Distributed Optimization Algorithms over Large-Scale Networks with Applications to Microgrids

Diyako Ghaderyan
Doctoral program in Applied Mathematics
Department of Mathematics
2023

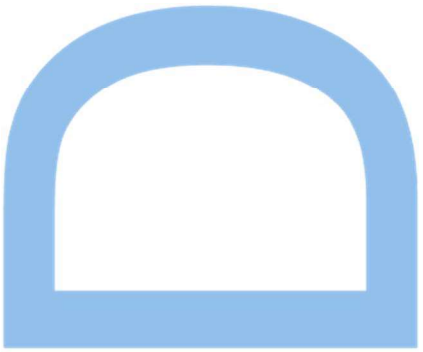
Supervisor
António Pedro Aguiar, Associated professor,
Faculty of Engineering, University of Porto

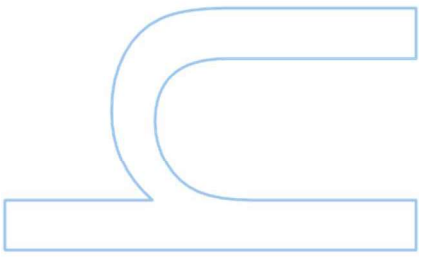


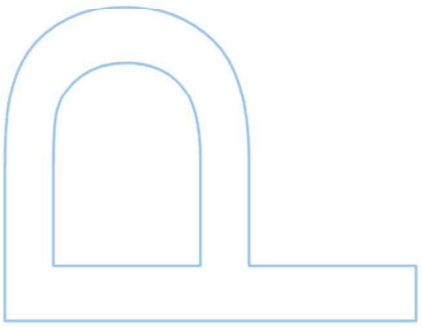


Universidade do Minho









UNIVERSIDADE DO PORTO

DOCTORAL THESIS

Distributed Optimization Algorithms over Large-Scale Networks with Applications to Microgrids

Author:
Diyako GHADERYAN

Supervisor:
A. Pedro AGUIAR

A thesis submitted in fulfilment of the requirements
for the degree of PhD in Applied Mathematics

at the

Faculdade de Ciências da Universidade do Porto
Departamento de Matemática

September 13, 2023

Sworn Statement

I, Diyako Ghaderyan, enrolled in the Doctor's Degree of Applied Mathematics at the Faculty of Sciences of the University of Porto hereby declare, in accordance with the provisions of paragraph a) of Article 14 of the Code of Ethical Conduct of the University of Porto, that the content of this thesis reflects perspectives, research work and my own interpretations at the time of its submission. By submitting this thesis, I also declare that it contains the results of my I further declare that all references to other authors fully comply with the I am aware that the practice of plagiarism and self-plagiarism constitute a form of academic offense.

Diyako Ghaderyan

Abstract

In the last years, rapid advances in artificial intelligence, embedded systems, and communication technologies have led to computational network systems that require algorithms to solve optimization problems with enormous physically distributed and/or private data sets 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 a consensus decision, optimization techniques can be used to solve a consensus problem in a distributed manner by employing only local computations and communication among neighboring nodes that can directly communicate with each other.

In the first part of this thesis, we propose a distributed optimization approach, which enjoys the advantages of both the primal and the dual domain methods, to solve the economic dispatch problem in a distributed way over undirected networks. A key requirement point is that the convergence rate of the optimization method must be fast when compared with recent state of the art distributed approaches. We show through simulations that for the IEEE 14-bus and the IEEE 39-bus distribution test systems, the proposed algorithm achieves indeed a significantly higher convergence rate effectively attaining this goal.

In the second part of the thesis, we develop 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. In contrast to the existing methods for directed networks, we show that FRSD enjoys linear convergence without employing a gradient tracking (GT) technique explicitly. The key idea is that it implements a GT implicitly through a novel momentum term, which leads to a significant reduction in communication and storage overhead for each node. This is particularly remarkable 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 showing that FRSD is indeed competitive with or outperforms other recent approaches.

Resumo

Nos últimos anos, com os rápidos avanços que se têm verificado na inteligência artificial e nas tecnologias de comunicação com especial relevo nos sistemas de redes computacionais, é cada vez mais possível endereçar problemas de otimização que envolvem enormes conjuntos de dados que são fisicamente distribuídos e/ou privados, a fim de atingir objetivos de desempenho ao nível do sistema, em que cada agente, representado por um nó na rede, tem de chegar a acordo sobre uma decisão comum. Para alcançar uma decisão de consenso ideal, podem ser utilizadas técnicas de otimização descentralizadas empregando apenas computações locais e comunicação entre nós vizinhos que possam comunicar diretamente uns com os outros.

