
A FAST ROW-STOCHASTIC DECENTRALIZED METHOD FOR DISTRIBUTED OPTIMIZATION OVER DIRECTED GRAPHS^{*†}

A PREPRINT

Diyako Ghaderyan

Research Center for Systems and Technologies(SYSTEC)
University of Porto(FEUP), Portugal
dghaderyan@fe.up.pt

Necdet Serhat Aybat

Industrial and Manufacturing Engineering Department
Penn State University
University Park, PA 16802,
nsa10@psu.edu

A. Pedro Aguiar

Research Center for Systems and Technologies(SYSTEC)
University of Porto(FEUP), Portugal
pedro.aguiar@fe.up.pt

Fernando Lobo Pereira

Research Center for Systems and Technologies(SYSTEC)
University of Porto(FEUP), Portugal
flp@fe.up.pt

ABSTRACT

In this paper, we introduce a fast row-stochastic decentralized algorithm, referred to as FRSD, to solve consensus optimization problems over directed communication graphs. The proposed algorithm only utilizes row-stochastic weights, leading to certain practical advantages in broadcast communication settings over those requiring column-stochastic weights. Under the assumption that each node-specific function is smooth and strongly convex, we show that the FRSD iterate sequence converges with a linear rate to the optimal consensus solution. In contrast to the existing methods for directed networks, FRSD enjoys linear convergence without employing a gradient tracking (GT) technique explicitly, rather it implements GT implicitly with the use of a novel momentum term, which leads to a significant reduction in communication and storage overhead for each node when FRSD is implemented for solving high-dimensional problems over small-to-medium scale networks. In the numerical tests, we compare FRSD with other state-of-the-art methods, which use row-stochastic and/or column-stochastic weights.

Keywords Distributed optimization, consensus, directed graphs, linear convergence, row-stochastic weights.

^{*}This work was supported by PDMA (NORTE-08-5369-FSE000061), SNAP - NORTE-01-0145-FEDER-000085, funded by ERDF NORTE2020/PORTUGAL2020; RELIABLE (PTDC/EEI-AUT/3522/2020), R&D Unit SYSTEC - Base (UIDB/00147/2020), Programmatic (UIDP/00147/2020) and ARISE (LA/ P/0112/2020) funded by national funds through FCT/MCTES (PIDDAC). The work of N. S. Aybat was supported by Office of Naval Research (ONR) through ONR research grant N00014-21-1-2271.

[†]The authors are listed according to their contribution to the work, from the most to the least.

1 Introduction

In recent years, rapid advances in artificial intelligence and communication technologies have led to computational network systems over which one has to solve optimization problems with enormous, physically distributed and/or private data sets in order to achieve system level objectives such that every agent, represented by a node in the network, has to agree on a common decision. To reach an optimal consensus decision, decentralized optimization techniques can be used to solve a consensus optimization problem in a distributed manner employing only local computations and communication among neighboring computing nodes that can directly communicate with each other. The classic consensus optimization problem has the following form:

$$\mathbf{x}^* \in \operatorname{argmin}_{\mathbf{x} \in \mathbb{R}^p} \bar{f}(\mathbf{x}) \triangleq \frac{1}{n} \sum_{i=1}^n f_i(\mathbf{x}), \quad (1)$$

where the objective function \bar{f} is the average of all individual cost functions $\{f_i\}_{i=1}^n$, where $f_i : \mathbb{R}^p \rightarrow \mathbb{R}$ is the private function of agent i . This problem appears in a variety of applications, e.g., sensor networks [1, 2], distributed control [3], large-scale machine learning [4–10], distributed estimation [11].

Next, we first discuss the previous work focusing on undirected networks, and then we summarize some related methods proposed for distributed consensus optimization over *directed* networks. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the network of collaborative agents, and $\tilde{\mathbf{x}}_i$ denote a local estimate of the optimal decision for $i \in \mathcal{V}$. For merely convex objectives, we call $\tilde{\mathbf{x}} = [\tilde{\mathbf{x}}_i]_{i \in \mathcal{V}}$ an ϵ -solution if $|\frac{1}{n} \sum_{i \in \mathcal{V}} f_i(\tilde{\mathbf{x}}_i) - f^*| \leq \epsilon$ and the consensus violation satisfies $\max\{\|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j\| : (i, j) \in \mathcal{E}\} \leq \epsilon$ for $i \in \mathcal{V}$, where f^* denotes the optimal value of (1). On the other hand, for strongly convex problems, we say that $\tilde{\mathbf{x}}$ is ϵ -optimal, if $\|\tilde{\mathbf{x}}_i - \mathbf{x}^*\| \leq \epsilon$ for all $i \in \mathcal{V}$.

Inspired by the seminal work [12], the authors in [13] proposed a distributed (sub)gradient method for solving (1). When each f_i is closed convex, the method in [13] is shown to have a sublinear convergence rate, i.e., to compute an ϵ -optimal solution, one needs to evaluate $\mathcal{O}(1/\epsilon^2)$ subgradients – in contrast to linear or $\mathcal{O}(1/\epsilon)$ sublinear convergence rates, the slower rate of subgradient methods is due to their use of a diminishing step-size sequence or of a small fixed step size $\alpha = \mathcal{O}(\epsilon)$. Moreover, when agents have simple closed convex constraint sets, distributed projected subgradient methods are proposed for solving

$$\min\{\bar{f}(\mathbf{x}) : \mathbf{x} \in \mathcal{X}\}; \quad (2)$$

e.g., the method in [14] solves (2) employing exact subgradient evaluations, while the method in [15] can handle subgradients corrupted by stochastic noise. In [16], algorithms based on dual averaging of subgradients are studied for solving (2) assuming $\mathbf{0} \in \mathcal{X}$. In [15], it is shown that an ϵ -optimal solution can be computed with $\mathcal{O}(n^3/\epsilon^2)$ iteration complexity that is independent of the network topology, whereas the algorithm proposed in [16] requires iteration complexity of $\mathcal{O}(n^2/\epsilon^2)$ for paths or simple cycle graphs, $\mathcal{O}(n/\epsilon^2)$ for 2- d grids, and $\mathcal{O}(1/\epsilon^2)$ for bounded degree expander graphs.

The authors in [17] propose a primal-dual subgradient algorithm to solve problems with a global constraint set defined as the intersection of local ones, i.e., $\mathcal{X} = \mathcal{X}_0 \cap (\bigcap_{i=1}^n \mathcal{X}_i)$ in (2) such that each agent- i only knows f_i , \mathcal{X}_i and \mathcal{X}_0 . In [18], under the assumption that the gradients are bounded and Lipschitz, an improved convergence rate of $\mathcal{O}(\log(k)/k^2)$ is obtained by employing Nesterov acceleration. In a similar setting, assuming convex functions with Lipschitz gradients, the EXTRA method [19], using a fixed step size and utilizing the difference of two consecutive gradients in each update, is shown to generate a sequence that converges with a $\mathcal{O}(1/k)$ rate, which improves to linear convergence under the additional assumption of strong convexity. There are also distributed algorithms based on alternating direction method of multipliers (ADMM) and the augmented Lagrangian method achieving similar rates, e.g., [20–23] –some handling composite convex functions [24, 25], and some even handling complicated constraints [26, 27].

The works that we have discussed above are designed for undirected networks; hence, they correspond to balanced graphs if we treat undirected networks as a special case of directed networks. However, directed networks may well be unbalanced; this situation arises especially for directed time-varying networks. For general directed networks, the first works to employ push-sum consensus protocol [28] (for computing an average over directed networks) within the distributed optimization framework (using the dual averaging method) are [9, 10]. A follow-up work in this direction is the subgradient-push method proposed in [29] that combines the push-sum protocol with the distributed subgradient method [13] (for minimization of convex functions). More precisely, the method in [29] applies to (1) when each f_i is a closed convex function and the directed communication network is time-varying, and achieves a sublinear rate of $\mathcal{O}(\log(k)/\sqrt{k})$ using a column-stochastic weight matrix and a diminishing step-size sequence. On the other hand, when each f_i is smooth (i.e., ∇f_i is Lipschitz continuous) and strongly convex, the DEXTRA algorithm proposed in [30], which is a distributed method for directed graphs, achieves R-linear convergence rate; it combines EXTRA [19] with push-sum approach [28]. The step-size in DEXTRA is constant and should be carefully chosen belonging to a

specific interval that may be unknown to the agents. Compared to DEXTRA, Push-DIGing [31] and ADD-OPT [32] have a simpler step-size rule and can achieve R-linear rate on directed graphs with time-varying and static topology, respectively, using sufficiently small constant step-size. These approaches employ a column-stochastic weight matrix to achieve R-linear rate over strongly connected networks. Unlike the previous methods that are based on push-sum, there are other works achieving linear convergence for the smooth and strongly convex setting by employing both column-stochastic and row-stochastic weights, e.g., \mathcal{AB} , \mathcal{ABm} and Push-Pull [33–35]. Later, \mathcal{ABN} method is proposed in [36] which incorporates Nesterov’s momentum term into \mathcal{AB} .

It is important to note that designing column-stochastic weights requires the knowledge of neighbors’ out-degree for each node; this requirement is impractical within broadcast-based communication systems. To address this issue, in [37], the authors proposed a method that only uses the *row-stochastic* weights. This line of research, i.e., using only *row-stochastic* weights, has attracted attention, and in follow-up papers, the algorithm in [37] is extended to handle uncoordinated step-sizes in the FROST algorithm [38], and to incorporate Nesterov acceleration leading to the FROZEN algorithm [36]. Finally, the D-DNGT algorithm proposed in [39] employs heavy-ball momentum and can handle nonuniform step-sizes. Some recent work extended the synchronous methods for directed networks to the asynchronous computation setting, in which agents asynchronously update their iterates by using the currently available (possibly old) information, and they do not wait for the other agents to update in order to proceed to the next update, i.e., there is no global clock, e.g., [40–42].

Contributions: In this paper, we propose a fast row-stochastic decentralized algorithm, referred to as FRSD, to solve distributed consensus optimization problems over directed communication networks. FRSD employs only row-stochastic weights, and we show that when $\{f_i\}_{i=1}^n$ are strongly convex and smooth, FRSD iterate sequence corresponding to a constant stepsize converges to the optimal consensus decision with a linear rate. While previous row-stochastic methods [36–39] crucially depend on the gradient tracking technique to establish linear rate, in this paper we achieve the same result through introducing a novel momentum term which leads to *implicit* gradient tracking, i.e., FRSD does *not* employ gradient tracking *explicitly* at the node level—hence, it does not require any computation simultaneously involving $\nabla f_i(\mathbf{x}_i(k+1))$ and $\nabla f_i(\mathbf{x}_i(k))$ for any node $i \in \mathcal{V}$ in the k -th iteration; but, it still manages to implement gradient tracking in an *implicit* manner. This new dynamics proposed in this paper leads to: (i) reduction in the data stored, and (ii) reduction in the data broadcast, for each node. More precisely, FRSD does not need to store \mathbf{x} iterate from the previous iteration while it is needed for all other methods explicitly using the gradient tracking term; furthermore, FRSD also eliminates the need for broadcasting a variable related to gradient tracking. In summary, in FRSD any agent- i only needs to store a $2p + n$ -dimensional vector, and to broadcast $n + p$ -dimensional vector. Comparing with the other row-stochastic methods Xi-row, FROZEN, and D-DNGT, communication requirement decreases from $2p + n$ to $p + n$. For settings where $p \gg n$, e.g., $n \approx 100$ nodes collectively solving an image/video processing problem with $p \approx 10^6$, this reduction is significant. The reduction in storage requirement is even more significant, see Table 1. In the numerical tests, we empirically show that FRSD is competitive against the other state-of-the-art methods: Xi-row [37], D-DNGT [39], FROZEN and \mathcal{ABN} [36], \mathcal{AB} [33], \mathcal{ABm} [34], Push-DIGing [31] and Push-Pull [35].

Notation: In this paper, the bold letters denote vectors, e.g., $\mathbf{x} \in \mathbb{R}^p$, and $[\mathbf{x}]_j$ denotes the j -th element of \mathbf{x} . The vector $\mathbf{0}_n$ and $\mathbf{1}_n$ represent the n -dimensional vectors of all zeros and ones. The uppercase of letters are reserved for matrices; given $X \in \mathbb{R}^{n \times n}$, $\text{diag}(X) \in \mathbb{R}^{n \times n}$ denotes the diagonal matrix of which diagonal is equal to that of $X \in \mathbb{R}^{n \times n}$. Moreover, given $\mathbf{v} \in \mathbb{R}^n$, $\text{diag}(\mathbf{v})$ is a diagonal matrix with its diagonal equal to \mathbf{v} . $I_n = [\mathbf{e}_1, \dots, \mathbf{e}_n]_{i=1}^n$ denotes the $n \times n$ identity matrix, where \mathbf{e}_i denotes the i -th unit vector. Throughout $\|\cdot\|$ denotes the Euclidean and the spectral norms depending on whether the argument is a vector or a matrix.

2 Design, Comparison and analysis of FRSD

Consider the consensus optimization problem (1) over a communication network which is represented as a *directed* graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} \triangleq \{1, 2, \dots, n\}$ is the set of nodes (agents), and \mathcal{E} is the set of directed communication links between the nodes. Each node $i \in \mathcal{V}$ has a private cost function $f_i : \mathbb{R}^p \rightarrow \mathbb{R}$, only known to node i . Furthermore, for each node $i \in \mathcal{V}$, we define its in-neighbors as the set of nodes that can send information to node i , i.e., $\mathcal{N}_i^{\text{in}} \triangleq \{j \in \mathcal{V} : (j, i) \in \mathcal{E}\} \cup \{i\}$. Since FRSD is a row-stochastic method, any node $i \in \mathcal{V}$ does not need to know its out-neighbors, i.e., the set of nodes receiving information from node i , which makes FRSD suitable for broadcast communication systems.

Throughout the paper we make the following assumptions.

Assumption 1. \mathcal{G} is directed and strongly connected.

Assumption 2. For every $i \in \mathcal{V}$, the local function f_i is L -smooth, i.e., it is differentiable with a Lipschitz gradient:

$$\|\nabla f_i(\mathbf{x}) - \nabla f_i(\mathbf{x}')\| \leq L\|\mathbf{x} - \mathbf{x}'\|, \quad \forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^p. \quad (3)$$

Assumption 3. For all $i \in \mathcal{V}$, f_i is μ -strongly convex, i.e.,

$$f_i(\mathbf{x}') \geq f_i(\mathbf{x}) + \nabla f_i(\mathbf{x})^\top (\mathbf{x}' - \mathbf{x}) + \frac{\mu}{2} \|\mathbf{x}' - \mathbf{x}\|^2 \quad (4)$$

for all $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^p$.

Remark 1. Under Assumption 3, the optimal solution to (1) is unique, denoted by \mathbf{x}^* .

Definition 1. Define $\mathbf{x} \triangleq [\mathbf{x}_1^\top, \dots, \mathbf{x}_n^\top]^\top \in \mathbb{R}^{np}$ and $\mathbf{y} \triangleq [\mathbf{y}_1^\top, \dots, \mathbf{y}_n^\top]^\top \in \mathbb{R}^{np}$, where $\mathbf{x}_i, \mathbf{y}_i \in \mathbb{R}^p$ are the local variables of agent- i for $i \in \mathcal{V} \triangleq \{1, \dots, n\}$, and in an algorithmic framework, their values at iteration $k \geq 0$ are denoted by $\mathbf{x}_i(k)$ and $\mathbf{y}_i(k)$ for $i \in \mathcal{V}$. Let $f : \mathbb{R}^{np} \rightarrow \mathbb{R}$ be a function of local variables $\{\mathbf{x}_i\}_{i \in \mathcal{V}}$ such that $f(\mathbf{x}) \triangleq \sum_{i \in \mathcal{V}} f_i(\mathbf{x}_i)$ for $\mathbf{x} \in \mathbb{R}^{np}$ and $\nabla f(\mathbf{x}) \triangleq [\nabla f_1(\mathbf{x}_1)^\top, \dots, \nabla f_n(\mathbf{x}_n)^\top]^\top \in \mathbb{R}^{np}$, where $\nabla f_i(\mathbf{x}_i) \in \mathbb{R}^p$ denotes the gradient of f_i at $\mathbf{x}_i \in \mathbb{R}^p$.

We next propose our decentralized optimization algorithm FRSD to solve the consensus optimization problem in (1).

