimated eigenvector is further used by a First Order Dynamic Average Consensus (FODAC) Algorithm to track the average gradient of all the local functions. The tracked average is then used in a gradient descent equation to reach the global optimum.

In this paper, we relax the requirement of synchronised communication while employing FROST by introducing a distributed co-operative control law with an associated gain variable. We assume that the nodes are unaware of each others local function. The main objective of this paper is to reach global optimum of the sum of these local functions and it is achieved under a suitable choice for the gain parameter.

The paper is organized in the following manner. In section II we draw a basic idea and terminology for the distributed network. In section III we revisit some related work and existing algorithms in distributed optimization. Section IV delineates the algorithm development under certain assumptions. Section V gives the convergence analysis while section VI provides simulation results to substantiate the theoretical results. Section VII concludes the work with possible directions for future work.

II. TERMINOLOGY FOR A DISTRIBUTED NETWORK

A graph is defined by \( \mathcal{G} = (V, \mathcal{E}) \), which contains a set of vertices (or nodes) \( V = \{v_i\}, \forall i \in \{1, \ldots, N\} \), and a set of edges represented by \( \mathcal{E} \subset (V \times V) = \{(v_i, v_j)\}, \forall i \neq j \). The graph is called directed if it satisfies either \( (v_i, v_j) \in \mathcal{E} \) or \( (v_j, v_i) \in \mathcal{E} \) and undirected if satisfies both. The set of in-neighbours for a node \( v_i \in V \) is defined by \( N_i^- \triangleq \{v_j \in V | (v_j, v_i) \in \mathcal{E}\} \), which signify the set of agents that can send information to \( v_i \), while the out-neighbour set is defined as \( N_i^+ \triangleq \{v_j \in V | (v_i, v_j) \in \mathcal{E}\} \). The in-degree and out-degree is the size or cardinality of respective set represented as \( d^-_i \triangleq |N_i^-| \) and \( d^+_i \triangleq |N_i^+| \). Clearly for a undirected graph \( d^-_i = d^+_i = d_i \). We define the degree matrix as \( D = \{d_{ij}\} \) with \( d_{ii} = \sum_{j=1}^{N} a_{ij} \). The Laplacian \( L \) is defined as \( L = D - A \).

We denote first left eigenvector of \( A \) by \( u = [u_1 \ldots u_N] \). A graph is said to be rooted if it is rooted at some vertex \( v_i \in V \) and strongly connected if there is a path from each node to other node in the graph. The normalised Laplacian is defined as \( L' = D^{-1} L \). Throughout this paper we assume that matrix \( A \) is row stochastic.
III. EXISTING ALGORITHMS

We discuss some related work in distributed optimization for with and without adversaries scenario.

For a distributed network each agent has its own local function \( f_i \) such that \( f_i : \mathbb{R} \rightarrow \mathbb{R} \) be convex with bounded sub gradients and only available to agent \( v_i \). To solve a global optimization problem in a distributed manner we formulate the objective function as

\[
\min_{x \in \mathbb{R}^n} F(x) = \frac{1}{N} \sum_{i=1}^{N} f_i(x)
\]  

(DGD (Distributed Gradient Descent) is a popular algorithm for doubly-stochastic (undirected graph) weights where each agent \( v_i \) maintains a local estimate \( x_i(t+1) \) and implements the updated iteration as follows:

\[
x_i(t+1) = \sum_{j=1}^{N} a_{ij} x_j(t) - \alpha_i \nabla f_i(x_i(t))
\]  

\( a_{ij} \) denotes the doubly-stochastic weights, \( \alpha_i \) is the diminishing step size satisfying \( \sum_{i=0}^{\infty} \alpha_i = \infty \) and \( \sum_{i=0}^{\infty} \alpha_i^2 < \infty \) and \( \nabla f_i(x_i(t)) \) gives the local gradient calculated at each node. To accelerate the convergence rate of DGD a method based on gradient tracking is proposed which does not involve calculating gradient at each node. The new iteration is:

\[
x_i(t+1) = \sum_{j=1}^{N} a_{ij} x_j(t) - \alpha_i y_i(t)
\]

\[
y_i(t+1) = \sum_{j=1}^{N} a_{ij} y_j(t) + \nabla f_i(x_i(t+1)) - \nabla f_i(x_i(t))
\]