Nesta tese, começamos em primeiro lugar por introduzir um método de otimização distribuída que goza das vantagens dos métodos de domínio primário e dual, para resolver o problema de despacho económico de uma forma distribuída sobre redes não direcionadas. Uma contribuição relevante é o projeto de um método de otimização distribuída, que exibe uma rápida taxa de convergência. Com efeito, apresentamos através de simulações que para os sistemas de teste de distribuição IEEE 14-bus e IEEE 39-bus, o algoritmo proposto atinge uma taxa de convergência significativamente mais elevada quando comparado com outras abordagens distribuídas recentemente desenvolvidas para resolver o problema de despacho económico proposto.

Na segunda parte da tese, desenvolvemos um algoritmo descentralizado e de rápida convergência, referido como FRSD, para resolver problemas de otimização de consensos sobre gráficos de comunicação dirigida. O algoritmo proposto utiliza apenas pesos estocásticos em linha, conduzindo a certas vantagens práticas na abordagem da comunicação quando comparado com técnicas que requerem pesos estocásticos em coluna. Em contraste com os métodos existentes para redes dirigidas, o FRSD goza de convergência linear sem empregar explicitamente uma técnica de seguimento de gradiente descendente (GD), uma vez que utiliza o GD implicitamente com o emprego de um termo de momentum. Observa-se que este novo termo leva a uma redução significativa na sobrecarga de comunicação e armazenamento para cada nó quando o FRSD é implementado para resolver problemas de alta dimensão em redes de pequena e média escala. Nos testes numéricos, comparamos o FRSD com outros métodos de última geração, que utilizam pesos de linha e ou de coluna.

Acknowledgements

I would like to express my deepest attitude to my supervisors Prof. A. Pedro Aguiar and Prof. Fernando Lobo Pereira for their support and guidance through my Ph.D career. It has been a great honor to work with them and to learn from them over the past four years. This thesis without their comments and support would not have been possible. Unfortunately, Prof. Fernando Lobo Pereira passed away and he is no longer with us. He is a good man and I hope his soul is at peace.

I am also grateful to Dr. Necdet S. Aybat (Penn State) for giving me the opportunity to collaborate with him during my graduate study. From him, I learned to be a qualified researcher as well as a tolerant person. I can always obtain helpful suggestions from his sharp insights into mathematical problems.

I would like to thank Prof. Maria do Rosário de Pinho for giving me an opportunity to work under her supervision as a researcher before starting my Ph.D career. I also would like to thank Paulo Lopes for all his support in administrative matters and for providing help on many personal issues. I also like to thank to my lab mates who made this journey an enjoyable experience. I also like to thank Doctoral program PDMA-NORTE-08-5369-FSE-000061 for financially supporting this work.

Saying thank you is not enough to my parents, brothers, and sister for limitless sacrifices, support and indefinable love. Last but not least, to the love of my life Nashmil. Thank you for all your unconditional love and support. To acknowledge your role and appreciate your love, I would like to dedicate my thesis to you.

Contents

Sworn Statement iii

Abstract v

Resumo vii

Acknowledgements ix

Contents xi

List of Figures xiii

List of Tables xv

Glossary xvii

1 Introduction 1
- 1.1 Context . . . 2
- 1.2 Distributed Architectures . . . 5
- 1.3 Distributed Optimization Methods . . . 8
- 1.4 Challenges . . . 18
- 1.5 Thesis outline and contributions . . . 18

2 Distributed Optimization for Power systems 21
- 2.1 Introduction . . . 21
- 2.2 Problem Formulation . . . 23
- 2.3 Distributed Scheduling . . . 25
- 2.4 Performance Assessment . . . 30

3 A Fast Row-Stochastic Decentralized Method for Distributed Optimization Over Directed Graphs 41
- 3.1 Introduction . . . 41
- 3.2 Design, Comparison and analysis of FRSD . . . 45
- 3.3 Convergence Results . . . 55
- 3.4 Numerical Results . . . 66

4 Conclusions and Future Work 75

A **77**

Bibliography **81**

List of Figures

1.1 Microgrid structure. ©[5] . . . 4
1.2 Master-slave architecture . . . 6
1.3 Distributed connected architecture . . . 7