Algorithm 1 FRSD

Input: $\mathbf{x}_i(0) \in \mathbb{R}^p, \forall i \in \mathcal{V}, \alpha, \beta > 0$ such that $\alpha\beta < 1, \bar{R} = [r_{ij}] \in \mathbb{R}^{n \times n}$ satisfies (5).

```

1:  $\mathbf{y}_i(0) \leftarrow \mathbf{0}_p, \mathbf{v}_i(0) \leftarrow \mathbf{e}_i \in \mathbb{R}^n$  for  $i \in \mathcal{V}$ 
2: for all  $k = 0, 1, \dots$  do
3:   Each  $i \in \mathcal{V}$  independently performs:
4:    $\bar{\mathbf{x}}_i(k) \leftarrow \sum_{j \in \mathcal{N}_i^{in}} r_{ij} \mathbf{x}_j(k)$ 
5:   if  $k > 0$  then
6:      $\mathbf{y}_i(k) \leftarrow \mathbf{y}_i(k-1) + \beta (\mathbf{x}_i(k) - \bar{\mathbf{x}}_i(k))$ 
7:   end if
8:    $\mathbf{x}_i(k+1) \leftarrow \bar{\mathbf{x}}_i(k) - \alpha \left( \frac{\nabla f_i(\mathbf{x}_i(k))}{[\mathbf{v}_i(k)]_i} + \mathbf{y}_i(k) \right)$ 
9:    $\mathbf{v}_i(k+1) \leftarrow \sum_{j \in \mathcal{N}_i^{in}} r_{ij} \mathbf{v}_j(k)$ 

```

10: **end for**

2.1 FRSD Algorithm

Consider FRSD displayed in Algorithm 1, at each iteration $k \geq 0$, every agent $i \in \mathcal{V}$ updates three variables $\mathbf{x}_i(k)$, $\mathbf{y}_i(k) \in \mathbb{R}^p$ and $\mathbf{v}_i(k) \in \mathbb{R}^n$, where $\alpha, \beta > 0$ and $\bar{R} = [r_{ij}] \in \mathbb{R}^{n \times n}$ are the parameters of the algorithm: α is the constant step-size and β is a momentum parameter such that $\alpha\beta < 1$, and \bar{R} is a row-stochastic matrix such that

$$r_{ij} = \begin{cases} > 0, & j \in \mathcal{N}_i^{in}, \\ 0, & \text{otherwise;} \end{cases} \quad \sum_{j \in \mathcal{V}} r_{ij} = 1, \quad \forall i \in \mathcal{V}. \quad (5)$$

Remark 2. Since \mathcal{G} is strongly connected and has finitely many nodes, the Markov chain corresponding to the transition probability matrix \bar{R} is irreducible and positive recurrent; moreover, since \bar{R} has a positive diagonal, it is also aperiodic; therefore, there exists a stationary distribution $\boldsymbol{\pi} \in \mathbb{R}^n$, i.e., $\boldsymbol{\pi} > 0$ and $\mathbf{1}_n^\top \boldsymbol{\pi} = 1$ such that $\boldsymbol{\pi}^\top \bar{R} = \boldsymbol{\pi}^\top$.

Definition 2. Each node $i \in \mathcal{V}$, initialized with $\mathbf{v}_i(0) = \mathbf{e}_i$, generates a sequence $\{\mathbf{v}_i(k)\}_{k \geq 0}$ initialized from $\mathbf{v}_i(0) = \mathbf{e}_i$, using the recursion in *line 9* of the FRSD algorithm. Define $R = \bar{R} \otimes I_p$, $\bar{V}(k) \triangleq [\mathbf{v}_1(k), \dots, \mathbf{v}_n(k)]^\top \in \mathbb{R}^{n \times n}$, i.e., $\mathbf{v}_i(k)$ is the i -th row of $\bar{V}(k)$, set $V(k) \triangleq \bar{V}(k) \otimes I_p$ and $\tilde{V}(k) \triangleq \text{diag}(V(k))$.

Given arbitrary $\mathbf{x}(0) \in \mathbb{R}^{np}$, we initialize $\mathbf{y}(0) \in \mathbb{R}^{np}$ such that $\mathbf{y}_i(0) = \mathbf{0}_p$ for $i \in \mathcal{V}$ and $\bar{V}(0) = I_n$. We present FRSD stated in Algorithm 1 in a compact form as follows:

$$\mathbf{x}(k+1) = R\mathbf{x}(k) - \alpha \left(\mathbf{y}(k) + \tilde{V}^{-1}(k) \nabla f(\mathbf{x}(k)) \right), \quad (6a)$$

$$\mathbf{y}(k+1) = \mathbf{y}(k) + \beta (I_n - R) \mathbf{x}(k+1), \quad (6b)$$

$$\bar{V}(k+1) = \bar{R} \bar{V}(k). \quad (6c)$$

Remark 3. In the numerical section (Section 4), we also considered a corrected step variant of FRSD, called FRSD-CS, which is only different from FRSD in the step size choice, i.e., FRSD-CS is obtained by replacing (6a) with

$$\mathbf{x}(k+1) = R\mathbf{x}(k) - \alpha \tilde{V}(k) \left(\mathbf{y}(k) + \tilde{V}^{-1}(k) \nabla f(\mathbf{x}(k)) \right).$$

We empirically observed that FRSD parameter tuning is pretty stable: once the parameters are tuned, the performance of the algorithm is robust to slight changes to the problem parameters, e.g., changes in the graph topology (through adding/deleting an edge), in the number of agents (increase/decrease), in convexity modulus and Lipschitz constants. For instance, in the Huber-loss minimization and logistic regression problems we tested in the numerical section, fixing $\alpha, \beta > 0$ such that $\alpha\beta = 0.05$ worked very well; thus, tuning hyper-parameters can be treated as one-dimensional search as in Xi-row, \mathcal{AB} , Push-DIGing, which are using a single parameter $\alpha > 0$. Furthermore, if the problem in (1) will be solved repetitively with slightly changing data, then having an additional parameter might be helpful as it gives the practitioner an extra degree of freedom to optimize the performance given that one time parameter tuning would work fairly well under slight changes.

In the rest, we focus on establishing asymptotic convergence guarantees for FRSD; therefore, we skip providing a termination condition as designing a locally implementable stopping mechanism for *decentralized optimization* algorithms is itself a complicated task, attracting recent research interest [43, 44].

2.2 Related Methods

Next, we discuss the existing distributed optimization methods for a directed graph \mathcal{G} satisfying Assumption 1. In particular, Push-DIGing using column-stochastic weights, \mathcal{AB} , \mathcal{ABm} , \mathcal{ABN} and Push-Pull using both row- and column-stochastic weights, and Xi-row, FROZEN, D-DGNT using only row-stochastic weights are closely related to our FRSD method and are described below in detail.

2.2.1 Push-DIGing

The Push-DIGing algorithm, proposed in [31], achieves a linear convergence rate for solving (1) over directed graphs (possibly time-varying) with a constant step-size under Assumptions 1-3. Given \mathcal{G} , Push-DIGing updates four variables $\mathbf{x}_i(k), \mathbf{y}_i(k), \mathbf{z}_i(k) \in \mathbb{R}^p$ and $v_i(k) \in \mathbb{R}$ for each agent $i \in \mathcal{V}$ as follows:

$$\begin{aligned} v_i(k+1) &= \sum_{j \in \mathcal{N}_i^{\text{in}}} b_{ij} v_j(k), \\ \mathbf{x}_i(k+1) &= \sum_{j \in \mathcal{N}_i^{\text{in}}} b_{ij} (\mathbf{x}_j(k) - \alpha \mathbf{y}_j(k)), \\ \mathbf{z}_i(k+1) &= \mathbf{x}_i(k+1)/v_i(k+1), \\ \mathbf{y}_i(k+1) &= \sum_{j \in \mathcal{N}_i^{\text{in}}} b_{ij} \mathbf{y}_j(k) + \nabla f_i(\mathbf{z}_i(k+1)) - \nabla f_i(\mathbf{z}_i(k)), \end{aligned}$$

where $\alpha > 0$ is the step size and $\bar{B} = [b_{ij}] \in \mathbb{R}^{n \times n}$ is a column-stochastic matrix compatible with \mathcal{G} . The Push-DIGing algorithm is initialized with $v_i(0) = 1$, $\mathbf{y}_i(0) = \nabla f_i(\mathbf{z}_i(0))$ and from an arbitrary $\mathbf{x}_i(0)$ for each $i \in \mathcal{V}$. Since directed graphs are not balanced in general, Push-DIGing adopts a push-sum strategy, which utilizes column-stochastic weights, requiring each agent to know its out-degree –this may not be practical within broadcast-based communication systems. Compared to using column-stochastic weights, adopting row-stochastic weights might be preferred in such a distributed environment where each agent only manages the weights on information pertaining to its in-neighbors.

2.2.2 $\mathcal{AB}/\mathcal{ABm}/\text{Push-Pull}$

In contrast to Push-DIGing, \mathcal{AB} [33] and \mathcal{ABm} [34] algorithms could get away with the nonlinear update due to eigenvector estimation. The \mathcal{AB} and \mathcal{ABm} ³ methods use both row-stochastic and column-stochastic weights simultaneously. At each iteration $k \geq 0$, they update two variables $\mathbf{x}_i(k), \mathbf{y}_i(k) \in \mathbb{R}^p$ for each agent $i \in \mathcal{V}$:

$$\begin{aligned} \mathbf{x}_i(k+1) &= \sum_{j \in \mathcal{N}_i^{\text{in}}} r_{ij} \mathbf{x}_j(k) - \alpha \mathbf{y}_i(k) + \beta (\mathbf{x}_i(k) - \mathbf{x}_i(k-1)), \\ \mathbf{y}_i(k+1) &= \sum_{j \in \mathcal{N}_i^{\text{in}}} b_{ij} (\mathbf{y}_j(k) + \nabla f_j(\mathbf{x}_j(k+1)) - \nabla f_j(\mathbf{x}_j(k))), \end{aligned}$$

³To present \mathcal{AB} and \mathcal{ABm} in a unified manner, we use an adapt-then-combine update for \mathbf{y}_i in \mathcal{ABm} similar to \mathcal{AB} [33]. In [34], it is mentioned that the \mathcal{ABm} method also works with this update; but, the originally stated version of \mathcal{ABm} uses an combine-then-adapt update: $\mathbf{y}_i(k+1) = \sum_{j \in \mathcal{N}_i^{\text{in}}} b_{ij} \mathbf{y}_j(k) + \nabla f_j(\mathbf{x}_j(k+1)) - \nabla f_j(\mathbf{x}_j(k))$ –we also considered the original version for our numerical tests.

where $\alpha > 0$ is the step-size, $\beta \geq 0$ is the momentum parameter, $\bar{R} = [r_{ij}] \in \mathbb{R}^{n \times n}$ and $\bar{B} = [b_{ij}] \in \mathbb{R}^{n \times n}$ denote the row-stochastic and column-stochastic weights, respectively, compatible with \mathcal{G} . For the \mathcal{AB} method, the momentum parameter $\beta = 0$, and the iterate sequence, initialized with an arbitrary $\mathbf{x}_i(0)$ and $\mathbf{y}_i(0) = \nabla f_i(\mathbf{x}_i(0))$ for each $i \in \mathcal{V}$, converges with a linear rate to the optimal solution under Assumptions 1-3. Setting $\beta > 0$, \mathcal{ABm} [34] combines the gradient tracking with a momentum term and can deal with nonuniform step-sizes, i.e., each agent- i can pick α_i and β_i .

Push-Pull, proposed in [35], is related to \mathcal{AB} , it is only different in its $\mathbf{x}_i(k+1)$ update:

$$\mathbf{x}_i(k+1) = \sum_{j \in \mathcal{N}_i^{in}} r_{ij}(\mathbf{x}_j(k) - \alpha \mathbf{y}_j(k)),$$

while $\mathbf{y}_i(k+1)$ update is the same with \mathcal{AB} . \mathcal{AB} approach is based on the Combine-And-Adapt based scheme; on the other hand, Push-Pull method can be considered as an Adapt-Then-Combine based approach –for more details see [45].

2.2.3 Xi-row

The method proposed in [37], which we call it as Xi-row in this paper, can solve (1) over directed networks with a linear convergence rate using a constant step-size uniform across the agents. Similar to our FRSD method, it only employs row-stochastic weights. Each agent $i \in \mathcal{V}$ updates three variables $\mathbf{x}_i(k)$, $\mathbf{y}_i(k)$, $\mathbf{v}_i(k) \in \mathbb{R}^p$ as follows:

$$\begin{aligned} \mathbf{x}_i(k+1) &= \sum_{j \in \mathcal{N}_i^{in}} r_{ij} \mathbf{x}_j(k) - \alpha \mathbf{y}_i(k), \\ \mathbf{v}_i(k+1) &= \sum_{j \in \mathcal{N}_i^{in}} r_{ij} \mathbf{v}_j(k), \\ \mathbf{y}_i(k+1) &= \sum_{j \in \mathcal{N}_i^{in}} r_{ij} \mathbf{y}_j(k) + \frac{\nabla f_i(\mathbf{x}_i(k+1))}{[\mathbf{v}_i(k+1)]_i} - \frac{\nabla f_i(\mathbf{x}_i(k))}{[\mathbf{v}_i(k)]_i}, \end{aligned}$$

where $\alpha > 0$ is the step-size and $\bar{R} = [r_{ij}] \in \mathbb{R}^{n \times n}$ is a row-stochastic matrix compatible with \mathcal{G} . The Xi-row iterates are initialized with arbitrary $\mathbf{x}_i(0)$, $\mathbf{v}_i(0) = \mathbf{e}_i$ and $\mathbf{y}_i(0) = \nabla f_i(\mathbf{x}_i(0))$ for each $i \in \mathcal{V}$. A variant of the Xi-row method, FROST [38] extends Xi-row to handle nonuniform step-sizes.

2.2.4 \mathcal{ABN} /FROZEN/D-DNGT

\mathcal{ABN} and FROZEN, proposed in [36], extend \mathcal{AB} and Xi-row to incorporate Nesterov's momentum term. Similar to \mathcal{AB} , \mathcal{ABN} uses both row-stochastic and column-stochastic weights, while FROZEN only requires row-stochastic weights. At each iteration $k \geq 0$, \mathcal{ABN} updates three variables $\mathbf{x}_i(k)$, $\mathbf{y}_i(k)$, $\mathbf{s}_i(k) \in \mathbb{R}^p$ for each agent $i \in \mathcal{V}$:

$$\mathbf{s}_i(k+1) = \sum_{j \in \mathcal{N}_i^{in}} r_{ij} \mathbf{x}_j(k) - \alpha \mathbf{y}_i(k) \tag{7a}$$

$$\mathbf{x}_i(k+1) = \mathbf{s}_i(k+1) + \beta (\mathbf{s}_i(k+1) - \mathbf{s}_i(k)), \tag{7b}$$

$$\mathbf{y}_i(k+1) = \sum_{j \in \mathcal{N}_i^{in}} b_{ij} \mathbf{y}_j(k) + \nabla f_i(\mathbf{x}_i(k+1)) - \nabla f_i(\mathbf{x}_i(k)),$$

and FROZEN updates $\mathbf{x}_i(k)$, $\mathbf{y}_i(k)$, $\mathbf{s}_i(k) \in \mathbb{R}^p$, and $\mathbf{v}_i(k) \in \mathbb{R}^n$, such that for each agent $i \in \mathcal{V}$, $\mathbf{s}_i(k+1)$ and $\mathbf{x}_i(k+1)$ are updated according to (7a) and (7b), respectively, $\mathbf{v}_i(k+1)$ and $\mathbf{y}_i(k+1)$ are updated according to

$$\begin{aligned} \mathbf{v}_i(k+1) &= \sum_{j \in \mathcal{N}_i^{in}} r_{ij} \mathbf{v}_j(k), \\ \mathbf{y}_i(k+1) &= \sum_{j \in \mathcal{N}_i^{in}} r_{ij} \mathbf{y}_j(k) + \frac{\nabla f_i(\mathbf{x}_i(k+1))}{[\mathbf{v}_i(k+1)]_i} - \frac{\nabla f_i(\mathbf{x}_i(k))}{[\mathbf{v}_i(k)]_i}, \end{aligned}$$

where $\alpha > 0$ is the step-size, $\beta \geq 0$ is the momentum parameter in both methods, $\bar{R} = [r_{ij}] \in \mathbb{R}^{n \times n}$ and $\bar{B} = [b_{ij}] \in \mathbb{R}^{n \times n}$ denote row-stochastic and column-stochastic weight matrices. For both methods, $\mathbf{x}_i(0)$ and $\mathbf{s}_i(0)$ are arbitrary, and the other variables are initialized as $\mathbf{v}_i(0) = \mathbf{e}_i$, $\mathbf{y}_i(0) = \nabla f_i(\mathbf{x}_i(0))$ for each $i \in \mathcal{V}$.