Here \( y_i(t) \) asymptotically tracks the average of local gradients:

\[
\frac{1}{N} \sum_{i=1}^{N} \nabla f_i(x_i(t))
\]

It is shown in [1] that \( x_i(t) \) converges linearly to \( x^* \) for a strongly connected convex function with sufficiently small and constant step size with \( \alpha_i = \alpha \). For column-stochastic graphs algorithms like push-sum consensus, ADD-OPT algorithm and A\( \beta \) algorithm are discussed. An additional algorithm based on the gradient tracking called FROST algorithm is introduced for a row-stochastic weight matrix. The updates involved can be given as follows [1]:

\[
y_i(t+1) = \sum_{j=1}^{N} a_{ij} y_j(t)
\]

\[
x_i(t+1) = \sum_{j=1}^{N} a_{ij} x_j(t) - \alpha_i z_i(t)
\]

\[
z_i(t+1) = \sum_{j=1}^{N} a_{ij} z_j(t) + \frac{\nabla f_i(x_i(t+1))}{y_i(t+1)} - \frac{\nabla f_i(x_i(t))}{y_i(t)}
\]

where \( \alpha_i \)'s are the uncoordinated step-sizes locally chosen at each agent and the row-stochastic weights, \( A = \{a_{ij}\} \), respect the graph topology such that:

\[
a_{ij} = \begin{cases} 
> 0, & j \in N_i^+, \\
0, & \text{otherwise}, \\
\sum_{j=1}^{n} a_{ij} = 1, & \forall i.
\end{cases}
\]

All these aforementioned algorithms however work for a healthy network which does not experience any adversarial attack. In the presence of adversaries these algorithms do not achieve the global optimum, as suggested by our simulations. [2] suggests that consensus can only be achieved in a convex hull of the local function’s minimizers.

The updates in [2] are designed differently to deal with adversaries and are given as:

\[
x_i(t+1) = a_{ii}(t)x_i(t) + \sum_{j \in N_i^-} a_{ij}(t)x_j(t) - \alpha_i d_i(t)
\]

In this update it always satisfies that \( a_{ii}(t) + \sum_{j \in N_i^-} a_{ij}(t) = 1 \) and \( d_i(t) \) is a sub-gradient of local function \( f_i \), evaluated at \( a_{ii}(t)x_i(t) + \sum_{j \in N_i^-} a_{ij}(t)x_j(t) \).

The equation (5) can be compactly written as

\[
x(t+1) = A(t)x(t) - \alpha d(t)
\]

where \( a_{ij}(t) \) and \( a_{ii} \) are the \((i,j)^{th}\) entry and diagonal elements of \( A(t) \) matrix respectively.

In [2] the authors proposed a filtering technique such that \( v_i \) node first sorts out the values from it’s in-neighbours and eliminates \( F \) smallest as well as \( F \) largest values compared to its own value, where \( F \) is an upper bound on the number of adversarial nodes in the network. This process can be called as a filtering technique as it basically tries to reduce the effect of extreme values to attain the consensus. In this process a node eliminates some adversarial node values along with some regular nodes values which restricts the nodal values in magnitude and forces them to reach consensus.

Equation (5) when combined with the above filtering process results in:

\[
x_i(t+1) = a_{ii}(t)x_i(t) + \sum_{j \in N_i^-} a_{ij}(t)x_j(t) - \alpha_i d_i(t)
\]

Where \( J_i(t) \subset N_i^- \) implies a set of in-neighbours of \( v_i \) node and \( d_i(t) \) is a sub-gradient of local function \( f_i \), evaluated at \( a_{ii}(t)x_i(t) + \sum_{j \in J_i(t)} a_{ij}(t)x_j(t) \). Equation (10) is also known as Local Filtering (LF) dynamics.

IV. ALGORITHM DEVELOPMENT

The FROST algorithm can only be used under certain assumptions on the local functions and graph topology. These assumptions are reproduced below for ease:

A1. The underlying graph is directed and strongly connected.

A2. Each local function is convex with bounded sub-gradient.