2.1 IEEE 14-bus System . . . 31
2.2 Active power of 14 bus . . . 32
2.3 Residual for the four approaches . . . 32
2.4 Marginal cost for all buses in IEEE 14-bus a) DLMC approach b) EXTRAC approach c) Consensus + Innovation approach d)Push-Sum approach . . . 33
2.5 Residual for the topology which is added by some links . . . 33
2.6 IEEE 39-bus . . . 34
2.7 Residual by number of iterations . . . 35
2.8 Residual by runtime . . . 35
2.9 Residual for IEEE 39-bus . . . 37
2.10 Marginal cost for all buses in IEEE 39-bus a) DLMC approach b) EXTRAC approach c) Consensus + Innovation approach d)Push-Sum approach . . . 37
2.11 Active power of 39 bus . . . 38
2.12 Marginal cost a)DLMC approach b)EXTRAC approach . . . 39
2.13 Charging/ Discharging Storage . . . 39
2.14 Power for Generations and Storage . . . 40
2.15 Consumption by Loads . . . 40
2.16 Residual for DLMC and EXTRAC approaches . . . 40

3.1 Strongly-connected digraphs tested in our experiments. . . . 67
3.2 Distributed Regression with Huber Loss . . . 68
3.3 Distributed Logistic Regression . . . 69
3.4 Distributed logistic regression problem ($n = 200$, $p = 15$) over 20 randomly directed graphs with a high connectivity ratio $\phi = 0.15$. Solid curves represent the average and the shaded region represents the range statistics. . . . 71
3.5 Distributed logistic regression problem ($n = 200$, $p = 15$) over 20 randomly directed graphs with a low connectivity ratio $\phi = 0.015$. Solid curves represent the average and the shaded region represents the range statistics. . . . 72
3.6 Distributed logistic regression problem ($n = 25$, $p = 301$) over 20 randomly directed graphs with a low connectivity ratio $\phi = 0.1$. Solid curves represent the average and the shaded region represents the range statistics. . . . 73

A.1 Distributed Regression with Huber Loss for Fig. 3.1a . . . 78
A.2 Distributed Regression with Huber Loss for Fig. 3.1b . . . 78

A.3 Distributed Regression with Huber Loss for Fig. 3.1c 79
A.4 Distributed Regression with Huber Loss for Fig. 3.1d 79
A.5 Distributed Regression with Huber Loss for Fig. 3.1e 80
A.6 Distributed Regression with Huber Loss for Fig. 3.1f 80

List of Tables

2.1 Generator Parameters (mu = Monetary Units) 31
2.2 Load Parameters (mu = Monetary Units) 31
2.3 Storage Parameters (mu = Monetary Units) 31
2.4 Load Parameters (mu = Monetary Units) 36
2.5 Generator Parameters (mu = Monetary Units) 36
2.6 Storage Parameters (mu = Monetary Units) 36

3.1 Comparison of methods for directed graphs in terms of storage and communication requirements. 53

List of Algorithms

1 Distributed algorithm (DLMC) 28
2 Fast row-stochastic decentralized algorithm (FRSD) 46

Glossary

EDP	Economic Dispatch Problem
ES	Energy Storage
ERM	Empirical Risk Minimization
GT	Gradient Tracking
ADMM	Alternating Direction Method of Multipliers
FRSD	Fast row-stochastic decentralized algorithm
DLM	Decentralized linearized alternating direction method
EXTRA	Exact first order algorithm
PG-EXTRA	Proximal gradient exact first-order algorithm

Chapter 1

Introduction

Over the past two decades, there has been a significant interest in optimization methods, which aim to find the optimal decision among a large number of alternatives. In simple terms, an optimization method can be described as the process of finding under a given set, the element (or elements) of that set that minimize (or maximize) a given objective function. Additionally, the decision variable may be subject to constraints. In general, optimization methods are iterative starting with some initial condition, and under adequate conditions, they should converge to an optimal solution as the number of iterations grows. A particular class of optimization problems that arise in many significant practical problems in a variety of fields, such as power systems, wireless networks, cyber-physical systems, and machine learning processes to name a few, is when the minimization occurs over a sum of (many) individual objective functions, and there is a significant amount of data to deal with it. In this case, it is challenging to address this class of problems using traditional approaches, which assume that all problem data are centralized and accessible to a central processor. Centralized methods, where datasets collected by devices are uploaded to a data center and analyzed centrally, are in many cases not practical due to the complexity of the network, the amount of data to address, and moreover it may not obey to privacy requirements. To address these issues, especially on large-scale networks, distributed optimization algorithms have been developed where each agent (node) in the network typically has access to its own private local data and can only communicate directly with its neighbors. This thesis contributes to this field with the aim of developing

distributed optimization approaches for minimizing the sum of individual costs associated with individual elements over undirected or directed large-scale networks [1].