Another momentum-based method is D-DNGT, proposed in [39]. D-DNGT is related to both FROZEN and \mathcal{ABm} :

$$\mathbf{s}_i(k+1) = \sum_{j \in \mathcal{N}_i^{in}} r_{ij} \mathbf{x}_j(k) + \beta (\mathbf{s}_i(k) - \mathbf{s}_i(k-1)) - \alpha \mathbf{y}_i(k)$$

$$\begin{aligned}
\mathbf{x}_i(k+1) &= \mathbf{s}_i(k+1) + \beta (\mathbf{s}_i(k+1) - \mathbf{s}_i(k)), \\
\mathbf{v}_i(k+1) &= \sum_{j \in \mathcal{N}_i^{\text{in}}} r_{ij} \mathbf{v}_j(k), \\
\mathbf{y}_i(k+1) &= \sum_{j \in \mathcal{N}_i^{\text{in}}} r_{ij} \mathbf{y}_j(k) + \frac{\nabla f_i(\mathbf{x}_i(k+1))}{[\mathbf{v}_i(k+1)]_i} - \frac{\nabla f_i(\mathbf{x}_i(k))}{[\mathbf{v}_i(k)]_i},
\end{aligned}$$

where the initial variables are set as in the FROZEN method.

2.2.5 Comparison of different dynamics

Relaxing the assumption that all nodes need to know their out-degree (a requirement for column-stochastic methods) comes at a cost. Indeed, to be able to implement FRSD or any other row-stochastic method, e.g., Xi-row, FROZEN, D-DNGT, one needs each agent $i \in \mathcal{V}$ to know the total number of agents in the network as well as its own rank in order to construct $\mathbf{v}_i \in \mathbb{R}^n$. Finally, while push-sum based methods only require an extra scalar to be stored for “debiasing,” row-stochastic methods, including FRSD, require storing an n -dimensional vector, which grows linearly with the number of agents in the network.

Next, to get a better insight about the FRSD update rule, we write $\mathbf{x}(k+2)$ in a recursive manner for FRSD and compare it against \mathcal{AB} and \mathcal{ABm} methods, which use both row- and column-stochastic matrices, and also with Xi-row which uses row-stochastic weights.

$\mathcal{AB}/\mathcal{ABm}$: For $k \geq 0$,

$$\begin{aligned}
\mathbf{x}(k+2) &= (R+B)\mathbf{x}(k+1) - BR\mathbf{x}(k) - \alpha B(\nabla f(\mathbf{x}(k+1)) - \nabla f(\mathbf{x}(k))) \\
&\quad + \beta(\mathbf{x}(k+1) - \mathbf{x}(k)) - \beta B(\mathbf{x}(k) - \mathbf{x}(k-1)).
\end{aligned}$$

$\mathcal{AB}/\mathcal{ABm}$ recursion using column-stochastic weights for the gradient tracking term, $B(\nabla f(\mathbf{x}(k+1)) - \nabla f(\mathbf{x}(k)))$, is fundamentally different from the row-stochastic methods.

Xi-row: For $k \geq 0$,

$$\begin{aligned}
\mathbf{x}(k+2) &= 2R\mathbf{x}(k+1) - R^2\mathbf{x}(k) \\
&\quad - \alpha(\tilde{V}^{-1}(k+1)\nabla f(\mathbf{x}(k+1)) - \tilde{V}^{-1}(k)\nabla f(\mathbf{x}(k))),
\end{aligned}$$

where for $k \geq 0$, $\tilde{V}(k) \triangleq \text{diag}(V(k))$ and $V(k) \triangleq [\mathbf{v}_1(k), \dots, \mathbf{v}_n(k)]^\top \in \mathbb{R}^{n \times n}$ –see Definition 2. Except for the difference in how gradient tracking is handled, \mathcal{AB} and Xi-row are closely related in terms of consensus dynamics, i.e., say $R = B = W$ for some doubly-stochastic mixing matrix W compatible with \mathcal{G} , then both \mathcal{AB} ($\beta = 0$) and Xi-row updates take the same form: $\mathbf{x}(k+2) = 2W\mathbf{x}(k) - W^2\mathbf{x}(k) + G(k)$, where $G(k)$ is the term related to gradient tracking. In contrast, FRSD has different consensus dynamics.

FRSD: For $k \geq 0$,

$$\begin{aligned}
\mathbf{x}(k+2) &= ((1+\alpha\beta)R + (1-\alpha\beta)I_{np})\mathbf{x}(k+1) - R\mathbf{x}(k) \\
&\quad - \alpha(\tilde{V}^{-1}(k+1)\nabla f(\mathbf{x}(k+1)) - \tilde{V}^{-1}(k)\nabla f(\mathbf{x}(k))).
\end{aligned}$$

For FRSD, we can set $\beta = c/\alpha$ for any $c \in (0, 1)$, and using this choice, FRSD updates reduces to

$$\begin{aligned}
\mathbf{x}(k+2) &= 2R\mathbf{x}(k+1) - R^2\mathbf{x}(k) + \Delta_c(k) \\
&\quad - \alpha(\tilde{V}^{-1}(k+1)\nabla f(\mathbf{x}(k+1)) - \tilde{V}^{-1}(k)\nabla f(\mathbf{x}(k))),
\end{aligned}$$

where $\Delta_c(k) \triangleq (1-c)(I-R)\mathbf{x}(k+1) - R(I-R)\mathbf{x}(k)$ is the difference term between FRSD and the other two recursion rules. Indeed, for $R = W$ as above, FRSD recursion takes the form: $\mathbf{x}(k+2) = 2W\mathbf{x}(k) - W^2\mathbf{x}(k) + G(k) + \Delta_c(k)$, where $G(k)$ is the FRSD gradient tracking term same with Xi-row.

Clearly, for arbitrary R and B that are compatible with a non-trivial directed graph \mathcal{G} , \mathcal{AB} , \mathcal{ABm} , Xi-row and FRSD are not the same, they generate distinct iterate sequences.

Methods	Variables	Memory	Comm.	Row S.	Col. S.
\mathcal{AB}	$\mathbf{x}, \mathbf{x}^p, \mathbf{y}$	$3p$	$2p$	✓	✓
Push-Pull	$\mathbf{x}, \mathbf{x}^p, \mathbf{y}$	$3p$	$2p$	✓	✓
\mathcal{ABN}	$\mathbf{x}, \mathbf{x}^p, \mathbf{y}, \mathbf{s}$	$4p$	$2p$	✓	✓
\mathcal{ABm}	$\mathbf{x}, \mathbf{x}^p, \mathbf{y}$	$3p$	$2p$	✓	✓
Push-Ding	$\mathbf{x}, \mathbf{x}^p, \mathbf{y}, \mathbf{v}$	$3p + 1$	$2p + 1$	✗	✓
Xi-row	$\mathbf{x}, \mathbf{x}^p, \mathbf{y}, \mathbf{v}$	$3p + n$	$2p + n$	✓	✗
D-DNGT	$\mathbf{x}, \mathbf{x}^p, \mathbf{y}, \mathbf{s}, \mathbf{s}^p, \mathbf{v}$	$5p + n$	$2p + n$	✓	✗
FROZEN	$\mathbf{x}, \mathbf{x}^p, \mathbf{y}, \mathbf{s}, \mathbf{v}$	$4p + n$	$2p + n$	✓	✗
FRSD	$\mathbf{x}, \mathbf{y}, \mathbf{v}$	$2p + n$	$p + n$	✓	✗
FRSD-CS	$\mathbf{x}, \mathbf{y}, \mathbf{v}$	$2p + n$	$p + n$	✓	✗

Table 1: Comparison of methods for directed graphs in terms of storage and communication requirements (The “Variables” column lists the variables stored at each node to carry out the computation – \mathbf{x}^p denotes the previous iterate), and whether they use row- and/or column-stochastic mixing matrices.

2.2.6 Implicit Gradient Tracking

It is important to emphasize that the gradient tracking component

$$\tilde{V}^{-1}(k+1)\nabla f(\mathbf{x}(k+1)) - \tilde{V}^{-1}(k)\nabla f(\mathbf{x}(k))$$

indeed appears in the FRSD recursion when written using only in \mathbf{x} variables. That is why we are able to obtain the linear convergence for the FRSD iterate sequence. However, it is also worth mentioning that the implementation of the FRSD algorithm in practice does not require gradient tracking for computations at the node level, unlike the other methods in the literature – as all other methods *explicitly* use the gradient tracking, e.g., Xi-row, \mathcal{AB} , Push-Pull, Push-DINGing. Using an *implicit* gradient tracking mechanism, FRSD does not need to store the previous iterates for neither \mathbf{x}_i nor \mathbf{y}_i variables, i.e., each agent- i needs to store only $\mathbf{x}_i(k)$, $\mathbf{y}_i(k-1)$ and $\mathbf{v}_i(k)$ to be able to update these iterates to $\mathbf{x}_i(k+1)$, $\mathbf{y}_i(k)$ and $\mathbf{v}_i(k+1)$; hence, to implement FRSD, agent- i needs to store a $2p + n$ -dimensional vector. Moreover, the novel \mathbf{y} -update (see line 6 of Algorithm 1), leading to *implicit* gradient-tracking, also result in a significant reduction in communication overhead; indeed, in order to implement FRSD, the agent- j needs to only broadcast $\mathbf{x}_j(k) \in \mathbb{R}^p$ and $\mathbf{v}_j(k) \in \mathbb{R}^n$; thus, $j \in \mathcal{V}$ needs to only transmit $n + p$ -dimensional vector – note that for small networks, i.e., when n is small, this is a significant reduction compared to $2p + 1$ required by both Push-DINGing and \mathcal{AB} /Push Pull. Furthermore, comparing FRSD with the other row-stochastic methods Xi-row, FROZEN, and D-DNGT communication requirement decreases from $2p + n$ to $p + n$. Therefore, for solving high-dimensional problems over small-to-medium size networks, i.e., $p \gg n$, FRSD becomes the method of choice – see Table 1.

2.3 Primal-Dual Algorithm Motivation

In a similar spirit with the discussion in [46], we can argue that FRSD is closely related to the primal-dual algorithms for saddle point problems studied within the optimization literature. More precisely, consider an equivalent formulation of the main problem in (1):

$$\min_{\{\mathbf{x}_i\}_{i \in \mathcal{V}}} \left\{ \sum_{i \in \mathcal{V}} f_i(\mathbf{x}_i) : \mathbf{x} \triangleq [\mathbf{x}_i]_{i \in \mathcal{V}} \in \mathcal{C} \right\},$$

where $\mathcal{C} \triangleq \{\mathbf{x} : \mathbf{x}_1 = \mathbf{x}_2 = \dots = \mathbf{x}_n\}$. Using the Fenchel duality, this problem can be written equivalently as

$$\min_{\mathbf{x}} \max_{\mathbf{y} \in \mathcal{C}^\perp} \sum_{i \in \mathcal{V}} f_i(\mathbf{x}_i) + \mathbf{y}_i^\top \mathbf{x}_i, \quad (8)$$

where \mathcal{C}^\perp is the orthogonal complement of the subspace \mathcal{C} , i.e., $\mathbf{y} \in \mathcal{C}^\perp$ if and only if $\sum_{i \in \mathcal{V}} \mathbf{y}_i = \mathbf{0}_p$. After switching the roles of \mathbf{x} and \mathbf{y} through multiplying (8) with -1 , if one naively implements a variant⁴ of the primal-dual algorithm proposed by Chambolle & Pock [48], we get

$$\mathbf{m}_i(k) \leftarrow (1 + \theta)\mathbf{y}_i(k) - \theta\mathbf{y}_i(k-1), \quad i \in \mathcal{V}, \quad (9a)$$

$$\mathbf{x}_i(k+1) \leftarrow \mathbf{x}_i(k) - \alpha \left(\nabla f_i(\mathbf{x}_i(k)) + \mathbf{m}_i(k) \right), \quad i \in \mathcal{V}, \quad (9b)$$

$$\mathbf{y}(k+1) \leftarrow \Pi_{\mathcal{C}^\perp} \left(\mathbf{y}(k) + \beta \mathbf{x}(k+1) \right), \quad (9c)$$

where $\alpha, \beta > 0$ are primal and dual step sizes, respectively, and $\theta \geq 0$ is the momentum parameter – here, $\Pi_{\mathcal{C}^\perp}(\cdot)$ denotes the Euclidean projection onto \mathcal{C}^\perp , i.e., for simplicity of the notation, assume $p = 1$; then, for any \mathbf{y} ,

⁴The variant we discussed here is proposed in [47] – see Eq (5) therein.

$\Pi_{C^\perp}(\mathbf{y}) = \mathbf{y} - \mathbf{1}\mathbf{1}^\top \mathbf{y}/n$. This explicit form of $\Pi_{C^\perp}(\cdot)$ and $\mathbf{y}(0) = \mathbf{0}_n$ initialization imply that (9c) is equivalent to $\mathbf{y}(k+1) \leftarrow \mathbf{y}(k) + \Pi_{C^\perp}(\mathbf{x}(k+1))$. This method is known to converge with a linear rate for appropriately chosen $\alpha, \beta, \theta > 0$; however, due to $\Pi_{C^\perp}(\cdot)$ in (9c), this algorithm is not decentralized.

To motivate FRSD through an analogy, suppose the underlying network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is undirected and we are given a doubly stochastic mixing matrix $W \in \mathbb{R}^{n \times n}$ such that $W = W^\top$ and $W_{ij} > 0$ if and only if $(i, j) \in \mathcal{E}$. Let $W_\infty \triangleq \lim_{k \rightarrow \infty} W^k = \mathbf{1}_n \mathbf{1}_n^\top / n$. Note that $\Pi_{C^\perp}(\mathbf{y}) = (I - W_\infty)\mathbf{y}$. For decentralized implementation, consider approximating $\Pi_{C^\perp}(\cdot)$ with $(I - W)(\cdot)$. Thus, we will approximate the update in (9c) with $\mathbf{y}(k+1) \leftarrow \mathbf{y}(k) + (I - W)\mathbf{x}(k+1)$, and with this approximation, the recursion in (9) takes the following form:

$$\mathbf{y}(k) \leftarrow \mathbf{y}(k-1) + \beta(I - W)\mathbf{x}(k), \quad (10a)$$

$$\mathbf{x}(k+1) \leftarrow \mathbf{x}(k) - \alpha \left(\nabla f(\mathbf{x}(k)) + \mathbf{y}(k) + \theta \beta (I - W)\mathbf{x}(k) \right), \quad (10b)$$

where $\nabla f(\mathbf{x}) = [\nabla f_1(\mathbf{x}_1)^\top, \dots, \nabla f_n(\mathbf{x}_n)^\top]^\top$. Hence, adding and subtracting $W\mathbf{x}(k)$ to (10b), we get

$$\begin{aligned} \mathbf{x}(k+1) &\leftarrow W\mathbf{x}(k) - \alpha \left(\nabla f(\mathbf{x}(k)) + \mathbf{y}(k) \right) \\ &\quad + (1 - \alpha\beta\theta)(I - W)\mathbf{x}(k). \end{aligned} \quad (11)$$

Therefore, given the primal, dual step sizes $\alpha, \beta > 0$ such that $\alpha\beta < 1$, setting the momentum parameter $\theta = (\alpha\beta)^{-1} > 1$, the last term in (11) disappears as $1 - \alpha\beta\theta = 0$, the primal-dual algorithm with approximate averaging in (10) reduces to

$$\mathbf{y}(k) \leftarrow \mathbf{y}(k-1) + \beta \left(\mathbf{x}(k) - W\mathbf{x}(k) \right), \quad (12a)$$

$$\mathbf{x}(k+1) \leftarrow W\mathbf{x}(k) - \alpha \left(\nabla f(\mathbf{x}(k)) + \mathbf{y}(k) \right). \quad (12b)$$

It can be clearly seen that the FRSD algorithm for directed networks can be obtained from (12) by replacing the doubly stochastic mixing matrix W with a row stochastic R and by “debiasing” through introducing $\{\mathbf{v}(k)\}_k \subset \mathbb{R}^n$ sequence since $R_\infty = \lim_{k \rightarrow \infty} R^k = \mathbf{1}_n \pi^\top$ for some $\pi \in \mathbb{R}^n$ such that $\pi > \mathbf{0}_n$ and $\mathbf{1}_n^\top \pi = 1$.

3 Main Results

In this section, we will show that the iterate sequence generated by the algorithm FRSD as stated in (6) converges to the optimal solution \mathbf{x}^* with a linear rate. This result only applies to the FRSD method, theoretical analysis of FRSD-CS is not considered in this paper.

Remark 4. Assumptions 2 and 3 imply that f is L -smooth, i.e., $\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{x}')\| \leq L \|\mathbf{x} - \mathbf{x}'\|$, and μ -strongly convex.

Remark 5. Since \bar{R} is row-stochastic, the spectral radius of \bar{R} is 1, $\rho(\bar{R}) = 1$; thus, $\lim_{k \rightarrow \infty} \bar{R}^k$ exists. In particular, since \bar{R} corresponds to an ergodic Markov chain, we get $\lim_{k \rightarrow \infty} \bar{R}^k = \mathbf{1}_n \pi^\top$ – see Remark 2.