A3. Each local function \( f_i \) is Lipschitz smooth with constants \( l_i \) is strongly convex with \( \sigma f_i \).
A4. We have \[ \sum_{i=1}^{N} \frac{l_i^2}{N} > \frac{3}{4} \forall i \in \{1, \ldots, N\}. \]

An amended algorithm incorporating time delay in FROST algorithm is proposed here. We begin by reproducing the iterations from [5] as follows:

\begin{align}
\dot{r}_i(n) &= \dot{z}_i(n) - \kappa' \sum_{i,j \in E} a_{ij}(r_i(n) - r_j(n-k)) \tag{11a} \\
\dot{s}_i(n) &= \dot{g}_i(n) - \kappa' \sum_{i,j \in E} a_{ij}(s_i(n) - s_j(n-k)) \tag{11b}
\end{align}

where,

\[ g_i(n) = 0.5(\text{sgn}(\sin2\pi \frac{n}{T_g}) + 1) \]

with \( z(n) = [z_i(n)]_{1 \leq i \leq N}^T \) being input dynamic state whose average is being tracked, \( k \) being time varying delay, \( y_i(0) = g_i(0) \) and \( \kappa' \) is the gain \( \kappa \) normalised by \( d_i \), which is the \( i^{th} \) diagonal entry of the corresponding degree matrix.

If the system in (11) reaches a consensus with \( r_i = c'_r \) and \( s_i = c'_s \) \( \forall i \) and the change in vector \( z \) equals, \( z = \Delta z \), the new equilibrium states \( c_r \) and \( c_s \) are given by [5]:

\[ c_r = c'_r + \frac{u \Delta z}{1 + \kappa \tau} \]
\[ c_s = c'_s + \frac{\Delta g}{1 + \kappa \tau} \]

Since \( |\Delta g| = 1 \), initialising \( c'_r = \frac{u a_0(0)}{1 + \kappa \tau} \) and ensuring that changes in \( z \) coincide with rising and falling edges of \( g \) we have,

\[ c_p = \frac{c_r}{c_s - c'_s} \]

tracking the weighted average of components of \( z \) where components are weighed by \( u \).

The parameter \( \kappa \) is chosen as \( \min(\frac{1}{2\sigma_2}, \max \lambda_i(L')) \) with \( \lambda_1 \) and \( \lambda_{N-1} \) being the largest and smallest non zero eigenvalues of \( L' \) and \( \max \lambda_i(L') \) denoting the second largest eigenvalue of \( L' \) [5]. The parameter \( T_g \) is chosen as \( T_g \geq 2(T_M + k' - 1) \) where \( T_M \) is the worst case convergence time which can be determined experimentally [3] and \( k' - 1 \) is the maximum possible value of delay.

In (11) \([a_{ij}]_3\) is a doubly stochastic matrix. For a row stochastic setting these equations take the form.

\begin{align}
e_i(n+1) &= \sum_{j=1}^{N} a_{ij} e_j(n) \tag{12a} \\
\dot{r}_i(n) &= \frac{z_i(n)}{e_{ii}(n)} - \frac{z_i(n-1)}{e_{ii}(n-1)} - \kappa' \sum_{i,j \in E} a_{ij}(r_i(n) - r_j(n-k)) \tag{12b} \\
\dot{s}_i(n) &= \frac{g_i(n)}{c_r(n-1)} - \kappa' \sum_{i,j \in E} a_{ij}(s_i(n) - s_j(n-k)) \tag{12c} \\
p_i(n) &= \sum_{l=0}^{n-1} \frac{c_r(T_g)}{c_s^n - c'_s} \tag{12d}
\end{align}

where \( e_{ii}(n) \) denotes the \( n^{th} \) iteration estimate of the \( i^{th} \) component of first left eigenvector and \( c_r(T_g) \) denotes the equilibrium consensus value of \( x_i \) at node \( i \) just before the arrival of the \( (r+1)^{th} \) edge of \( g_i \) for every \( i \).