1.1 Context

Consider a general unconstrained optimization problem of the form:

$$\min_{x\in\mathbb{R}^p} f(\boldsymbol{x}), \tag{1.1}$$

where the vector $\boldsymbol{x} \in \mathbb{R}^p$ is the decision variable, and it is assumed that the objective function $f : \mathbb{R}^p \to \mathbb{R}$ is continuously differentiable and (1.1) is well-posed, that is, there is an optimal solution $x^* \in \mathbb{R}^p$ such that $f(x^*) = f^* = \inf_{x\in\mathbb{R}^p} f(x)$ is finite. In this setup, a simple approach to solve (1.1) is to apply the gradient descent algorithm, which starts at a given initial point $x(0) \in \mathbb{R}^p$ and generates a sequence according to

$$\boldsymbol{x}(k+1) = \boldsymbol{x}(k) - \alpha(k)\nabla f(\boldsymbol{x}(k)), \quad k = 0, 1, ... \tag{1.2}$$

where the scalar $\alpha(k) > 0$ is the step-size coefficient that can be dependent on the iteration k, and $\nabla f = \frac{\partial f}{\partial x}$ denotes the gradient of f with respect to x. At this point, it is important to stress that it is unclear whether the sequence $x(k)$ can converge or even how large the step-size must be to guarantee convergence.

Therefore, it is expected to assume that some certain regularity assumptions regarding the objective function should be applied.. First, it is necessary to impose a restriction on the magnitude of the gradient in order to ensure that there are step sizes such that gradient descent converges, which in turn will have a direct impact on the convergence.

Definition 1.1. The function f is L-smooth, i.e., it is differentiable with a Lipschitz gradient, if there exists a positive constant L such that

$$\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{x}')\| \leq L\|\boldsymbol{x} - \boldsymbol{x}'\|, \quad \forall\, \boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^p. \tag{1.3}$$

Next, we consider that f is convex, which will ensure that all local minima are global minima.

Definition 1.2. A differentiable function f is called to be convex when the following holds

$$f(\boldsymbol{x}) \geq f(\boldsymbol{x}') + \nabla f(\boldsymbol{x}')(\boldsymbol{x} - \boldsymbol{x}'), \quad \forall\, \boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^p. \tag{1.4}$$

[1] A graph (or network) whose edges are all bidirectional is known as an *undirected graph*, whereas a *directed graph* has at least one edge pointing only one direction.

Next, we present an additional restriction that can significantly improve the convergence rate of the gradient descent algorithm.

Definition 1.3. A differentiable function f is μ-strongly convex, if there is a positive constant μ such that for all $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^p$,

$$f(\boldsymbol{x}') \geq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^\top (\boldsymbol{x}' - \boldsymbol{x}) + \frac{\mu}{2} \parallel \boldsymbol{x}' - \boldsymbol{x} \parallel^2. \tag{1.5}$$

We can now state the following result whose proof can be found, for instance, in [1, 2].

Proposition 1.4. *Consider the unconstrained optimization problem (1.1) and let the objective function f be continuously differentiable, $L-$smooth, and $\mu-$strongly convex. Then, the optimal solution is unique and for any initial condition $x(0) \in R^p$, the gradient descent method (1.2) generates a sequence that converges to the optimal. Moreover, the convergence rate can be improved to a linear rate.*

There are some accelerated first-order methods[1] that improve convergence rate based on the same prior assumptions. The most popular is the Nesterov acceleration [1], which is written in the following form

$$\boldsymbol{x}(k+1) = \boldsymbol{y}(k) - \frac{1}{L} \nabla f(\boldsymbol{y}(k)), \tag{1.6}$$

$$\boldsymbol{y}(k+1) = \boldsymbol{x}(k+1) + \frac{1 - \sqrt{\kappa_f}}{1 + \sqrt{\kappa_f}} \Big(x(k+1) - x(k) \Big), \tag{1.7}$$

where $x(0)$ and $y(0)$ are arbitrary starting points, and $\kappa_f = \frac{L}{\mu}$. Heavy Ball Acceleration is another accelerated first-order method proposed by the author in [3], but it has a lower convergence rate than Nesterov acceleration in general. The accelerated methods improve the convergence rate of first-order methods by a significant amount; for details, see [4].

In the rest of the thesis, we consider a specific objective function which is a finite sum problem

$$\min_{\boldsymbol{x} \in \mathbb{R}^p} f(\boldsymbol{x}) \triangleq \frac{1}{n} \sum_{i=1}^{n} f_i(\boldsymbol{x}) \tag{1.8}$$

where each $f_i : \mathbb{R}^p \to \mathbb{R}$ is differentiable and $L-$smooth. One direct approach is to solve the problem (1.8) by directly applying the gradient method (1.2) using $\nabla f(\boldsymbol{x}) = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(\boldsymbol{x})$. However, in this thesis, the goal is to find a minimizer of the objective function (1.8) in a distributed way, which will be discussed in the following sections. Before

[1] First-Order methods are iterative algorithms that only utilize the objective function and the gradient/sub-gradient information.