Definition 3. Define $V_\infty \triangleq \lim_{k \rightarrow \infty} V(k)$ and $\tilde{V}_\infty \triangleq \text{diag}(V_\infty)$. Furthermore, let $v \triangleq \sup_{k \geq 0} \|V(k)\|$ and $\tilde{v} \triangleq$

$$\sup_{k \geq 0} \|\tilde{V}^{-1}(k)\|.$$

Remark 6. Since $\bar{V}(0) = I_n$, $\lim_{k \rightarrow \infty} \bar{R}^k = \mathbf{1}_n \pi^\top$, we get $V_\infty = (\mathbf{1}_n \pi^\top) \otimes I_p$ and $\tilde{V}_\infty = \text{diag}(V_\infty) = \pi \otimes \mathbf{1}_p$. Note $\{V(k)\}_k$ is convergent; hence, it is bounded, implying that $v \in \mathbb{R}$ exists. Furthermore, Remark 2 shows that $\pi > 0$; therefore, $\tilde{v} \in \mathbb{R}$ also exists.

Remark 7. Since \bar{R} corresponds to an Ergodic Markov chain, Remarks 2 and 5 imply that $V_\infty R = R V_\infty = V_\infty V_\infty = V_\infty$. Moreover, the spectral radius of $R - V_\infty$ satisfies $\rho(R - V_\infty) = \rho(\bar{R} - \mathbf{1}_n \pi^\top) < 1$.

Next, we define some auxiliary sequences that will be used in the analysis. For $k \geq 0$, let $\hat{\mathbf{x}}(k) \triangleq V_\infty \mathbf{x}(k) = (\mathbf{1}_n \otimes I_p)(\pi^\top \otimes I_p)\mathbf{x}(k) = (\mathbf{1}_n \otimes I_p)\hat{\mathbf{x}}(k) \in \mathbb{R}^{np}$, where $\hat{\mathbf{x}}(k) \triangleq (\pi^\top \otimes I_p)\mathbf{x}(k) \in \mathbb{R}^p$, i.e., $\hat{\mathbf{x}}(k) = \mathbf{1}_n \otimes \hat{\mathbf{x}}(k)$. Let $\mathbf{x}^* \triangleq \mathbf{1}_n \otimes \mathbf{x}^*$ where $\mathbf{x}^* \in \mathbb{R}^p$ is the unique optimal solution to (1). Thus, Definition 1 implies that $\nabla f(\hat{\mathbf{x}}(k)) = [\nabla f_1(\hat{\mathbf{x}}(k))^\top, \dots, \nabla f_n(\hat{\mathbf{x}}(k))^\top]^\top \in \mathbb{R}^{np}$ and $\nabla f(\mathbf{x}^*) = [\nabla f_1(\mathbf{x}^*)^\top, \dots, \nabla f_n(\mathbf{x}^*)^\top]^\top \in \mathbb{R}^{np}$.

Remark 8. From the optimality condition for (1), $(\mathbf{1}_n^\top \otimes I_p)\nabla f(\mathbf{x}^*) = 0$.

The structure of our proof was inspired by [37] and [49]. In particular, we construct a linear system of inequalities and use the deterministic version of the celebrated supermartingale convergence theorem [50] to prove the convergence results. We were able to show that FRSD iterates converge to the optimal consensus solution with a linear rate as in [33, 35, 39].

In the rest of this section, we establish the linear convergence; but, first, we state some preliminary results which will be used later.

Definition 4. Given $\alpha, \beta > 0$ such that $\alpha\beta \in (0, 1)$, let $C = \bar{C} \otimes I_p$ and $\bar{C} \triangleq (1 - \alpha\beta)I_n + \alpha\beta\bar{R}$, where $\bar{R} = [r_{ij}] \in \mathbb{R}^{n \times n}$ is the row-stochastic matrix as given in (5).

The Markov chain associated with \bar{C} is the lazy version of the Markov chain corresponding to \bar{R} ; thus, it has the same stationary distribution, i.e., $\lim_{k \rightarrow \infty} \bar{C}^k = \lim_{k \rightarrow \infty} \bar{R}^k = \mathbf{1}_n \pi^\top$. Next, we state two technical results that will help us derive our main result.

Lemma 1. Given R and C as defined above, there exist vector norms $\|\cdot\|_R, \|\cdot\|_C$ such that $\|\cdot\| \leq \|\cdot\|_R$ and $\|\cdot\| \leq \|\cdot\|_C$, and there exist constants $\sigma_R, \sigma_C \in (0, 1)$ such that

$$\|R\mathbf{x} - \hat{\mathbf{x}}\|_R \leq \sigma_R \|\mathbf{x} - \hat{\mathbf{x}}\|_R, \quad (13)$$

$$\|C\mathbf{x} - \hat{\mathbf{x}}\|_C \leq \sigma_C \|\mathbf{x} - \hat{\mathbf{x}}\|_C, \quad (14)$$

for any $\mathbf{x} \in \mathbb{R}^n$ and $\hat{\mathbf{x}} = V_\infty \mathbf{x}$.

Lemma 1 directly follows from (5) and Assumption 1 – for the proof of (13), see [37, Lemma 2], and (14) can be shown similarly since $\lim_{k \rightarrow \infty} C^k = \lim_{k \rightarrow \infty} R^k = (\mathbf{1}_n \pi^\top) \otimes I_p$. Indeed, since $\rho(R - V_\infty) < 1$ – see Remark 7, [51, Lemma 5.6.10] implies that there exists invertible $S \in \mathbb{R}^{np \times np}$ such that $\|\mathbf{x}\|_R \triangleq \|S\mathbf{x}\|_1$; moreover, the matrix norm $\|\cdot\|$ induced by $\|\cdot\|_R$ satisfies $\|R - V_\infty\| \in (0, 1)$. Finally, through properly scaling $\|\cdot\|_R$, we immediately get $\|\cdot\| \leq \|\cdot\|_R$, which does not affect $\|\cdot\|$ since $\|B\| = \max\{\|B\mathbf{x}\|_R / \|\mathbf{x}\|_R : \mathbf{x} \neq \mathbf{0}\}$ for any $B \in \mathbb{R}^{np \times np}$. Same arguments can be used for showing (14) as we also have $\rho(C - V_\infty) < 1$.

Remark 9. Let $\|\cdot\|$ represent the matrix norm induced by $\|\cdot\|_R$. According to [51, Lemma 5.6.10], the constant $\sigma_R \in (0, 1)$ in Lemma 1 has an explicit form, $\sigma_R = \|R - V_\infty\|$.

First, we remark that all vector norms on a finite dimensional vector spaces are equivalent, i.e., there exist $\kappa_1, \kappa_2, \kappa_3, \kappa_4 > 0$ such that

$$\begin{aligned} \|\cdot\|_R &\leq \kappa_1 \|\cdot\|_C, & \|\cdot\|_C &\leq \kappa_2 \|\cdot\|_R, \\ \|\cdot\|_R &\leq \kappa_3 \|\cdot\|, & \|\cdot\|_C &\leq \kappa_4 \|\cdot\|. \end{aligned} \quad (15)$$

Similar to the results in [29], we also have $\|V(k) - V_\infty\| \leq \Lambda \lambda^k$ for some $0 < \Lambda \in \mathbb{R}$ and $\lambda \in (0, 1)$. Below we analyze the dependence of λ and Λ on R .

Lemma 2. Let $V(k) = R^k$ for $k \geq 0$ and $V_\infty = \lim_{k \rightarrow \infty} R^k$. Then, for $\kappa_3 > 0$ defined in (15) and $\sigma_R \in (0, 1)$ given in Remark 9, the following bound holds:

$$\|\bar{V}(k) - \mathbf{1}_n \pi^\top\| = \|V(k) - V_\infty\| \leq \kappa_3 \sigma_R^k, \quad \forall k \geq 0. \quad (16)$$

Proof. It immediately follows from Remark 7 that

$$\|V(k) - V_\infty\| \leq \|(R - V_\infty)^k\| \leq \kappa_3 \|(R - V_\infty)^k\| \leq \kappa_3 \sigma_R^k,$$

holds for $k \geq 1$, where the second inequality follows from

$$\|A\| = \max_{\|\mathbf{v}\| \leq 1} \|A\mathbf{v}\| \leq \max_{\|\mathbf{v}\|_R \leq \kappa_3} \|A\mathbf{v}\|_R = \kappa_3 \|A\|,$$

for all $A \in \mathbb{R}^{np \times np}$ and the third inequality is due to $\|\cdot\|$ being submultiplicative as it is an induced norm. Finally, the equality in (16) follows from the fact that singular values of $\bar{V}(k) - \mathbf{1}_n \pi^\top$ and $V(k) - V_\infty$ are the same. \square \square

Lemma 3. The following inequalities hold for all $k \geq 0$:

$$\|\tilde{V}^{-1}(k) - \tilde{V}_\infty^{-1}\| \leq \tilde{V}^2 \sqrt{n} \kappa_3 \sigma_R^k \quad (17a)$$

$$\|\tilde{V}^{-1}(k) - \tilde{V}^{-1}(k-1)\| \leq 2\tilde{v}^2 \sqrt{n} \kappa_3 \sigma_R^k. \quad (17b)$$

Proof. The proof follows from [37, Lemma 3]. Indeed, note that $\tilde{V}^{-1}(k) - \tilde{V}_\infty^{-1} = \tilde{V}^{-1}(k)(\tilde{V}(k) - \tilde{V}_\infty)\tilde{V}_\infty^{-1}$; hence, $\|\tilde{V}^{-1}(k) - \tilde{V}_\infty^{-1}\| \leq \tilde{v}^2 \|\mathbf{diag}(V(k) - V_\infty)\| \leq \tilde{v}^2 \sqrt{n} \|\bar{V}(k) - \mathbf{1}_n \pi^\top\|$, where we used $\|A\|_F \leq \sqrt{n} \|A\|_2$ for any $A \in \mathbb{R}^{n \times n}$. Thus, the result follows from Lemma 2. \square \square

Lemma 4. *The following inequality holds for all $k \geq 0$:*

- (a) $\|V_\infty \tilde{V}^{-1}(k) \nabla f(\mathbf{x}(k))\| \leq v\tilde{v}^2 \sqrt{n\kappa_3\sigma_R^k} \|\nabla f(\mathbf{x}(k))\| + nL\|\mathbf{x}(k) - \hat{\mathbf{x}}(k)\|_C + nL\|\hat{\mathbf{x}}(k) - \mathbf{x}^*\|$
- (b) $\|V_\infty \tilde{V}^{-1}(k-1) \nabla f(\mathbf{x}(k))\| \leq 3v\tilde{v}^2 \sqrt{n\kappa_3\sigma_R^k} \|\nabla f(\mathbf{x}(k))\| + nL\|\mathbf{x}(k) - \hat{\mathbf{x}}(k)\|_C + nL\|\hat{\mathbf{x}}(k) - \mathbf{x}^*\|$
- (c) $\|\hat{\mathbf{x}}(k) - \hat{\mathbf{x}}(k-1)\| \leq \alpha v\tilde{v}L \|\mathbf{x}(k) - \mathbf{x}(k-1)\|_R + \alpha 3v\tilde{v}^2 \sqrt{n\kappa_3\sigma_R^k} \|\nabla f(\mathbf{x}(k))\|$
 $+ \alpha nL\|\mathbf{x}(k) - \hat{\mathbf{x}}(k)\|_C + \alpha nL\|\hat{\mathbf{x}}(k) - \mathbf{x}^*\|$
- (d) $\|\tilde{V}^{-1}(k) \nabla f(\mathbf{x}(k)) - \tilde{V}^{-1}(k-1) \nabla f(\mathbf{x}(k-1))\| \leq \tilde{v}L\|\mathbf{x}(k) - \mathbf{x}(k-1)\|_R + 2\tilde{v}^2 \sqrt{n\kappa_3\sigma_R^k} \|\nabla f(\mathbf{x}(k))\|.$

Proof. First, we prove the part (a).

$$\begin{aligned}
& \|V_\infty \tilde{V}^{-1}(k) \nabla f(\mathbf{x}(k))\| \\
& \leq \|V_\infty \tilde{V}^{-1}(k) \nabla f(\mathbf{x}(k)) - V_\infty \tilde{V}_\infty^{-1} \nabla f(\mathbf{x}(k))\| + \|V_\infty \tilde{V}_\infty^{-1} \nabla f(\mathbf{x}(k))\| \\
& \leq \|V_\infty\| \|\tilde{V}^{-1}(k) - \tilde{V}_\infty^{-1}\| \|\nabla f(\mathbf{x}(k))\| + \|V_\infty \tilde{V}_\infty^{-1} \nabla f(\mathbf{x}(k)) - (\mathbf{1}_n \otimes I_p)(\mathbf{1}_n^\top \otimes I_p) \nabla f(\mathbf{x}^*)\| \\
& \leq v\tilde{v}^2 \sqrt{n\kappa_3\sigma_R^k} \|\nabla f(\mathbf{x}(k))\| + nL\|\mathbf{x}(k) - \mathbf{x}^*\|,
\end{aligned}$$

which together with triangular inequality implies (a), where $\tilde{V}_\infty = \text{diag}(V_\infty)$ –see Definition 3. In the second inequality, we use Remark 8, and the third inequality follows from (17a) in Lemma 3 and we also use Remark 4 along with $V_\infty \tilde{V}_\infty^{-1} = (\mathbf{1}_n \otimes I_p)(\mathbf{1}_n^\top \otimes I_p) = (\mathbf{1}_n \mathbf{1}_n^\top) \otimes I_p$; hence, $\|(\mathbf{1}_n \mathbf{1}_n^\top) \otimes I_p\| = n$. Next, we prove part (b):

$$\begin{aligned}
& \|V_\infty \tilde{V}^{-1}(k-1) \nabla f(\mathbf{x}(k))\| \\
& \leq \|V_\infty \tilde{V}^{-1}(k-1) \nabla f(\mathbf{x}(k)) - V_\infty \tilde{V}^{-1}(k) \nabla f(\mathbf{x}(k))\| + \|V_\infty \tilde{V}^{-1}(k) \nabla f(\mathbf{x}(k))\| \\
& \leq \|V_\infty\| \|\tilde{V}^{-1}(k) - \tilde{V}^{-1}(k-1)\| \|\nabla f(\mathbf{x}(k))\| + \|V_\infty \tilde{V}^{-1}(k) \nabla f(\mathbf{x}(k))\|;
\end{aligned}$$

hence, the part (b) follows from (17b) in Lemma 3 and from part (a) of Lemma 4.

Now we consider part (c). Since $\mathbf{y}(0) = \mathbf{0}_{np}$, it follows from (6b) that $\mathbf{y}(k) = \beta(I_{np} - R) \sum_{\ell=1}^k \mathbf{x}(\ell)$. Since $V_\infty R = V_\infty - \text{see Remark 7, we have } V_\infty \mathbf{y}(k) = \mathbf{0}_{np} \text{ for all } k \geq 0 \text{ as } V_\infty(I_{np} - R) = \mathbf{0}_{np \times np}$. Hence, using } $\hat{\mathbf{x}}(k) = V_\infty \mathbf{x}(k)$ for $k \geq 0$, when we multiply V_∞ on both side of (6a), we get

$$\hat{\mathbf{x}}(k) = \hat{\mathbf{x}}(k-1) - \alpha V_\infty \tilde{V}^{-1}(k-1) \nabla f(\mathbf{x}(k-1)).$$

Therefore, the part (c) immediately follows from using Remark 4 and the part (b) of Lemma 4 on

$$\begin{aligned}
& \|\hat{\mathbf{x}}(k) - \hat{\mathbf{x}}(k-1)\| \\
& \leq \alpha \|V_\infty \tilde{V}^{-1}(k-1) (\nabla f(\mathbf{x}(k-1)) - \nabla f(\mathbf{x}(k)))\| + \alpha \|V_\infty \tilde{V}^{-1}(k-1) \nabla f(\mathbf{x}(k))\|.
\end{aligned}$$

Finally, we consider the part (d).

$$\begin{aligned}
& \|\tilde{V}^{-1}(k) \nabla f(\mathbf{x}(k)) - \tilde{V}^{-1}(k-1) \nabla f(\mathbf{x}(k-1))\| \\
& \leq \|\tilde{V}^{-1}(k-1) \nabla f(\mathbf{x}(k)) - \tilde{V}^{-1}(k-1) \nabla f(\mathbf{x}(k-1))\| + \|\tilde{V}^{-1}(k) \nabla f(\mathbf{x}(k)) - \tilde{V}^{-1}(k-1) \nabla f(\mathbf{x}(k))\| \\
& \leq \|\tilde{V}^{-1}(k-1)\| \|\nabla f(\mathbf{x}(k)) - \nabla f(\mathbf{x}(k-1))\| + \|\tilde{V}^{-1}(k) - \tilde{V}^{-1}(k-1)\| \|\nabla f(\mathbf{x}(k))\|.
\end{aligned}$$

Hence, the part (d) follows from (17b) of Lemma 3 and Remark 4. \square

For the sake of completeness we provide another technical result –for its proof, see [49, Lemma 10].