We now utilise the average tracker from (12) in the gradient descent equation to reach the global optimum. This leads to following iterates:

\begin{align}
e_i(n+1) &= \sum_{j=1}^{N} a_{ij} e_j(n) \tag{13a} \\
\dot{r}_i(n) &= \frac{\nabla f_i(x_i(n))}{e_{ii}(n)} - \frac{\nabla f_i(x_i(n-1))}{e_{ii}(n-1)} \sum_{l=0}^{\infty} \delta(n-l T_g) - \kappa' \sum_{i,j \in E} a_{ij}(r_i(n) - r_j(n-k(n))) \tag{13b} \\
\dot{s}_i(n) &= \frac{g_i(n)}{c_r(n-1)} - \kappa' \sum_{i,j \in E} a_{ij}(s_i(n) - s_j(n-k(n))) \tag{13c} \\
x_i(n) &= \sum_{j=1}^{N} a_{ij} x_j(n) - \kappa \sum_{i,j \in E} a_{ij}(x_i(n-1) - x_j(n-k(n))) + \alpha \delta(T_g) \tag{13d} \\
p_i(n) &= \sum_{l=0}^{n-1} \frac{c_r(T_g)}{c_s^n - c'_s} + (u \otimes I_n) \left[ \frac{\nabla f_i(x_i(0))}{e_{ii}(0)} \right]^T \tag{13e}
\end{align}

where \( k_{ij}(n) \) is the delay on the link from node \( j \) to node \( i \) at time \( n \).

V. Convergence Analysis

Let \( [\nabla f_i(k)]_{1 \leq i \leq N} = \left[ \frac{\nabla f_i(x_k(i))}{e_{ii}(k)} \right]^T \). It can be seen that (13e) converges to the weighted average of gradients, \( \nabla f_i(x_k(n)) \), as \( n \) approaches time instants \( m T_g \) for \( m \in \mathbb{Z} \) as:

\[ c_r(T_g) \frac{c_r(T_g)}{c_s^n - c'_s} = (u \otimes I_n) \left[ \nabla f_i \left( \frac{n T_g}{2} \right) - \nabla f_i \left( \frac{(l-1) T_g}{2} \right) \right]_{1 \leq i \leq N} \]

Substituting in (13e) we have,

\[ p_i \left( \frac{n T_g}{2} \right) = (u \otimes I_n) \left[ \nabla f_i \left( \frac{n T_g}{2} \right) \right]_{1 \leq i \leq N} \]
We have established the convergence of (13b), (13c) and (13e). In what follows we shall establish the convergence of (13d) to the optimum \( x^* \) for each agent. To this end we shall first show in Lemma \( \square \) that each agent converges to a consensus value. This is followed by Theorem \( \square \) and Theorem \( \square \) which will give the final convergence result.

**Lemma 1.** Let, \( \nabla F \) be Lipschitz continuous with constant \( L_f > 0 \). If \( F \) is strongly convex with parameter \( \sigma_f \), then

\[
(x - y, \nabla F(x) - \nabla F(y)) \geq \mu_1 \| \nabla F(x) - \nabla F(y) \|^2 + \mu_2 \| x - y \|^2,
\]

for all \( x, y \in F \), with \( \mu_1 = \frac{1}{\sigma_f + L_f} \) and \( \mu_2 = \frac{\sigma_f L_f}{\sigma_f + L_f} \).

**Proof.** Proof can be found in Appendix A. \( \square \)

In the following \( k \) denotes the time instant \( \frac{kT_f}{2} \).

**Lemma 2.** Let, \( x(k) = [x_i(k)]_{i \leq s \leq N} \) be the sequence generated in (13d) and \( \bar{u} = u \otimes I_n \). Then, we have

\[
\| (1_N^T \otimes \bar{u}) x(k) - x(k) \| = \| (A \otimes I_n - (1_N^T \otimes \bar{u})) k x(0) \|
\]

**Proof.**

\[
x(k - 1) = (A \otimes I_n - 1_N^T \otimes \bar{u}) x(k - 2) + (1_N^T \otimes \bar{u}) x(0)
\]

Repeating this equation for \( k - 1 \) gives,

\[
x(k) = (A \otimes I_n - 1_N^T \otimes \bar{u}) x(k - 2) + (1_N^T \otimes \bar{u}) x(0)
\]

for all \( k > k_0 \).

Define \( \hat{p}(k) = \sum_{i=1}^{N} \frac{u_i \nabla f_i(\bar{u}_i(k))}{e_{i_i(k)}} \) and \( p(k) = [p_1(k) \ldots p_N(k)]^T \). From Lemma 2 we have, \( k_0 \) such that \( \forall k > k_0 \)

\[
\| 1_N^T \otimes \hat{p}(k) - p(k) \| \leq \sum_{i=1}^{N} \frac{u_i \nabla f_i(\bar{u}_i(k))}{e_{i_i(k)}} - \frac{u_i \nabla f_i(x_i(k))}{e_{i_i(k)}} \leq 2 \mu_1 \| \nabla f_i(x_i(k)) \| + \mu_2 \| x_i(k) \| = L_k \sum_{i=1}^{N} \frac{u_i}{e_{i_i(k)}} \approx N L_k \varepsilon
\]

for "large" \( k \).