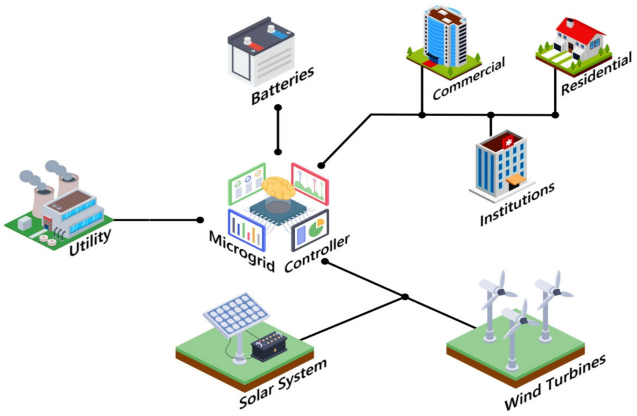


FIGURE 1.1: Microgrid structure. ©[5]

this, we first show that the problem (1.8) is quite prevalent among a diversity of applications through some illustrative examples.

1.1.1 Examples

In the sequel, we introduce a set of applications that can be formulated as an optimization problem with objective functions being composed of a finite sum of local functions (1.8).

1.1.1.1 Economic Dispatch Problem

In power systems, a typical problem termed an Economic Dispatch Problem (EDP) [6, 7], as illustrated in Figure 1.1, consists in finding the operation of generation facilities, that is, obtaining the best power output combination of all generators, to produce energy at the lowest operating cost while satisfying several constraints including the total power demand. It turns out, that this problem can be cast in the form (1.8). More precisely, in this thesis, the EDP is formulated as follows:

$$(P): \quad \min_{p_i} \sum_{i=1}^{n} C_i(p_i) \tag{1.9}$$

$$s.t. \quad \sum_{i=1}^{n} p_i = D \tag{1.10}$$

$$p_i^{min} \leq p_i \leq p_i^{max} \tag{1.11}$$

where n is the number of generators, the decision variable p_i is the power generated by generator i, $C_i(p_i)$ is the cost function, D is the total demand, and p_i^{min} and p_i^{max} are the

upper and lower power limits of the generators. We will provide a fast algorithm for solving this formulation in Chapter 2.

1.1.1.2 Empirical Risk Minimization

In machine learning, Empirical Risk Minimization (ERM) [8, 9], covers a family of learning algorithms. Given a model $b = h_x(M)$, where $b \in \mathbb{R}^{p_b}$, $M \in \mathbb{R}^{p_M}$, and h is a given map parameterized by $x \in \mathbb{R}^p$, a training dataset composed by pairs of input-output samples, and a loss function L, where $L(h(M,x),b)$ captures the error made by the model h when predicting a given input-output sample (M,b), the goal is to obtain the model (by selecting conveniently x) that achieves the lowest error on the training dataset. If we define the global objective function f as the average of the component functions f_i, and each node i has access to a set of data samples (M_i, b_i), then the empirical risk minimization is written as follows:

$$\min_x \quad \frac{1}{n}\sum_{i=1}^{n} f_i(x) = \min_x \quad \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(h(M_i,x),b_i)$$

The ERM formulation addresses a multitude of significant machine learning problems, for example, by letting $\mathcal{L}(h(M,x),b) = log(1+exp(-b.M^\top x))$, we can obtain the logistic regression problem, or when $\mathcal{L}(h(M,x),b) = \frac{1}{2}(b - M^\top x)^2$, we have the least-squares regression problem, which we consider in the numerical experiments in Chapter 3.

1.2 Distributed Architectures

In this section, we discuss how the gradient descent method may be implemented in a distributed manner. To find a minimizer of $f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{x})$ in the distributed form, we consider that there is an underline communication network which is represented for simplicity as an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} \triangleq \{1, 2, \ldots, n\}$ is the set of nodes (agents), and $\mathcal{E}$ is the set of directed communication links between the nodes. The cost function $f_i : \mathbb{R}^p \to \mathbb{R}$ represents the cost function of agent i, which is only known to agent i, and x is a global decision vector that the agents are trying to determine with local information. Each network agent generates and maintains estimates of the optimal decision vector based on its own cost function f_i, and exchanges these estimates with other

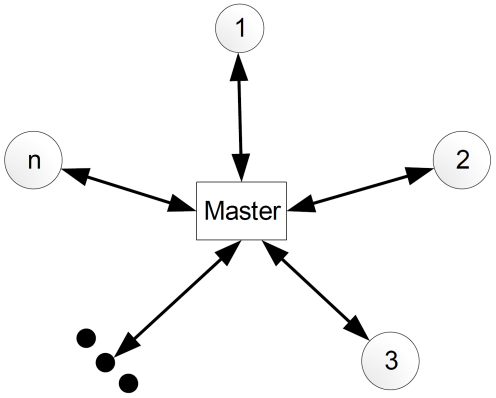


FIGURE 1.2: Master-slave architecture

agents. Eventually, the optimal global solution is achieved using only local communications and computations. Given this setup, we will discuss two widely used distributed architectures: 1) master-slave architecture, and 2) distributed connected architecture.