Lemma 5. *Given $\alpha \in (0, \frac{2}{nL})$, let $\eta \triangleq \max\{|1 - nL\alpha|, |1 - n\mu\alpha|\}$. If Assumptions 2 and 3 hold, then for all $\mathbf{x} \in \mathbb{R}^p$, one has*

$$\|\mathbf{x} - \alpha \sum_{i \in \mathcal{V}} \nabla f_i(\mathbf{x}) - \mathbf{x}^*\| \leq \eta \|\mathbf{x} - \mathbf{x}^*\|.$$

Next, we will obtain bounds on $\|\mathbf{x}(k+1) - \hat{\mathbf{x}}(k+1)\|_C$, $\|\hat{\mathbf{x}}(k+1) - \mathbf{x}^*\|$ and $\|\mathbf{x}(k+1) - \mathbf{x}(k)\|_R$. Combining these results will help us establish the linear rate for FRSD.

Lemma 6. *The following inequality holds for all $k \geq 0$:*

$$\begin{aligned} & \|\mathbf{x}(k+1) - \hat{\mathbf{x}}(k+1)\|_C \\ & \leq (\sigma_C + \alpha\kappa_4 nL) \|\mathbf{x}(k) - \hat{\mathbf{x}}(k)\|_C + (\kappa_2 \|R\| + \alpha\kappa_4 \tilde{v}L) \|\mathbf{x}(k) - \mathbf{x}(k-1)\|_R + \alpha\kappa_4 nL \|\hat{\mathbf{x}}(k) - \mathbf{x}^*\| \\ & \quad + \alpha\kappa_4 (2+v) \tilde{v}^2 \sqrt{n\kappa_3} \sigma_R^k \|\nabla f(\mathbf{x}(k))\|, \end{aligned}$$

where $\|\cdot\|$ denotes the induced matrix norm corresponding to the vector norm $\|\cdot\|_R$.

Proof. Using (6a) twice, one for $\mathbf{x}(k+1)$ and one for $\hat{\mathbf{x}}(k+1) = V_\infty \mathbf{x}(k+1)$, and using $V_\infty R = V_\infty$ together with $V_\infty \mathbf{y}(k) = 0$, we get the first equality below:

$$\begin{aligned} & \|\mathbf{x}(k+1) - \hat{\mathbf{x}}(k+1)\|_C \\ & = \|R\mathbf{x}(k) - \alpha\mathbf{y}(k) - \alpha\tilde{V}^{-1}(k)\nabla f(\mathbf{x}(k)) - \hat{\mathbf{x}}(k) + \alpha V_\infty \tilde{V}^{-1}(k)\nabla f(\mathbf{x}(k))\|_C \\ & = \|R\mathbf{x}(k) - R\mathbf{x}(k-1) + \mathbf{x}(k) - \hat{\mathbf{x}}(k) - \alpha\beta(I_n - R)\mathbf{x}(k) + \alpha\tilde{V}^{-1}(k-1)\nabla f(\mathbf{x}(k-1)) \\ & \quad - \alpha\tilde{V}^{-1}(k)\nabla f(\mathbf{x}(k)) + \alpha V_\infty \tilde{V}^{-1}(k)\nabla f(\mathbf{x}(k))\|_C \\ & \leq \|((1-\alpha\beta)I_n + \alpha\beta R)\mathbf{x}(k) - \hat{\mathbf{x}}(k)\|_C + \kappa_2 \|R\mathbf{x}(k) - R\mathbf{x}(k-1)\|_R \\ & \quad + \alpha\kappa_4 \|\tilde{V}^{-1}(k)\nabla f(\mathbf{x}(k)) - \tilde{V}^{-1}(k-1)\nabla f(\mathbf{x}(k-1))\| + \alpha\kappa_4 \|V_\infty \tilde{V}^{-1}(k)\nabla f(\mathbf{x}(k))\|, \end{aligned} \quad (18)$$

where in the second equality we first use (6b) to represent $\mathbf{y}(k)$ in terms of $\mathbf{x}(k)$ and $\mathbf{y}(k-1)$, and next we use (6a) to get rid of the term $-\alpha\mathbf{y}(k-1)$.

Next, using (14) of Lemma 1, we can bound the first term on the right-hand-side of (18) as follows:

$$\|((1-\alpha\beta)I_n + \alpha\beta R)\mathbf{x}(k) - \hat{\mathbf{x}}(k)\|_C = \|C\mathbf{x}(k) - \hat{\mathbf{x}}(k)\|_C \leq \sigma_C \|\mathbf{x}(k) - \hat{\mathbf{x}}(k)\|_C,$$

where C is given in Definition 4. Clearly, we can also bound the second term in (18) with $\|R\| \|\mathbf{x}(k) - \mathbf{x}(k-1)\|_R$. Finally, using the parts (d) and (a) of Lemma 4 for the third and the fourth terms, respectively, we get the desired result. \square

Remark 10. For FRSD the step size $\alpha = \mathcal{O}(\frac{1}{nL})$, i.e., there exists n_0 and $C > 0$ such that $\alpha \leq C/(nL)$ for all $n \geq n_0$, this bound is comparable to the step size bounds used in other related works, e.g., the AB, Push-DIGing, Xi-row methods.

Lemma 7. When $0 < \alpha < \frac{2}{nL}$, it holds that for $k \geq 0$:

$$\begin{aligned} & \|\hat{\mathbf{x}}(k+1) - \mathbf{x}^*\| \\ & \leq \eta \|\hat{\mathbf{x}}(k) - \mathbf{x}^*\| + \alpha nL \|\mathbf{x}(k) - \hat{\mathbf{x}}(k)\|_C + \alpha v \tilde{v}^2 \sqrt{n\kappa_3} \sigma_R^k \|\nabla f(\mathbf{x}(k))\|. \end{aligned}$$

Proof. Using (6a) for $\hat{\mathbf{x}}(k+1) = V_\infty \mathbf{x}(k+1)$ together with $V_\infty R = V_\infty$ and $V_\infty \mathbf{y}(k) = 0$, we get

$$\begin{aligned} & \|\hat{\mathbf{x}}(k+1) - \mathbf{x}^*\| \\ & = \|\hat{\mathbf{x}}(k) - \alpha V_\infty \tilde{V}^{-1}(k)\nabla f(\mathbf{x}(k)) - \mathbf{x}^*\| \\ & \leq \alpha \|((\mathbf{1}_n \mathbf{1}_n^\top) \otimes I_p) \nabla f(\hat{\mathbf{x}}(k)) - V_\infty \tilde{V}^{-1}(k)\nabla f(\mathbf{x}(k))\| + \|\hat{\mathbf{x}}(k) - \alpha((\mathbf{1}_n \mathbf{1}_n^\top) \otimes I_p) \nabla f(\hat{\mathbf{x}}(k)) - \mathbf{x}^*\|. \end{aligned} \quad (19)$$

Now, we bound the first term on the right-hand-side of (19):

$$\begin{aligned} & \|((\mathbf{1}_n \mathbf{1}_n^\top) \otimes I_p) \nabla f(\hat{\mathbf{x}}(k)) - V_\infty \tilde{V}^{-1}(k)\nabla f(\mathbf{x}(k))\| \\ & \leq \|((\mathbf{1}_n \mathbf{1}_n^\top) \otimes I_p) \nabla f(\hat{\mathbf{x}}(k)) - V_\infty \tilde{V}^{-1}(k)\nabla f(\mathbf{x}(k))\| + \|V_\infty \tilde{V}^{-1}(k)\nabla f(\mathbf{x}(k)) - V_\infty \tilde{V}_\infty^{-1} \nabla f(\mathbf{x}(k))\| \\ & \leq nL \|\mathbf{x}(k) - \hat{\mathbf{x}}(k)\|_C + v \tilde{v}^2 \sqrt{n\kappa_3} \sigma_R^k \|\nabla f(\mathbf{x}(k))\|, \end{aligned} \quad (20)$$

where we used $V_\infty \tilde{V}_\infty^{-1} = (\mathbf{1}_n \mathbf{1}_n^\top) \otimes I_p$, Assumption 2 and Lemma 3. Next, the second term on the right-hand-side of (19) can be bounded using Lemma 5:

$$\begin{aligned} \|\hat{\mathbf{x}}(k) - \alpha((\mathbf{1}_n \mathbf{1}_n^\top) \otimes I_p) \nabla f(\hat{\mathbf{x}}(k)) - \mathbf{x}^*\| & = \|\mathbf{1}_n \otimes (\hat{\mathbf{x}}(k) - \mathbf{x}^* - \alpha \sum_{i=1}^n \nabla f_i(\hat{\mathbf{x}}(k)))\| \\ & \leq \eta \|\mathbf{1}_n \otimes (\hat{\mathbf{x}}(k) - \mathbf{x}^*)\| = \eta \|\hat{\mathbf{x}}(k) - \mathbf{x}^*\|, \end{aligned} \quad (21)$$

where $\eta = \max\{|1 - nL\alpha|, |1 - n\mu\alpha|\}$, $\mathbf{x}^* = \mathbf{1}_n \otimes \mathbf{x}^*$, and $\hat{\mathbf{x}}(k) \triangleq (\pi^\top \otimes I_p)\mathbf{x}(k)$, i.e., $\hat{\mathbf{x}}(k) = \mathbf{1}_n \otimes \hat{\mathbf{x}}(k)$. Finally, Lemma 7 follows from (19)-(21). \square

Lemma 8. *The following inequality holds for all $k \geq 0$:*

$$\begin{aligned} & \|\mathbf{x}(k+1) - \mathbf{x}(k)\|_R \\ & \leq (\sigma_R + \alpha(1+v)\kappa_3\tilde{v}L)\|\mathbf{x}(k+1) - \mathbf{x}(k)\|_R + \alpha(\beta\kappa_1\|I_n - R\| + \kappa_3nL)\|\mathbf{x}(k) - \hat{\mathbf{x}}(k)\|_C \\ & \quad + \alpha\kappa_3nL\|\hat{\mathbf{x}}(k) - \mathbf{x}^*\| + \kappa_3^2\alpha(3v+2)\tilde{v}^2\sqrt{n}\sigma_R^k\|\nabla f(\mathbf{x}(k))\|. \end{aligned}$$

Proof. We use (6a) and (6b) for rewriting $\mathbf{x}(k+1)$ and $\mathbf{y}(k)$ respectively, to derive the first two equations:

$$\begin{aligned} & \|\mathbf{x}(k+1) - \mathbf{x}(k)\|_R \tag{22} \\ & = \|R\mathbf{x}(k) - \alpha\mathbf{y}(k) - \alpha\tilde{V}^{-1}(k)\nabla f(\mathbf{x}(k)) - \mathbf{x}(k)\|_R \\ & = \|R\mathbf{x}(k) - \alpha\mathbf{y}(k-1) - \alpha\beta(I_n - R)\mathbf{x}(k) - \alpha\tilde{V}^{-1}(k)\nabla f(\mathbf{x}(k)) - \mathbf{x}(k)\|_R \\ & = \|R(\mathbf{x}(k) - \mathbf{x}(k-1)) + \alpha\tilde{V}^{-1}(k-1)\nabla f(\mathbf{x}(k-1)) - \alpha\beta(I_n - R)\mathbf{x}(k) - \alpha\tilde{V}^{-1}(k)\nabla f(\mathbf{x}(k))\|_R \\ & \leq \|R\mathbf{x}(k) - R\mathbf{x}(k-1) - \hat{\mathbf{x}}(k) + \hat{\mathbf{x}}(k-1)\|_R + \kappa_3\|\hat{\mathbf{x}}(k) - \hat{\mathbf{x}}(k-1)\| + \alpha\beta\|(I_n - R)\mathbf{x}(k)\|_R \\ & \quad + \alpha\kappa_3\|\tilde{V}^{-1}(k)\nabla f(\mathbf{x}(k)) - \tilde{V}^{-1}(k-1)\nabla f(\mathbf{x}(k-1))\| \end{aligned}$$

where in the third equation, we use (6a) to get rid of the term $-\alpha\mathbf{y}(k-1)$ as we did previously to derive (18). We bound the first term above using Remark 9, i.e.,

$$\begin{aligned} & \|R\mathbf{x}(k) - R\mathbf{x}(k-1) - \hat{\mathbf{x}}(k) + \hat{\mathbf{x}}(k-1)\|_R \\ & = \|(R - V_\infty)(\mathbf{x}(k) - \mathbf{x}(k-1))\|_R \leq \sigma_R\|\mathbf{x}(k) - \mathbf{x}(k-1)\|_R. \end{aligned} \tag{23}$$

We can use Lemma 4 (c) to bound $\|\hat{\mathbf{x}}(k) - \hat{\mathbf{x}}(k-1)\|$, and Lemma 4 (d) to bound the fourth term. Then, the remaining third term in (22) can be bounded as

$$\begin{aligned} \|(I_n - R)\mathbf{x}(k)\|_R & = \|(I_n - R)(\mathbf{x}(k) - \hat{\mathbf{x}}(k))\|_R \\ & \leq \|I_n - R\|\|\mathbf{x}(k) - \hat{\mathbf{x}}(k)\|_R \\ & \leq \kappa_1\|I_n - R\|\|\mathbf{x}(k) - \hat{\mathbf{x}}(k)\|_C, \end{aligned}$$

where the first equality follows from $(I_n - R)V_\infty = \mathbf{0}$ due to $RV_\infty = V_\infty$; hence, we can add $(I_n - R)\hat{\mathbf{x}}(k)$ to $(I_n - R)\mathbf{x}(k)$. Combining all bounds gives the desired result. \square

Combining the results of Lemmas 6, 7 and 8, we will construct a linear dynamical system prove the linear convergence of the proposed algorithm. For the sake of notational simplicity, we define some constants below:

$$\begin{aligned} s_1 & \triangleq \kappa_4nL, & s_2 & \triangleq \kappa_4\tilde{v}L, & s_3 & \triangleq nL, \\ s_4 & \triangleq \kappa_3nL, & s_5 & \triangleq \kappa_3(1+v)\tilde{v}L, & s(\beta) & \triangleq \beta\kappa_1\|I_n - R\|, \\ p_1 & \triangleq \kappa_3\kappa_4(2+v)\tilde{v}^2\sqrt{n}, & p_2 & \triangleq \kappa_3v\tilde{v}^2\sqrt{n}, & p_3 & \triangleq \kappa_3^2(3v+2)\tilde{v}^2\sqrt{n}. \end{aligned}$$

For $\alpha \in (0, \frac{2}{nL})$ and $\beta > 0$ such that $\alpha\beta < 1$, FRSD sequence $\{\mathbf{x}(k)\}_{k \geq 0}$ satisfies the following system:

$$\theta_{k+1} \leq \Upsilon_{\alpha,\beta} \theta_k + \Phi_k \Psi_k, \quad \forall k \geq 0, \tag{24}$$

where θ_k, Φ_k, Ψ_k and $\Upsilon_{\alpha,\beta} \triangleq \Upsilon_1(\alpha, \beta) + \alpha\Upsilon_2$ are defined as

$$\begin{aligned} \theta_k & = \begin{bmatrix} \|\mathbf{x}(k) - \hat{\mathbf{x}}(k)\|_C \\ \|\hat{\mathbf{x}}(k) - \mathbf{x}^*\| \\ \|\mathbf{x}(k) - \mathbf{x}(k-1)\|_R \end{bmatrix}, & \Phi_k & = \sigma_R^k \alpha \begin{bmatrix} p_1 & 0 & 0 \\ p_2 & 0 & 0 \\ p_3 & 0 & 0 \end{bmatrix}, & \Psi_k & = \begin{bmatrix} \|\nabla f(\mathbf{x}(k))\| \\ 0 \\ 0 \end{bmatrix}, \\ \Upsilon_1(\alpha, \beta) & = \begin{bmatrix} \sigma_C & 0 & \kappa_2\|R\| \\ 0 & 1 & 0 \\ \alpha s(\beta) & 0 & \sigma_R \end{bmatrix}, & \Upsilon_2 & = \begin{bmatrix} s_1 & s_1 & s_2 \\ s_3 & -n\mu & 0 \\ s_4 & s_4 & s_5 \end{bmatrix}. \end{aligned}$$

Theorem 1. *Suppose Assumptions 1-3 holds. Let $\alpha, \beta > 0$ such that $\alpha \in (0, \bar{\alpha})$ and $\alpha\beta < 1$, where $\bar{\alpha} > 0$ is defined as*

$$\begin{aligned} \bar{\alpha} & \triangleq \sup_{\delta_1, \delta_2} \min \left\{ \frac{(1 - \sigma_C) - \kappa_2\|R\|\delta_2}{s_1(1 + \delta_1) + s_2\delta_2}, \frac{(1 - \sigma_R)\delta_2}{s_4(1 + \delta_1) + s_5\delta_2 + s(\beta)}, \frac{1}{nL} \right\} \\ \text{s.t.} \quad & \frac{L}{\mu} < \delta_1, \quad 0 < \delta_2 < \frac{1 - \sigma_C}{\kappa_2\|R\|}. \end{aligned} \tag{25}$$

Then, the spectral radius satisfies $\rho(\Upsilon_{\alpha,\beta}) < 1$.