**Theorem 1.** Let Assumptions \( A \sqcup A \) hold. Let \( x(k) = [x_i(k)]_{i \leq s \leq N} \) be the sequence of estimates generated by (13d). Let \( L_k = \max_{1 \leq s \leq N} l_i \) and \( \mu_1, \mu_2 \) be the corresponding constants for \( f_i \) as defined in Lemma 1. Then \( \exists k_1 \) such that \( \forall k \geq k_1 \) we have

\[
\| \bar{u}(k + 1) - x^* \|^2 \leq \alpha_1(k) \| \bar{u}(k) - x^* \|^2 + O(\varepsilon).
\]

where \( \alpha_1(k) = 4 + 4 N \sum_{i=1}^{N} \left( \frac{l_i \alpha_k u_i}{e_{i_i(k)}} \right)^2 - 8 \alpha_k u_i \frac{l_i}{e_{i_i(k)}} \)

**Proof.**

\[
\| \bar{u}(k + 1) - x^* \|^2 \leq \alpha_k | \bar{u}(k) - [\nabla f_i(x_i(k))]_{1 \leq i \leq N} - x^* |^2
\]

\[
= \| \bar{u}(k) - \alpha_k(\bar{u}_k - \alpha_k \hat{p}(k) - \alpha_k \hat{p}(k) - x^*) \|^2
\]

\[
\leq 2 \| \bar{u}(k) - x^* - \alpha_k \hat{p}(k) \|^2
\]

\[
+ 2 \varepsilon \| \sum_{i=1}^{N} \frac{u_i \nabla f_i(x_i(k))}{e_{i_i(k)}} - \hat{p}(k) \|^2
\]

From (15) there is \( k_0 \) so that \( \forall k > k_0 \) the second term is bounded by \( \varepsilon \). The first term in (16) can be written as,

\[
2 \| \bar{u}(k) - x^* \|^2 - \alpha_k \sum_{i=1}^{N} \frac{u_i}{e_{i_i(k)}} | \nabla f_i(\bar{u}_i(k)) - \nabla f_i(x_i(k)) |^2
\]

\[
+ \sum_{i=1}^{N} \frac{\alpha_k u_i}{e_{i_i(k)}} \| \nabla f_i(x_i^*) \|^2
\]

From optimality condition, \( \sum_{i=1}^{N} \nabla f_i(x^*) = 0 \). Hence,

\[
= 2 \| \bar{u}(k) - x^* \|^2 - \alpha_k \sum_{i=1}^{N} \frac{u_i}{e_{i_i(k)}} | \nabla f_i(\bar{u}_i(k)) - \nabla f_i(x_i^*) |^2
\]

\[
+ \sum_{i=1}^{N} \frac{\alpha_k u_i}{e_{i_i(k)}} \| \nabla f_i(x_i^*) \|^2 + \alpha_k \sum_{i=1}^{N} \| \nabla f_i(x_i^*) \|^2
\]
The first term in (20) can be bounded as,

\[ \leq 4\|\bar{u}x(k) - x^*\| - \alpha_k \sum_{i=1}^{N} \frac{u_i}{e_{ii}(k)} \left( \nabla f_i(\bar{u}x(k)) - \nabla f_i(x^*) \right) + 4 \sum_{i=1}^{N} \alpha_k \nabla f_i(x^*) \left( 1 - \frac{u_i}{e_{ii}(k)} \right) \| \]

(19)

Thus, there is \( k_2 \) so that \( \forall k > k_2 \) the R.H.S. in (23) is less than \( \varepsilon \). Choosing \( k_1 = \max\{k_0, k_2\} \) and combining (16), (20), (21), (23) we get the result.

\[ \Box \]

**Theorem 2.** Let Assumptions [A1, A4] hold. Let \( \{x_i(k)\}_{1 \leq i \leq N} \) be the sequence generated in (13d). Then \( \forall i \) there is \( k_3 \) such that for each \( k \geq k_3 \),

\[ \|x_i(k) - x^*\| < \varepsilon(\varepsilon) \]

under appropriate choice of step size \( \alpha_k \).