1.2.1 Master-slave architecture

In a master-slave architecture, as depicted in Figure 1.2, one node performs as the master and maintains the optimization variable. By starting at some initial value $x(0)$, at each iteration, the master node sends the current parameter $x(k)$ to each agent, followed by each agent sending $\nabla f_i(x(k))$ to the master. Before beginning a new iteration, the master node calculates its averages and updates each agent with the following configuration:

$$\boldsymbol{x}(k+1) = \boldsymbol{x}(k) - \alpha(k)\left(\frac{1}{n}\sum_{i=1}^{n}\nabla f_i(\boldsymbol{x}(k))\right) \qquad k = 0, 1, ... \tag{1.12}$$

where step-size $\alpha(k)$ is positive scalar. The implementation of the master-slave architecture is simple and straightforward; however, there are drawbacks that make this architecture difficult or impossible to use in practice. First, if the number of nodes N is exceedingly high, it may be impracticable for the master to communicate with each node due to bandwidth limitations in the communication resources. In addition, the computation of $\sum_{i=1}^{n}\nabla f_i(\boldsymbol{x}(k))$ may become quite large and may be impossible to perform on a single computer. Second, the entire network may fail if the master node becomes a bottleneck and fails. Therefore, it may not be practical to have a master node that communicates with all agents in numerous applications.

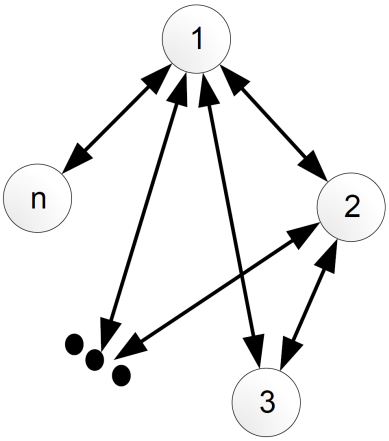


FIGURE 1.3: Distributed connected architecture

1.2.2 Distributed connected architecture

In this case, there is no master node, and therefore the issues of the master-slave architecture do not appear, resulting in a distributed connected architecture where each node communicates directly with some but not necessarily all of the other nodes, see Figure 1.3. More precisely, each node $i = 1, ..., n$ maintain the optimization variable $x_i(k)$, and a list of nodes that sends information to node i, which is denoted by

$$\mathcal{N}_i \triangleq \{j \in \mathcal{V} : \ (j,i) \in \mathcal{E}\} \cup \{i\} \tag{1.13}$$

If we use a similar approach in master-slave architecture, and assuming that $x(0)$ is initialized to the same value on all nodes, each node i computes its local gradient $\nabla f_i(x_i(k))$ and sends it and receive $\nabla f_j(x_j(k))$ from its neighbors $j \in \mathcal{N}_i$. Once a node i has received the gradients from its neighbors, it can average them and update a new value for $x_i(k+1)$ by gradient descent in a decentralized manner as follows

$$\boldsymbol{x_i}(k+1) = \boldsymbol{x_i}(k) - \alpha(k)\left(\frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i}\nabla f_j(\boldsymbol{x_j}(k))\right) \qquad k = 0, 1, ... \tag{1.14}$$

where $|\mathcal{N}_i|$ is the size of $\mathcal{N}_i$. In this method, the agent i only averages the gradients received from its neighbors, and the communication topology network among agents may be static or it may vary from iteration to iteration.

Note that in the distributed connected architecture, if each node connects to every other node in the network, then we have a **full-connected architecture**, and node i is

updated according to:

$$\boldsymbol{x_i}(k+1) = \boldsymbol{x_i}(k) - \alpha(k)\left(\frac{1}{n}\sum_{j=1}^{n}\nabla f_j(\boldsymbol{x_j}(k))\right) \qquad k = 0, 1, ... \tag{1.15}$$

where each node acts like a master in the master-slave architecture. In this case, it can be shown that the values $x_i(k)$ are identical at all nodes and the convergence results for the centralized case can be straightforwardly applied. However, in this case, as in the master-slave architecture, the fully-connected architecture suffers from high computation costs.