Proof. Given $\alpha \in (0, \frac{2}{nL})$ and $\beta > 0$ such that $\alpha\beta < 1$, it follows from Lemmas 6-8 that (24) holds for $k \geq 0$. Next, we show $\rho(\Upsilon_{\alpha,\beta}) < 1$. Since $\Upsilon_{\alpha,\beta}$ has all non-negative entries, it is sufficient to show that $\Upsilon_{\alpha,\beta} \gamma < \gamma$ for some positive $\gamma = [\gamma_1, \gamma_2, \gamma_3]^\top \in \mathbb{R}_+^3$ —see [51, Corollary 8.1.29]. Since $L \geq \mu$, according to the definition of η in Lemma 5, $\eta = 1 - \alpha n \mu$ for $\alpha \in (0, \frac{1}{nL})$. Hence, $\Upsilon_{\alpha,\beta} \gamma < \gamma$ is equivalent to

$$(s_1\gamma_1 + s_1\gamma_2 + s_2\gamma_3)\alpha < \gamma_1(1 - \sigma_C) - \kappa_2 \|R\| \gamma_3, \quad (26a)$$

$$s_3\gamma_1\alpha - \gamma_2 n \mu \alpha < 0, \quad (26b)$$

$$((s_4 + s(\beta))\gamma_1 + s_4\gamma_2 + s_5\gamma_3)\alpha < \gamma_3(1 - \sigma_R). \quad (26c)$$

Clearly, (26) holds for all $\alpha \in (0, \bar{\alpha})$ and $\gamma \in \mathbb{R}^3$ such that $\gamma_2 = \delta_1\gamma_1$ and $\gamma_3 = \delta_2\gamma_1$ for any $\gamma_1 > 0$ and $\delta_1, \delta_2 > 0$ satisfying (25); thus, we get $\rho(\Upsilon_{\alpha,\beta}) < 1$. \square

Remark 11. Note δ_1 and δ_2 are only required to satisfy (25). To provide a lower bound on an admissible α , we compute a lower bound on $\bar{\alpha}$ by setting $\delta_2 = \frac{1-\sigma_C}{2\kappa_2 \|R\|}$ satisfying (25). The supremum over δ_1 subject to (25) is achieved at $\delta_1 = \frac{L}{\mu}$. For this particular choice we get $\bar{\alpha} < \frac{1}{nL}$ and $\bar{\alpha} \geq \min\{\alpha_1, \alpha_2\}$, where

$$\begin{aligned} \alpha_1 &\triangleq \left[\frac{2\kappa_4}{1-\sigma_C} \left(\frac{L}{\mu} + 1 \right) nL + \frac{\kappa_4}{\kappa_2} \tilde{v}L \right]^{-1}, \\ \alpha_2 &\triangleq (1 - \sigma_R) \left[\frac{\kappa_2 \kappa_3 \|R\|}{1 - \sigma_C} \left(\frac{L}{\mu} + 1 \right) nL + \frac{\kappa_1 \kappa_2 \|R\| \|I - R\| \beta}{1 - \sigma_C} + \kappa_3(1 + v)\tilde{v}L \right]^{-1}, \end{aligned}$$

where we used $1 = \rho(R) \leq \|R\|$.

Finally, in the next theorem, we prove that FRSD iterate sequence converges with a linear rate through showing a linear decay for $\{\Phi_k\}$. First, we state a classic result that will be useful in our analysis; for its proof, see [50, 52].

Lemma 9. Let $\{a_k\}$, $\{b_k\}$, $\{c_k\}$ and $\{d_k\}$ be non-negative sequences such that $\sum_{k=0}^{\infty} c_k < \infty$, $\sum_{k=0}^{\infty} d_k < \infty$, and

$$a_{k+1} \leq (1 + c_k)a_k - b_k + d_k, \quad \forall k \geq 0.$$

Then $\{a_k\}$ converges and $\sum_{k=0}^{\infty} b_k < \infty$.

Theorem 2. Let Assumptions 1-3 hold. For any step-size $\alpha \in (0, \bar{\alpha})$, the sequence $\{\mathbf{x}(k)\}$ converges with a linear rate to $\mathbf{x}^* = \mathbf{1}_n \otimes \mathbf{x}^*$ with a rate arbitrarily close to $\varphi_{\alpha,\beta} \triangleq \max\{\rho(\Upsilon_{\alpha,\beta}), \sigma_R\} \in (0, 1)$, where $\bar{\alpha}$ is defined in the Theorem 1, and $\sigma_R = \left\| \left(\bar{R} - (\mathbf{1}_n \boldsymbol{\pi}^\top) \right) \otimes I_p \right\| < 1$.

Proof. The proof follows from similar arguments as in the proof of [37, Lemma 5]. Theorem 1 shows that $\rho(\Upsilon_{\alpha,\beta}) < 1$; hence, from [51, Lemma 5.6.10], given any positive $\zeta < 1 - \varphi_{\alpha,\beta}$, there exists a matrix norm $\|\cdot\|_{(\zeta)}$ such that $\|\Upsilon_{\alpha,\beta}\|_{(\zeta)} \leq \rho(\Upsilon_{\alpha,\beta}) + \frac{\zeta}{2}$. Since norms are equivalent in finite-dimensional spaces, we conclude that there exists some $\Gamma > 0$ such that we have

$$\|\Upsilon_{\alpha,\beta}^k\| \leq \Gamma \tilde{\lambda}^k, \quad \|\Upsilon_{\alpha,\beta}^{k-j-1} \Phi_j\| \leq \Gamma \tilde{\lambda}^k, \quad (27)$$

for all $0 \leq j \leq k-1$, where $\tilde{\lambda} \triangleq \max\{\sigma_R, \rho(\Upsilon_{\alpha,\beta}) + \frac{\zeta}{2}\} < 1$. By writing (24) recursively, we get, for all $k \geq 0$,

$$\theta_k \leq \Upsilon_{\alpha,\beta}^k \theta_0 + \sum_{j=0}^{k-1} \Upsilon_{\alpha,\beta}^{k-j-1} \Phi_j \Psi_j. \quad (28)$$

Since all the terms in (28) have non-negative entries, using (27), we get for all $k \geq 0$,

$$\|\theta_k\| \leq \|\Upsilon_{\alpha,\beta}^k\| \|\theta_0\| + \sum_{j=0}^{k-1} \|\Upsilon_{\alpha,\beta}^{k-j-1} \Phi_j\| \|\Psi_j\| \leq \Gamma \tilde{\lambda}^k (\|\theta_0\| + \sum_{j=0}^{k-1} \|\Psi_j\|). \quad (29)$$

For any $k \geq 0$, we can bound $\|\Psi_k\|$ as follows:

$$\begin{aligned} \|\Psi_k\| &\leq \|\nabla f(\mathbf{x}(k)) - \nabla f(\mathbf{x}^*)\| + \|\nabla f(\mathbf{x}^*)\| \leq L \|\mathbf{x}(k) - \hat{\mathbf{x}}(k)\| + L \|\hat{\mathbf{x}}(k) - \mathbf{x}^*\| + \|\nabla f(\mathbf{x}^*)\| \\ &\leq 2L \|\theta_k\| + \|\nabla f(\mathbf{x}^*)\|. \end{aligned} \quad (30)$$

Thus, for all $k \geq 0$, combining (29) and (30) we get

$$\|\theta_k\| \leq \left(\|\theta_0\| + 2L \sum_{j=0}^{k-1} \|\theta_j\| + k \|\nabla f(\mathbf{x}^*)\| \right) \Gamma \tilde{\lambda}^k.$$

For $k \geq 0$, let $a_k \triangleq \sum_{j=0}^{k-1} \|\theta_j\|$, $b_k \triangleq 0$, $\tilde{c} \triangleq 2L\Gamma$, and $\tilde{d}_k \triangleq \Gamma \|\theta_0\| + k\Gamma \|\nabla f(\mathbf{x}^*)\|$; hence, we get

$$\|\theta_k\| = a_{k+1} - a_k \leq (\tilde{c}a_k + \tilde{d}_k) \tilde{\lambda}^k, \quad \forall k \geq 0. \quad (31)$$

Define $c_k \triangleq \tilde{c} \tilde{\lambda}^k \geq 0$ and $d_k \triangleq \tilde{d}_k \tilde{\lambda}^k \geq 0$ for $k \geq 0$. Since $\tilde{\lambda} \in (0, 1)$, we have $\sum_{k=0}^{\infty} c_k + d_k < \infty$; therefore, Lemma 9 implies that $\{a_k\}$ converges. Furthermore, our choice of $\zeta > 0$ and the definition of $\tilde{\lambda}$ imply that $\tilde{\lambda} + \frac{\zeta}{2} \in (0, 1)$; therefore, since $\{a_k\}$ is bounded, (31) leads to

$$\lim_{k \rightarrow \infty} \frac{\|\theta_k\|}{(\tilde{\lambda} + \frac{\zeta}{2})^k} \leq \frac{(\tilde{c}a_k + \tilde{d}_k) \tilde{\lambda}^k}{(\tilde{\lambda} + \frac{\zeta}{2})^k} = 0. \quad (32)$$

Thus, there exist $c > 0$ such that

$$\|\theta_k\| \leq c(\tilde{\lambda} + \frac{\zeta}{2})^k, \quad \forall k \geq 0, \quad (33)$$

Thus, we get the desired result since for all $k \geq 0$,

$$\|\mathbf{x}(k) - \mathbf{x}^*\| \leq \|\mathbf{x}(k) - \hat{\mathbf{x}}(k)\| + \|\hat{\mathbf{x}}(k) - \mathbf{x}^*\| \leq 2\|\theta_k\| \leq 2c(\tilde{\lambda} + \frac{\zeta}{2})^k \leq 2c(\varphi_{\alpha, \beta} + \zeta)^k,$$

and we clearly have $\varphi_{\alpha, \beta} + \zeta < 1$. \square \square

4 Numerical Results

In this section, we provide some numerical results to demonstrate the performance of the proposed method against the state-of-the-art competitive algorithms designed for consensus optimization over directed graphs. In our experiments, we considered two types of distributed regression problems of the form given in (1) to test FRSD method: (a) regression with Huber loss, (b) the logistic regression, which are described in Sections 4.1 and 4.2, respectively. Throughout the experiments, we use the uniform weights for the row-stochastic matrix defined in (5), i.e., $r_{ij} = 1/|\mathcal{N}_i^{\text{in}}|$ for all $i \in \mathcal{V}$, and we use coordinated step-size and momentum parameters for \mathcal{ABm} , \mathcal{ABN} , FROZEN and D-DNGT to have a fair comparison with other methods.

For both distributed regression problems, we compare FRSD with Xi-row [37], FROZEN [36] and D-DNGT [39], which use only *row-stochastic* weights similar to our method, with Push-DIGing [31], which utilizes *column-stochastic* weights, and also with \mathcal{AB} [33], \mathcal{ABm} [34], \mathcal{ABN} [36] and Push-Pull [35], which use both *row-stochastic* and *column-stochastic* weights over six different time-invariant directed graphs with $n = 10, 30, 50, 100$ and 200 nodes (agents), see Figure 1.

Furthermore, in Section 4.2.2, we also conducted a numerical test over the random graphs comparing the proposed method with the other row-stochastic methods, i.e., Xi-row, FROZEN and D-DNGT.

4.1 Distributed Regression with Huber Loss

Suppose $\tilde{\mathbf{x}} \in \mathbb{R}^p$ is the *unknown* linear model, and for each $i \in \mathcal{V}$, let $\mathbf{b}_i \in \mathbb{R}^{m_i}$ be the corresponding noisy measurement vector, i.e., $\mathbf{b}_i = M_i \tilde{\mathbf{x}} + \mathbf{n}_i$ where $\mathbf{n}_i \in \mathbb{R}^{m_i}$ is the measurement noise vector. Given parameter $\xi > 0$, the Huber loss function $H_\xi : \mathbb{R} \rightarrow \mathbb{R}_+$ is defined as

$$H_\xi(z) = \begin{cases} \frac{1}{2}z^2, & \text{if } |z| \leq \xi; \\ \xi \left(|z| - \frac{1}{2}\xi \right) & \text{otherwise.} \end{cases}$$

For any $m \in \mathbb{Z}_+$, we also define $\mathbf{H}_\xi : \mathbb{R}^m \rightarrow \mathbb{R}^m$ such that $\mathbf{H}_\xi(\mathbf{z}) = [H_\xi(z_j)]_{j=1}^m$ where $\mathbf{z} = [z_j]_{j=1}^m \in \mathbb{R}^m$.

In this experiment, the goal is to estimate $\tilde{\mathbf{x}}$ with an optimal solution \mathbf{x}^* to the regression problem with Huber loss:

$$\mathbf{x}^* \in \underset{\mathbf{x} \in \mathbb{R}^p}{\operatorname{argmin}} \bar{f}(\mathbf{x}) \triangleq \frac{1}{n} \sum_{i \in \mathcal{V}} \mathbf{H}_\xi(M_i \mathbf{x} - \mathbf{b}_i). \quad (34)$$

In the experiments, we solve (34) over the set of directed graphs shown in Figure 1. We generate data as in [31] using $p = 5$ and $m_i = 10$ for $i \in \mathcal{V}$. We set the Huber loss parameter $\xi = 2$, and for each $i \in \mathcal{V}$, we generated $f_i(\mathbf{x}) = \mathbf{H}_\xi(M_i \mathbf{x} - \mathbf{b}_i)$ as described in [31, Sec. 6] such that $L_i = 1$. Moreover, we also initialized all the methods we tested from $\mathbf{x}_i(0) = \mathbf{0}$ for all $i \in \mathcal{V}$. In our experiments, $n \in \{10, 30, 50, 100, 200\}$, $m_i = 10$ for all $i \in \mathcal{V}$ and $p = 5$; therefore, \bar{f} and f_i for $i \in \mathcal{V}$ are restricted strongly convex when the regression error is small. In Fig. 2, we plot the

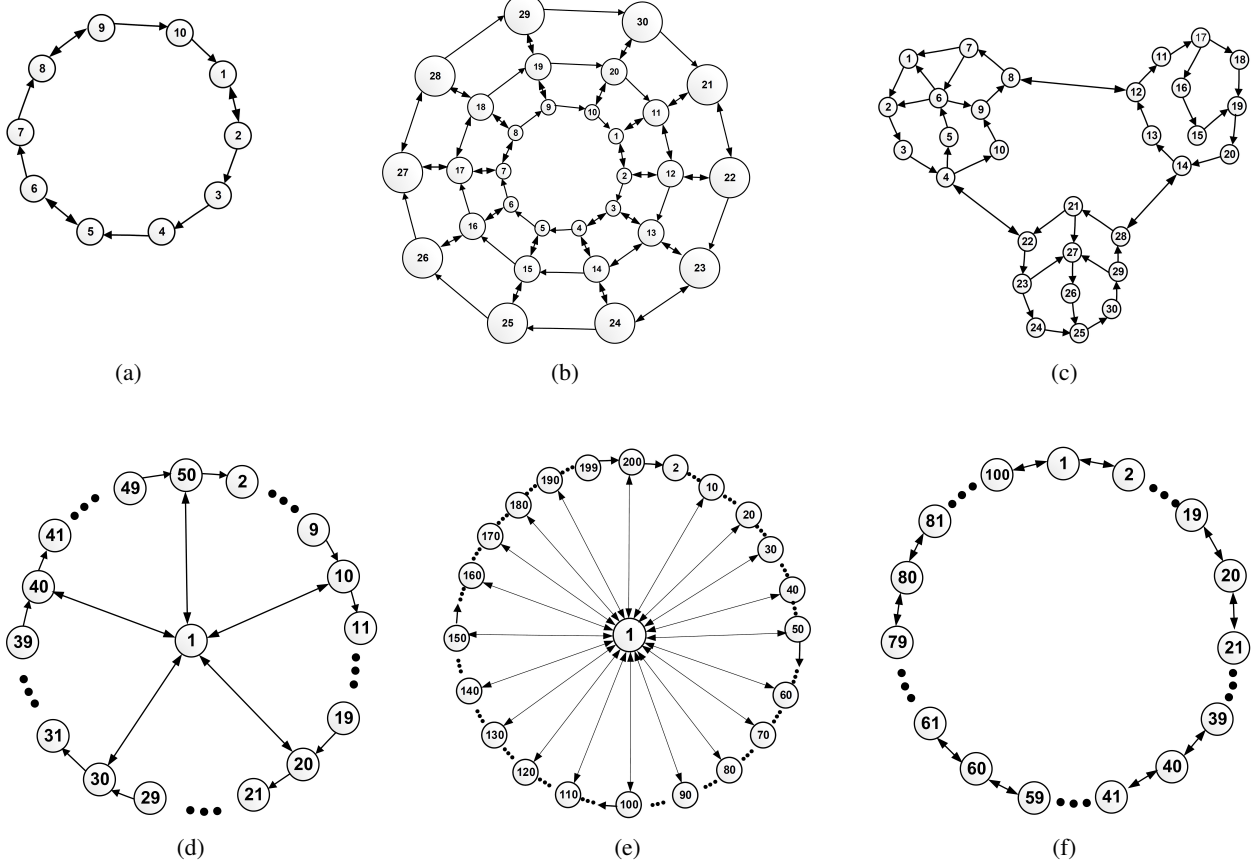


Figure 1: Strongly-connected digraphs tested in our experiments.

residual sequence $\{r(k)\}_{k \geq 0}$ for all the methods where $r(k) \triangleq \frac{\|\mathbf{x}(k) - \mathbf{x}^*\|}{\|\mathbf{x}(0) - \mathbf{x}^*\|}$. To optimize the convergence rate, we tuned parameters for all algorithms.