**Proof.** We must choose \( \alpha_k \) so that \( |a_i(k)| < 1 \) for each \( k \geq 0 \). Observe that \( a_i(k) \) is a quadratic in \( \alpha_k \). Thus, rearranging \( a_1(k) \) in Theorem 2 it is desired that,

\[ 0 < A_k \alpha_k^2 + B_k \alpha_k + 4 < 1 \]

(24)

where,

\[ A_k = 4 \sum_{i=1}^{N} \left( \frac{u_i}{e_{ii}(k)} \right)^2, \quad B_k = \sum_{i=1}^{N} -8 \frac{u_i^2}{e_{ii}(k)} \]

The minimum value of the expression above is \( 4 - \frac{B_k^2}{4A_k} \), which is obtained at \( \alpha_k = \frac{-B_k + \sqrt{B_k^2 - 4A_k}}{2A_k} \). By Assumption [A4] we have, \( \left( \sum l_i \right)^2 > \frac{k^2}{2} \) and from Cauchy-Schwarz Inequality we have, \( \left( \sum l_i \right)^2 < 1 \). Thus, \( 0 < 4 - \frac{B_k^2}{4A_k} < 1 \).

Reproducing the relation from Theorem 1 for \( k = 0 \), we have,

\[ \|\bar{u}x(1) - x^*\|^2 \leq a_1(0)\|\bar{u}x(0) - x^*\|^2 + O(\varepsilon). \]

(25)

and for \( k = 1 \),

\[ \|\bar{u}x(2) - x^*\|^2 \leq a_1(1)\|\bar{u}x(1) - x^*\|^2 + O(\varepsilon). \]

(26)

Substituting (25) in (26) gives,

\[ \|\bar{u}x(2) - x^*\|^2 \leq a_1(1)a_1(0)\|\bar{u}x(0) - x^*\|^2 + O(\varepsilon). \]

(27)

Generalizing,

\[ \|\bar{u}x(k) - x^*\|^2 \leq \prod_{s=0}^{k-1} a_1(s)\|\bar{u}x(s) - x^*\|^2 + O(\varepsilon). \]

(28)

Since \( a_1(s) < 1 \) for each \( s \) by our choice of \( \alpha_s \), \( \exists k_3 \) such that \( \forall k > k_3 \) \( \prod_{s=0}^{k-1} a_1(s) < \varepsilon \), which immediately gives the result in the theorem.

\[ \Box \]
VI. SIMULATIONS

For the purpose of simulations we analyse two different cases. It is easy to verify that the functions assigned to nodes in both the scenarios are consistent with the assumptions $A_{1-A_{4}}$. In the first case $f_{i}(x)$ is assigned to be $(x + i)^2 \forall i$. In this case the local minimas are regularly spaced and the global minima coincides with the average of local minimas. We observe that in accordance with our theoretical analysis, convergence to global minima for algorithm (13) is observed in the figures 1 and 2.

In the second scenario we force the minima of the first local function to become an outlier. Hence our choice of local functions is as follows:

\[ f_{i}(x) = \begin{cases} 
(x + i)^2, & \text{if } i \neq 1 \\
(x + 100)^2, & \text{otherwise} 
\end{cases} \]

The convergence to global minima in this case for agents 1 and 10 can be observed in Figures 3 and 4 respectively. We demonstrate the utility of the iterates in (13a-e) by comparing the average tracker in (13b) and (13c) with a naive scheme where the gradients are being communicated by a node to its out neighbours, followed by a direct averaging. In our system of 22 nodes, the nodes have been initialised as $r_{i}(1) = \nabla f_{i}(x_{i}(0)) e_{ii}(0)$. The nodes are to track the average of these quantities weighted by elements of the first left eigenvector of the adjacency matrix at every $T_{g}$ time steps. The maximum allowed delay between nodes is 157 units which corresponds to the choice $\kappa = 0.01$. Under naive averaging each node will have to wait for 157 x 21 = 3297 time units in the worst case for obtaining the local gradient information from all the nodes. However, under average tracking we see that the nodes approach the weighted average well within 3000 time units. This is summarised in Fig. 5. Additionally, we compare the nature of convergence to the optimum with average gradient tracking and with naive averaging in Fig. 6.
In this paper, we consider distributed optimization for graphs with row-stochastic weights. Most of the existing algorithms are based on synchronised communication among agents, which may be infeasible to implement in many practical scenarios since communication delays are inevitable. We propose an algorithm inspired by FROST and show that it converges under appropriate choice of step sizes. Our algorithm inherits the convergence properties of FROST while being robust to time varying communication delays. Simulation results substantiate our theoretical claims. Several possibilities of future work include replicating our algorithm in an adversarial setting while ensuring convergence to the optimum.