For the distributed connected architecture described in 1.14, there is a significant difference when compared with the gradient descent method, as only the average of a subset of the gradient is used. Note that

$$\left(\frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i}\nabla f_j(\boldsymbol{x_j}(k))\right) \neq \nabla f(\boldsymbol{x_i}(k)) \tag{1.16}$$

Therefore, one needs to investigate under which conditions the distributed connected method is guaranteed to converge to an optimal solution. The following sections address this point, where it will be clear that the communications network between agents has a major impact on the convergence rate of the distributed optimization methods.

1.3 Distributed Optimization Methods

In this section, we outline the distributed optimization methods over undirected and directed graphs. The goal is to minimize the average objective function

$$\min_x f(x) = \min_x \frac{1}{n}\sum_{i=1}^{n} f_i(x) \tag{1.17}$$

in a distributed way where n is the total number of agents. In this setup, each agent i does not share its local function $f_i : \mathbb{R}^p \to \mathbb{R}$ with other agents, and the information is not accessible to the general public. Moreover, every agent is aware of its graph's local connectivity. In a nutshell, we must find the minimum of the objective function by taking into account the cooperation between agents, despite the agents' limited knowledge and connectivity.

1.3.1 Distributed optimization methods for undirected graphs

In this section, we consider that the nodes are connected over undirected graphs, in which node i receives information from node j if and only if node j also receives information from node i. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a given undirected graph, where $\mathcal{V} \triangleq \{1, 2, \ldots, n\}$ is the set of nodes (agents), and $\mathcal{E}$ is the set of edges. The aim is to solve the problem (1.17) in a distributed way taking into account that each agent i does not share its data and its loss functions f_i but it intends to collaborate and share some estimated value with only its neighborhood $\mathcal{N}_i$, defined in (1.13), which includes the list of nodes connected with i for $i = 1, ..., n$ directly $-$ and agent i itself.

To this effect, we start to note that there is an alternative formulation of the problem (1.17), in which each f_i depends on its own variable x_i; however, these variables are coupled through the constraint $x_i = x_j$ for all $(i, j) \in \mathcal{E}$, that is,

$$\begin{aligned} \min_{\{x_i\}_{i=1}^n} \quad & \frac{1}{n} \sum_{i=1}^{n} f_i(x_i) \\ s.t. \quad & x_i = x_j \qquad \forall (i, j) \in \mathcal{E} \end{aligned} \tag{1.18}$$

This reformulation is derived from (1.17), by considering x_i for each node i, and requiring x_i to be identical to x for all agents. Therefore, when the graph is connected, the previous equations $x_i = x$ for $i = 1, ...n$ is equivalent to the constraint $x_i = x_j$ for all $(i, j) \in \mathcal{E}$ in the problem (1.18) which enforces all local variables be equal $(x_i = x_j, \forall i, j \in \mathcal{V})$. Hence, both problems (1.17) and (1.18) have the same objective.

Note that the objective function of the problem (1.18) is separable, although the variables x_i are connected to each other through the constraint (1.18). Thus, the overall problem can be solved in a distributed way by allowing each agent to know their network neighbors so that they can apply a consensus mechanism, and then solve its own local portion.

1.3.1.1 The consensus distributed method

We now present the consensus algorithm used to design a distributed optimization method for solving the problem (1.18) in a distributed way, in which the agents determine the vector x collectively while satisfying the constraint (1.18). The idea is that the agents must asymptotically agree on a decision vector while exchanging information with only their

neighbors on the graph[10–12]. More precisely, each agent i has a variable $x_i(k)$ at iteration k, that is sent to each agent j, which is in neighborhood $\mathcal{N}_i$, and receives x_j from them. Then, every agent i performs the update step

$$x_i(k+1) = \sum_{i=1}^{n} w_{ij} x_j(k) \tag{1.19}$$

where $w_{ij} > 0$, for $j \in \mathcal{N}_i$, otherwise $w_{ij} = 0, j \neq i, j \notin \mathcal{N}_i$. To guarantee the convergence of the consensus approach, we first need to introduce some basic concepts.

Strongly connected graph: a graph is considered to be a *strongly connected graph*: if every node in the network can be reached from every other node in the graph.