4.2 Distributed Logistic Regression

We now consider the distributed binary classification problem using the logistic regression to train a linear classifier. Suppose each node (agent) $i \in \mathcal{V}$ has access to $(M_i, \mathbf{b}_i) \in \mathbb{R}^{m_i \times p} \times \{-1, +1\}^{m_i}$. Let $L : \mathbb{R} \times \{-1, 1\} \rightarrow \mathbb{R}_+$ such that $L(u, v) = \ln(1 + \exp(-uv))$; and for any $m \in \mathbb{Z}_+$, we also define $\mathbf{L} : \mathbb{R}^m \times \{-1, 1\}^m \rightarrow \mathbb{R}_+^m$ such that $\mathbf{L}(\mathbf{u}, \mathbf{v}) = [L(u_j, v_j)]_{j=1}^m$ where $\mathbf{u} = [u_j]_{j=1}^m$ and $\mathbf{v} = [v_j]_{j=1}^m$. The linear classifier \mathbf{x}^* is computed by solving the regularized logistic regression problem:

$$\mathbf{x}^* = \underset{\mathbf{x} \in \mathbb{R}^p}{\operatorname{argmin}} \bar{f}(\mathbf{x}) \triangleq \frac{1}{n} \sum_{i \in \mathcal{V}} \left(\mathbf{L}(M_i \mathbf{x}, \mathbf{b}_i) + \frac{\lambda}{2} \|\mathbf{x}\|_2^2 \right). \quad (35)$$

where using regularization parameter $\lambda > 0$ improves the statistical properties of \mathbf{x}^* – see [53].

4.2.1 Tests on Specific Graph Topologies

In the first set of experiments, we use the Australian-scale data set [54] with 790 data points where each data point consists of a 14-dimensional feature vector, i.e., $p = 15$ to model the intercept, and the corresponding binary label.

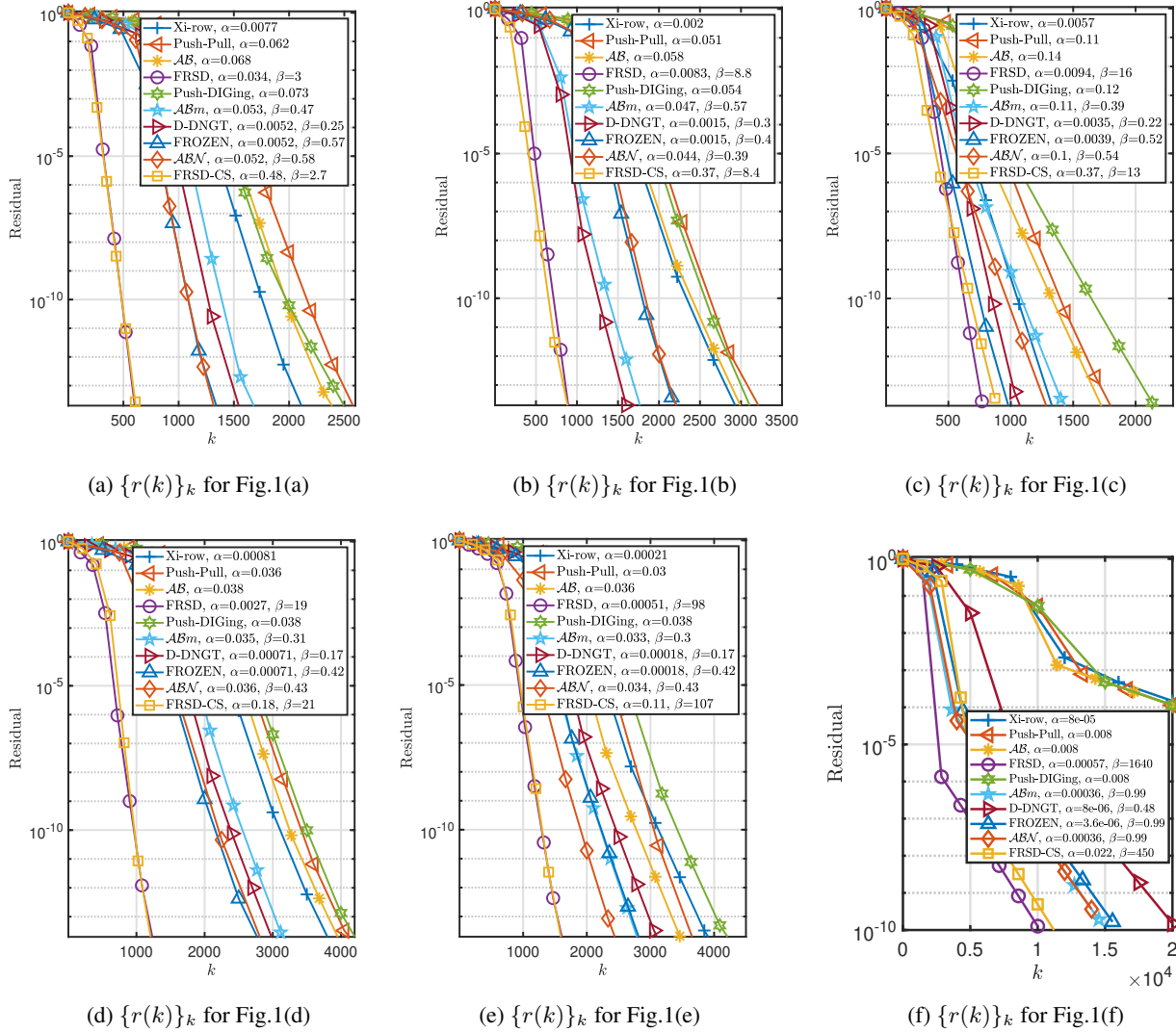


Figure 2: Distributed Regression with Huber Loss

Suppose each agent $i \in \mathcal{V}$ samples $m_i = 10$ data points uniformly at random from the training set with replacement. Hence, for each $i \in \mathcal{V}$, we form $M_i \in \mathbb{R}^{m_i \times p}$ using m_i data points with $p - 1$ features and set the last column of M_i to $\mathbf{1}_{m_i}$ in order to model the intercept. We test the proposed method FRSD against the same methods that we compared with in Section 4.1. The residual sequence $\{r(k)\}_{k \geq 1}$ for all the methods are shown in Fig. 3, where $r(k)$ is defined in Section 4.1.

4.2.2 Tests on Random Graphs

In the second set of experiments, we tested FRSD and FRSD-CS over random graphs against the other row-stochastic methods, i.e., Xi-row, FROZEN and D-DGNT, to solve the distributed logistic regression problem defined in Section 4.2. We considered two scenarios: Scenario I $n > p$ and Scenario II $p > n$ –recall that n and p denote the number of nodes in the network and the dimension of the decision variable, respectively. For Scenario I, i.e., $n > p$, we looked at two cases: low connectivity ratio (sparser graphs) and high connectivity ratio (denser graphs), where the connectivity ratio is defined as the ratio of the number edges to $n(n - 1)$ –note that $n(n - 1)$ is equal to the number of all possible edges excluding self-loops. For each scenario, we ran all 5 algorithms on 20 different randomly generated graphs. We reported the residual $r(k) \triangleq \|\mathbf{x}(k) - \mathbf{x}^*\| / \|\mathbf{x}(0) - \mathbf{x}^*\|$ against iteration counter k and we also reported the residual against the amount communication per node by the end of iteration k –recall that at each iteration FRSD and FRSD-CS require each node to broadcast only $n + p$ -dimensional vector while the others, i.e., Xi-row, FROZEN and D-DGNT, require

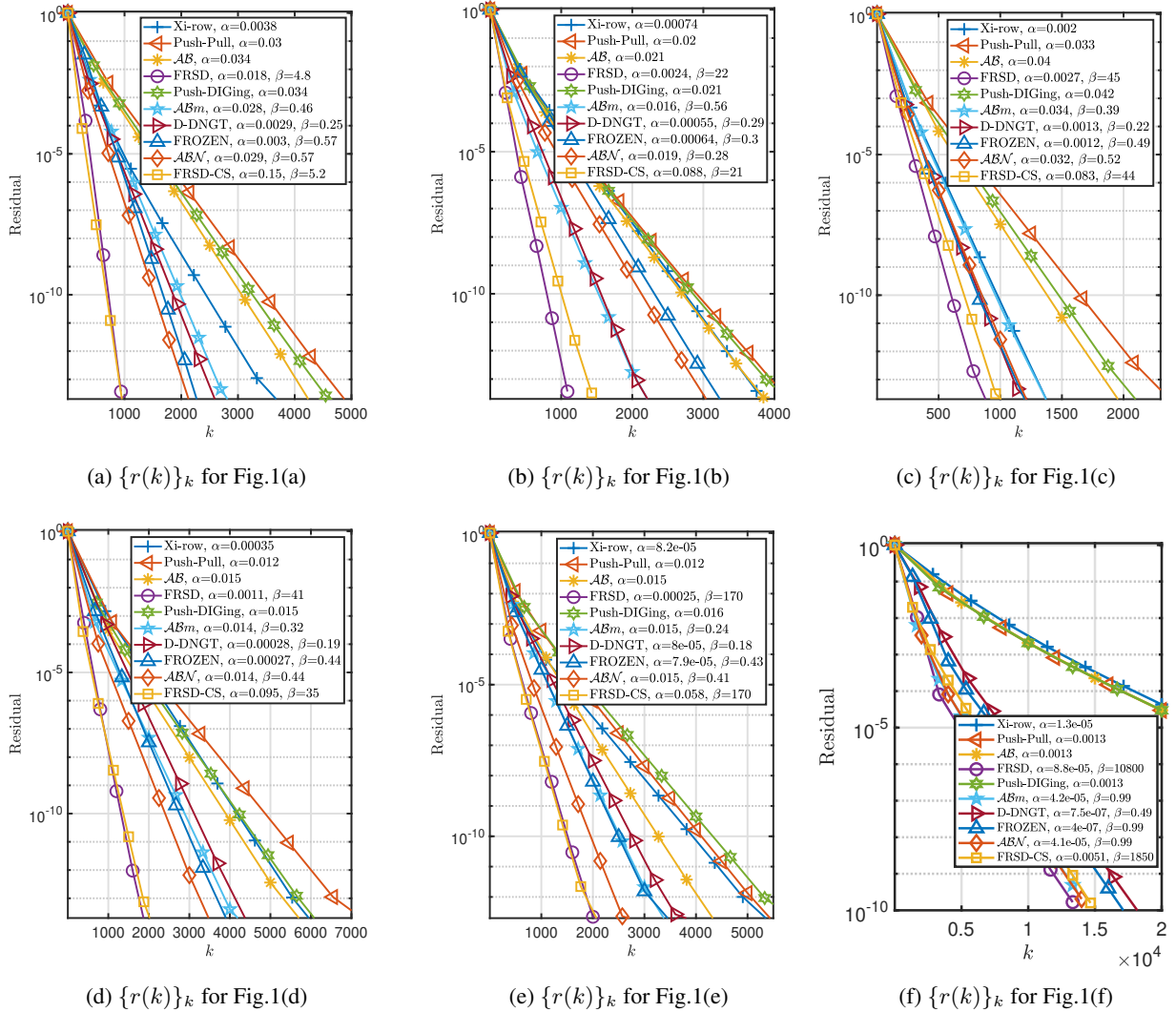


Figure 3: Distributed Logistic Regression

each node to broadcast $n + 2p$ -dimensional vector. We have observed that both FRSD and FRSD-CS are competitive against the state of the art row stochastic methods, and the performance of our algorithms is superior either when $n < p$ or when the graphs are sparse, which is indeed the case for most of the real-life networks. Next we describe how we generated the random graphs.

Random Graph Generation We used DGen code⁵ to generate strongly connected random graphs. The algorithm DGen, implemented in MATLAB, receives two input: number of nodes n , and the connectivity ratio $\phi \in (0, 1]$. Given n and ϕ , DGen generates a strongly connected random graph with $|\mathcal{E}| = \lceil \phi n(n-1) \rceil \triangleq m_{n,\phi}$ edges. Let $\{p_i\}_{i=1}^n$ be a permutation of $[n] \triangleq \{1, \dots, n\}$ chosen uniformly at random, and let $\mathcal{I} = \{i \in [n] : p_i \neq i\}$. Then DGen creates a directed cycle \mathcal{C} using the nodes $\{p_i\}_{i \in \mathcal{I}}$. Now consider a smaller dimensional graph with nodes $\{i : i \in [n] \setminus \mathcal{I}\} \cup \{c^*\}$ where c^* is a “super-node” representing the cycle \mathcal{C} . Note that this new graph has $n - |\mathcal{I}| + 1$ nodes; one can repeat the above process by setting $n \leftarrow n - |\mathcal{I}| + 1$, and whenever we connect a node from $[n] \setminus \mathcal{I}$ with c^* , one randomly picks a node belonging to \mathcal{C} . This process ends when the smaller dimensional graph has only a single super-node with no other nodes, which gives us a strongly connected graph. Say this graph has \tilde{m} nodes, then the remaining $m_{n,\phi} - \tilde{m}$ edges are randomly added to obtain a strongly connected graph with connectivity ratio ϕ .

⁵DGen code is written by Dr. W. Shi, see <https://sites.google.com/view/wilburshi/home/research/software-a-z-order/graph-tools/dgen>

In the experiments with randomly generated strongly connected graphs, we only tested row-stochastic methods. Indeed, being able to get away with the eigenvector estimation through employing both row- and column-stochastic mixing matrices, AB-type methods, e.g., \mathcal{AB} [33], \mathcal{ABm} [34], Push-Pull [35] and \mathcal{ABN} [36], perform better on these experiments than the first-order methods using row-stochastic weights alone. That is why we only reported the results for the row-stochastic methods to have a fair comparison among equivalent methods.

Scenario I ($n > p$) We set $n = 200$ and generated 20 random graphs for each connectivity ratio $\phi \in \{0.015, 0.15\}$. We use the same dataset and same problem setup with Section 4.2.1, i.e., $p = 15$ and the number of data points $m_i = 10$ for all $i \in \mathcal{V}$. In Figures 4 and 5, we report the results for *high* connectivity ratio $\phi = 0.15$ and the *low* one $\phi = 0.015$, respectively. For Scenario I, i.e., when $n > p$, we observe that FRSD and FRSD-CS are competitive against FROZEN and D-DGNT while performing better than Xi-row. Furthermore, we also observe that the performances of FRSD and FRSD-CS improve as the random graphs get sparser, i.e., they perform significantly better for $\phi = 0.015$ when compared to their performance for $\phi = 0.15$. Finally, as expected, for $n > p$, the residual shows the same decay patterns with respect to increase in either iteration counter or the amount of data broadcast per node.

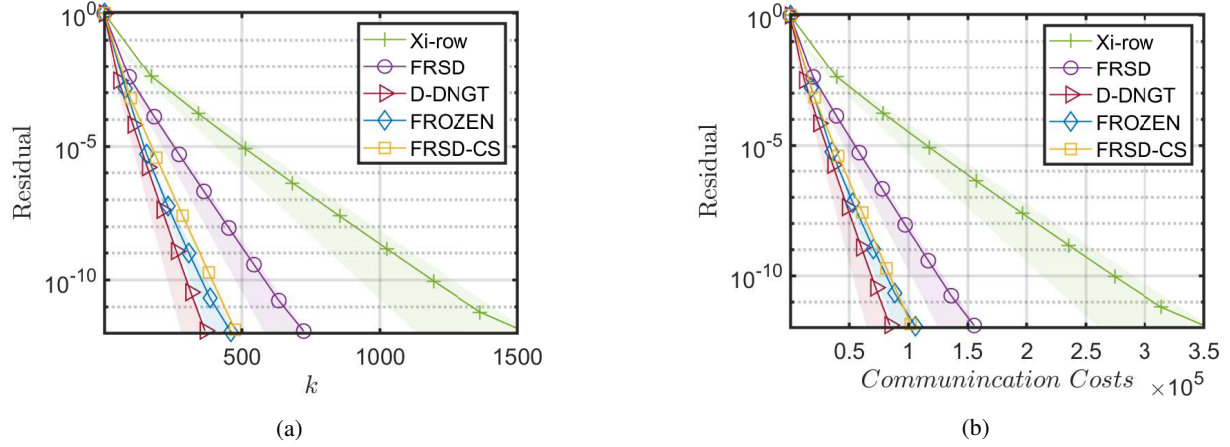


Figure 4: Distributed logistic regression problem ($n = 200$, $p = 15$) over 20 random directed graphs with a high connectivity ratio $\phi = 0.15$. Solid curves represent the average and the shaded region represents the range statistics.