VII. CONCLUSION

In this paper, we consider distributed optimization for graphs with row-stochastic weights. Most of the existing algorithms are based on synchronised communication among agents, which may be infeasible to implement in many practical scenarios since communication delays are inevitable. We propose an algorithm inspired by FROST and show that it converges under appropriate choice of step sizes. Our algorithm inherits the convergence properties of FROST while being robust to time varying communication delays. Simulation results substantiate our theoretical claims. Several possibilities of future work include replicating our algorithm in an adversarial setting while ensuring convergence to the optimum.

REFERENCES

[1] R. Xin, C. Xi, and U. A. Khan, “Frost—fast row-stochastic optimization with uncoordinated step-sizes,” EURASIP Journal on Advances in Signal Processing, vol. 2019, no. 1, pp. 1–14, 2019.

[2] S. Sundaram and B. Gharesifard, “Distributed optimization under adversarial nodes,” IEEE Transactions on Automatic Control, vol. 64, no. 3, pp. 1063–1076, 2018.

[3] Z. Qu, C. Li, and F. Lewis, “Cooperative control with distributed gain adaptation and connectivity estimation for directed networks,” International Journal of Robust and Nonlinear Control, vol. 24, no. 3, pp. 450–476, 2014.

[4] C. Xi and U. A. Khan, “Directed-distributed gradient descent,” in 2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton). IEEE, 2015, pp. 1022–1026.

[5] Y. Du, H. Tu, H. Yu, and S. Lukic, “Accurate consensus-based distributed averaging with variable time delay in support of distributed secondary control algorithms,” IEEE Transactions on Smart Grid, 2020.

[6] R. A. Horn and C. R. Johnson, Matrix analysis. Cambridge university press, 2012.

[7] V. S. Mai and E. H. Abed, “Distributed optimization over weighted directed graphs using row stochastic matrix,” in 2016 American Control Conference (ACC). IEEE, 2016, pp. 7165–7170.

[8] P.-A. Bliman, A. Nedic, and A. Ozdaglar, “Rate of convergence for consensus with delays,” in 2008 47th IEEE Conference on Decision and Control. IEEE, 2008, pp. 4849–4854.

VIII. APPENDIX A

Proof. Define,

\[ g(x) = F(x) - \frac{\sigma_f}{2} \|x\|^2 \]

Strong convexity of \( F(x) \) implies that \( g(x) \) is convex. Consider the function,

\[ h(x) = \frac{L_f - \sigma_f}{2} \|x\|^2 - g(x) = \frac{L_f}{2} \|x\|^2 - F(x) \]

Smoothness of \( F(x) \) implies that \( h(x) \) is convex which in turn implies that \( g(x) \) is smooth with parameter \( L_f - \sigma_f \).

Applying co-coercivity property for smooth functions on \( g(x) \) gives,

\[ \langle \nabla g(x) - \nabla g(y), x - y \rangle \geq \frac{1}{L_f - \sigma_f} \|\nabla g(x) - \nabla g(y)\|^2 \]

\[ \Rightarrow \langle \nabla F(x) - \nabla F(y) - \sigma_f(x - y), x - y \rangle \]

\[ \geq \frac{1}{L_f - \sigma_f} \|\nabla F(x) - \nabla F(y) - \sigma_f(x - y)\|^2 \]

\[ \Rightarrow (1 + \frac{2\sigma_f}{L_f - \sigma_f}) \langle \nabla F(x) - \nabla F(y), x - y \rangle \]

\[ \geq (\frac{1}{L_f - \sigma_f}) \|\nabla F(x) - \nabla F(y)\|^2 + (\frac{\sigma_f^2}{L_f - \sigma_f} + \sigma_f) \|x - y\|^2 \]

which gives the result in Lemma 3.

\[ \blacksquare \]