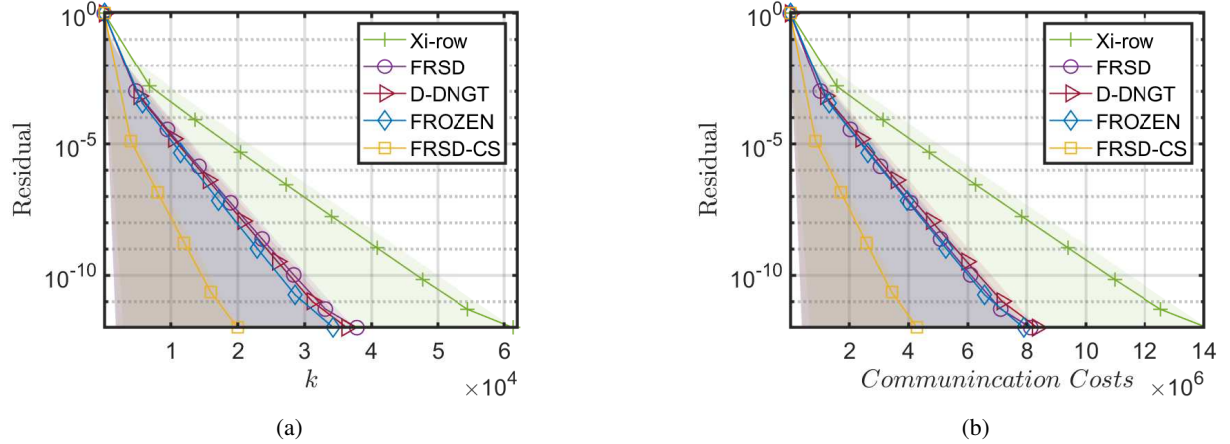


Figure 5: Distributed logistic regression problem ($n = 200$, $p = 15$) over 20 random directed graphs with a low connectivity ratio $\phi = 0.015$. Solid curves represent the average and the shaded region represents the range statistics.

Scenario II ($n < p$) We set $n = 25$ and generated 20 random graphs with connectivity ratio $\phi = 0.1$, i.e., each random graph has 60 edges. We used w1a.t (testing) dataset [54] with 47,272 data points with each data point consisting of 300 features vector – implying $p = 301$ to model the intercept. For classification, we again used the binary logistic regression model of Section 4.2.1 with $p = 301$ and $m_i = 400$ for all $i \in \mathcal{V}$. In Figure 6, we report the

results for this scenario. Indeed, when $n < p$, we observe that FRSD and FRSD-CS are competitive against FROZEN while performing better than both Xi-row and D-DGNT. Finally, unlike Scenario I, when $n < p$, the advantage of FRSD and FRSD-CS over the other row-stochastic method in terms of lower communication overhead becomes more apparent: while the residuals for FRSD and FROZEN show the same decay patterns as the iteration counter increases, one can observe that FRSD performs better than FROZEN considering the amount of data broadcast per node since both FRSD and FRSD-CS need each node to broadcast $n + p$ -dimensional vector, i.e., $\approx p$ as $p \gg n$, FROZEN requires broadcasting $n + 2p$ -dimensional vector, i.e., $\approx 2p$; hence, almost twice the communication overhead of FRSD.

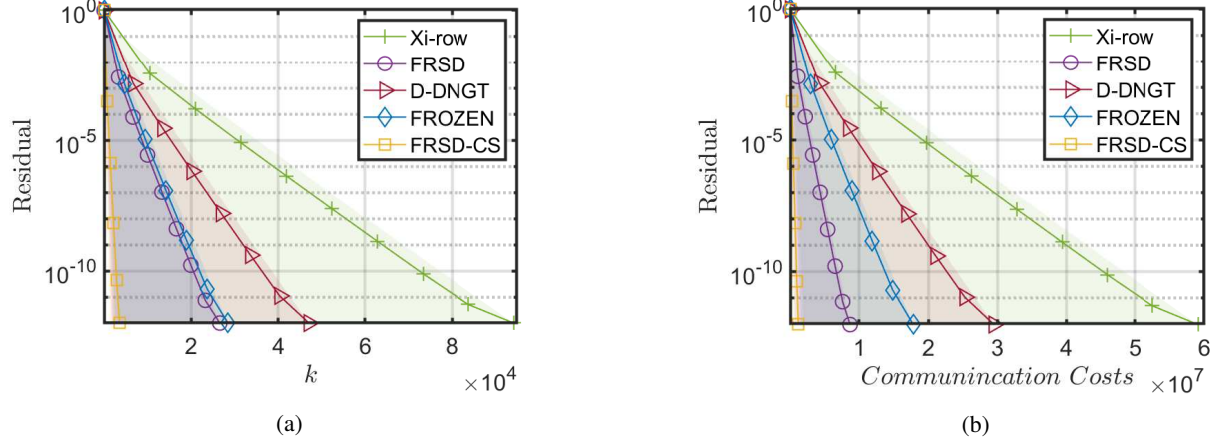


Figure 6: Distributed logistic regression problem ($n = 25$, $p = 301$) over 20 random directed graphs with a low connectivity ratio $\phi = 0.1$. Solid curves represent the average and the shaded region represents the range statistics.

5 Conclusion

In this paper, we proposed a distributed optimization algorithm, FRSD, for decentralized consensus optimization over directed graphs. FRSD only employs a row-stochastic matrix for local messaging with neighbors, making it desirable for broadcast-based communication systems. The proposed algorithm achieves a geometric convergence to the global optimal when agents' cost functions are strongly convex with Lipschitz continuous gradients. Empirical results demonstrated the efficacy of the implicit gradient tracking technique employed by FRSD, which led to: (i) reduction in the data stored, and (ii) reduction in the data broadcast, for each node. More precisely, FRSD does not need to store x iterate from the previous iteration while it is needed for all other methods explicitly using the gradient tracking term; furthermore, FRSD also eliminates the need for broadcasting a variable related to gradient tracking. As a future research direction, we consider extending our results to the asynchronous computation setting over directed communication graphs.

References

- [1] M. Rabbat and R. Nowak, "Distributed optimization in sensor networks," in *Proceedings of the 3rd international symposium on Information processing in sensor networks*, 2004, pp. 20–27.
- [2] U. A. Khan, S. Kar, and J. M. Moura, "DILAND: An algorithm for distributed sensor localization with noisy distance measurements," *IEEE Transactions on Signal Processing*, vol. 58, no. 3, pp. 1940–1947, 2009.
- [3] F. Bullo, J. Cortes, and S. Martinez, *Distributed control of robotic networks: a mathematical approach to motion coordination algorithms*. Princeton University Press, 2009, vol. 27.
- [4] V. Cevher, S. Becker, and M. Schmidt, "Convex optimization for big data: Scalable, randomized, and parallel algorithms for big data analytics," *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 32–43, 2014.
- [5] S. Boyd, N. Parikh, and E. Chu, *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.
- [6] R. Bekkerman, M. Bilenko, and J. Langford, *Scaling up machine learning: Parallel and distributed approaches*. Cambridge University Press, 2011.

- [7] H. Raja and W. U. Bajwa, “Cloud k-svd: A collaborative dictionary learning algorithm for big, distributed data,” *IEEE Transactions on Signal Processing*, vol. 64, no. 1, pp. 173–188, 2015.
- [8] M. Assran, N. Loizou, N. Ballas, and M. Rabbat, “Stochastic gradient push for distributed deep learning,” in *International Conference on Machine Learning*. PMLR, 2019, pp. 344–353.
- [9] K. I. Tsianos, S. Lawlor, and M. G. Rabbat, “Push-sum distributed dual averaging for convex optimization,” in *2012 IEEE 51st IEEE conference on decision and control (cdc)*. IEEE, 2012, pp. 5453–5458.
- [10] —, “Consensus-based distributed optimization: Practical issues and applications in large-scale machine learning,” in *2012 50th annual allerton conference on communication, control, and computing (allerton)*. IEEE, 2012, pp. 1543–1550.
- [11] K. Yuan, B. Ying, X. Zhao, and A. H. Sayed, “Exact diffusion for distributed optimization and learning—part i: Algorithm development,” *IEEE Transactions on Signal Processing*, vol. 67, no. 3, pp. 708–723, 2018.
- [12] J. Tsitsiklis, D. Bertsekas, and M. Athans, “Distributed asynchronous deterministic and stochastic gradient optimization algorithms,” *IEEE Transactions on Automatic Control*, vol. 31, no. 9, pp. 803–812, 1986.
- [13] A. Nedic and A. Ozdaglar, “Distributed subgradient methods for multi-agent optimization,” *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.
- [14] A. Nedic, A. Ozdaglar, and P. A. Parrilo, “Constrained consensus and optimization in multi-agent networks,” *IEEE Transactions on Automatic Control*, vol. 55, no. 4, pp. 922–938, 2010.
- [15] S. S. Ram, A. Nedić, and V. V. Veeravalli, “Distributed stochastic subgradient projection algorithms for convex optimization,” *Journal of optimization theory and applications*, vol. 147, no. 3, pp. 516–545, 2010.
- [16] J. C. Duchi, A. Agarwal, and M. J. Wainwright, “Dual averaging for distributed optimization: Convergence analysis and network scaling,” *IEEE Transactions on Automatic control*, vol. 57, no. 3, pp. 592–606, 2011.
- [17] M. Zhu and S. Martínez, “On distributed convex optimization under inequality and equality constraints,” *IEEE Transactions on Automatic Control*, vol. 57, no. 1, pp. 151–164, 2011.
- [18] D. Jakovetić, J. Xavier, and J. M. Moura, “Fast distributed gradient methods,” *IEEE Transactions on Automatic Control*, vol. 59, no. 5, pp. 1131–1146, 2014.
- [19] W. Shi, Q. Ling, G. Wu, and W. Yin, “Extra: An exact first-order algorithm for decentralized consensus optimization,” *SIAM Journal on Optimization*, vol. 25, no. 2, pp. 944–966, 2015.
- [20] E. Wei and A. Ozdaglar, “On the $\mathcal{O}(1/k)$ convergence of asynchronous distributed alternating direction method of multipliers,” in *2013 IEEE Global Conference on Signal and Information Processing*. IEEE, 2013, pp. 551–554.
- [21] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, “On the linear convergence of the ADMM in decentralized consensus optimization,” *IEEE Transactions on Signal Processing*, vol. 62, no. 7, pp. 1750–1761, 2014.
- [22] A. Mokhtari, W. Shi, Q. Ling, and A. Ribeiro, “DQM: Decentralized quadratically approximated alternating direction method of multipliers,” *IEEE Transactions on Signal Processing*, vol. 64, no. 19, pp. 5158–5173, 2016.
- [23] —, “A decentralized second-order method with exact linear convergence rate for consensus optimization,” *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 4, pp. 507–522, 2016.
- [24] N. S. Aybat, Z. Wang, T. Lin, and S. Ma, “Distributed linearized alternating direction method of multipliers for composite convex consensus optimization,” *IEEE Transactions on Automatic Control*, vol. 63, no. 1, pp. 5–20, 2017.
- [25] N. Aybat, Z. Wang, and G. Iyengar, “An asynchronous distributed proximal gradient method for composite convex optimization,” in *International Conference on Machine Learning*. PMLR, 2015, pp. 2454–2462.
- [26] N. S. Aybat and E. Yazdandoost Hamedani, “A primal-dual method for conic constrained distributed optimization problems,” in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016.
- [27] N. S. Aybat and E. Y. Hamedani, “A distributed admm-like method for resource sharing over time-varying networks,” *SIAM Journal on Optimization*, vol. 29, no. 4, pp. 3036–3068, 2019.
- [28] D. Kempe, A. Dobra, and J. Gehrke, “Gossip-based computation of aggregate information,” in *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings*. IEEE, 2003, pp. 482–491.
- [29] A. Nedić and A. Olshevsky, “Distributed optimization over time-varying directed graphs,” *IEEE Transactions on Automatic Control*, vol. 60, no. 3, pp. 601–615, 2014.
- [30] C. Xi and U. A. Khan, “Dextra: A fast algorithm for optimization over directed graphs,” *IEEE Transactions on Automatic Control*, vol. 62, no. 10, pp. 4980–4993, 2017.

- [31] A. Nedic, A. Olshevsky, and W. Shi, “Achieving geometric convergence for distributed optimization over time-varying graphs,” *SIAM Journal on Optimization*, vol. 27, no. 4, pp. 2597–2633, 2017.
- [32] C. Xi, R. Xin, and U. A. Khan, “ADD-OPT: Accelerated distributed directed optimization,” *IEEE Transactions on Automatic Control*, vol. 63, no. 5, pp. 1329–1339, 2017.
- [33] R. Xin and U. A. Khan, “A linear algorithm for optimization over directed graphs with geometric convergence,” *IEEE Control Systems Letters*, vol. 2, no. 3, pp. 315–320, 2018.
- [34] —, “Distributed heavy-ball: A generalization and acceleration of first-order methods with gradient tracking,” *IEEE Transactions on Automatic Control*, 2019.
- [35] S. Pu, W. Shi, J. Xu, and A. Nedic, “Push-pull gradient methods for distributed optimization in networks,” *IEEE Transactions on Automatic Control*, 2020.
- [36] R. Xin, D. Jakovetić, and U. A. Khan, “Distributed nesterov gradient methods over arbitrary graphs,” *IEEE Signal Processing Letters*, vol. 26, no. 8, pp. 1247–1251, 2019.
- [37] C. Xi, V. S. Mai, R. Xin, E. H. Abed, and U. A. Khan, “Linear convergence in optimization over directed graphs with row-stochastic matrices,” *IEEE Transactions on Automatic Control*, vol. 63, no. 10, pp. 3558–3565, 2018.
- [38] 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.
- [39] Q. Lü, X. Liao, H. Li, and T. Huang, “A nesterov-like gradient tracking algorithm for distributed optimization over directed networks,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2020.
- [40] Y. Tian, Y. Sun, and G. Scutari, “Asy-sonata: Achieving linear convergence in distributed asynchronous multiagent optimization,” in *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2018, pp. 543–551.
- [41] J. Zhang and K. You, “Asyspa: An exact asynchronous algorithm for convex optimization over digraphs,” *IEEE Transactions on Automatic Control*, vol. 65, no. 6, pp. 2494–2509, 2019.
- [42] M. S. Assran and M. G. Rabbat, “Asynchronous gradient push,” *IEEE Transactions on Automatic Control*, vol. 66, no. 1, pp. 168–183, 2020.
- [43] P. Xie, K. You, and C. Wu, “How to stop consensus algorithms, locally?” in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 4544–4549.
- [44] M. Prakash, S. Talukdar, S. Attree, V. Yadav, and M. V. Salapaka, “Distributed stopping criterion for consensus in the presence of delays,” *IEEE Transactions on Control of Network Systems*, vol. 7, no. 1, pp. 85–95, 2019.
- [45] A. H. Sayed, “Adaptation, learning, and optimization over networks,” *Foundations and Trends in Machine Learning*, vol. 7, no. ARTICLE, pp. 311–801, 2014.
- [46] J. Xu, Y. Tian, Y. Sun, and G. Scutari, “Accelerated primal-dual algorithms for distributed smooth convex optimization over networks,” in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 2381–2391.
- [47] E. Y. Hamedani and N. S. Aybat, “A decentralized primal-dual method for constrained minimization of a strongly convex function,” *IEEE Transactions on Automatic Control*, vol. 67, no. 11, pp. 5682–5697, 2021.
- [48] A. Chambolle and T. Pock, “On the ergodic convergence rates of a first-order primal–dual algorithm,” *Mathematical Programming*, vol. 159, no. 1, pp. 253–287, 2016.
- [49] G. Qu and N. Li, “Harnessing smoothness to accelerate distributed optimization,” *IEEE Transactions on Control of Network Systems*, vol. 5, no. 3, pp. 1245–1260, 2017.
- [50] H. Robbins and D. Siegmund, *Optimizing methods in statistics (Proc. Sympos., Ohio State Univ., Columbus, Ohio, 1971)*. New York: Academic Press, 1971, ch. A convergence theorem for non negative almost supermartingales and some applications, pp. 233 – 257.
- [51] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge university press, 2012.
- [52] B. T. Polyak, “Introduction to optimization. optimization software,” *Inc., Publications Division, New York*, vol. 1, 1987.
- [53] K. Sridharan, S. Shalev-Shwartz, and N. Srebro, “Fast rates for regularized objectives,” *Advances in neural information processing systems*, vol. 21, pp. 1545–1552, 2008.
- [54] C.-C. Chang and C.-J. Lin, “LIBSVM: A library for support vector machines,” *ACM transactions on intelligent systems and technology (TIST)*, vol. 2, no. 3, pp. 1–27, 2011.