Doubly/Column/Row-stochastic matrix: A square matrix of nonnegative real numbers $W \in \mathbb{R}^{n\times n}$ is said to be a *doubly-stochastic matrix* if each row and column sum to one. In that case, both the left and right eigenvectors corresponding to eigenvalue 1 are an all-one-vector, i.e.,

$$\mathbf{1}_n^\top W = \mathbf{1}_n^\top, \qquad W\mathbf{1}_n = \mathbf{1}_n \tag{1.20}$$

where $W \in \mathbb{R}^{n\times n}$ is a doubly-stochastic matrix and $\mathbf{1}_n \in \mathbb{R}^n$ is the $n-$dimensional all-one vector. In addition, a *column-stochastic matrix* $\overline{B} \in \mathbb{R}^{n\times n}$, is defined as a real square matrix satisfying the criterion that the sum of each column is 1. In this case, we have

$$\mathbf{1}_n^\top \overline{B} = \mathbf{1}_n^\top, \qquad \overline{B}\boldsymbol{\pi}_b = \boldsymbol{\pi}_b \tag{1.21}$$

where $\boldsymbol{\pi}_b \in \mathbb{R}^n$ is not necessary equals to $\mathbf{1}_n$. Similarly, a real square stochastic matrix $\overline{R} \in \mathbb{R}^{n\times n}$ is known as *row-stochastic matrix*, if the sum of each row is 1, and in this case

$$\boldsymbol{\pi}^\top \overline{R} = \boldsymbol{\pi}^\top, \qquad \overline{R}\mathbf{1}_n = \mathbf{1}_n \tag{1.22}$$

where $\boldsymbol{\pi} \in \mathbb{R}^n$.

Returning now to the consensus update step (1.19), note that it is required to consider $W \in \mathbb{R}^{n\times n}$ with entries w_{ij} as a row-stochastic matrix. More specifically, if the graph is undirected as it is here, then matrix W should be a doubly stochastic matrix. If W is a row stochastic matrix and $w_{ij} = 0, j \neq i, j \notin \mathcal{N}_i$, the consensus algorithm (1.19) converge to a limit point, called consensus point, and we denote it by $\hat{x}$ [10–12]. If we denote vector π as the unique left eigenvector of matrix W corresponding to the eigenvalue $\lambda = 1$, then we have

$$\hat{\mathrm{x}} = \mathbf{1}_n \pi \mathrm{x}(0) \tag{1.23}$$

where $\hat{x}= [\hat{x}, ..., \hat{x}]$, and $x(0) = [x_1(0), ..., x_n(0)]$ is the vector of the initial values . When the graph is undirected, π is the n-dimensional vector of all entries $\frac{1}{n}$, hence, the consensus method (1.19) converge to $\hat{x} = \frac{1}{n}\sum_{i=1}^{n} x_i(0)$ for each agent i.

We now explain how to solve the problem (1.18) using a distributed algorithm. On each iteration k, each agent i retains two variables, $x_i(k)$ and $y_i(k)$, and sends $x_i(k)$ to its neighbors $j \in \mathcal{N}_i$; consequently, each agent receives $x_j(k)$ from the agent $j \in \mathcal{N}_i$, then by initializing $x_i(0)$ for all i, we have the following steps for each agent i:

$$\begin{aligned} y_i(k) &= \sum_{j=1}^{n} w_{ij} x_j(k) \\ x_i(k+1) &= y_i(k) - \alpha(k)\nabla f_i(y_i(k)) \end{aligned} \tag{1.24}$$

where $\alpha(k) > 0$ is a positive step-size, and the weights w_{ij} satisfies the consensus assumptions, that is, W is a double stochastic matrix, and $w_{ij} = 0$ for $j \notin \mathcal{N}_i$. Since each agent performs its own updates using the gradient of its own private function, the algorithm in (1.24) is a distributed method. Moreover, the information is exchanged only between the neighbors.

The iterative method (1.24) that employs a consensus algorithm with stochastic matrices is also known as a weighted-averaging method because, if we take the average of equation (1.24), we obtain:

$$\begin{aligned} \frac{1}{n}\sum_{i=1}^{n} x_i(k+1) &= \frac{1}{n}\sum_{i=1}^{n} y_i(k) - \alpha(k)\frac{1}{n}\sum_{i=1}^{n}\nabla f_i(y_i(k)) \\ &= \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} x_j(k) - \alpha(k)\frac{1}{n}\sum_{i=1}^{n}\nabla f_i(y_i(k)) \end{aligned} \tag{1.25}$$

Since W is a double stochastic matrix, we have $\sum_{i=1}^{n} w_{ij} = 1$, then the first term (1.25) can be simplified to $\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}x_j(k) = \frac{1}{n}\sum_{j=1}^{n}\left(\sum_{i=1}^{n} w_{ij}\right)x_j(k) = \frac{1}{n}\sum_{j=1}^{n} x_j(k)$, hence, we have

$$\frac{1}{n}\sum_{i=1}^{n} x_i(k+1) = \frac{1}{n}\sum_{j=1}^{n} x_j(k) - \alpha(k)\frac{1}{n}\sum_{i=1}^{n}\nabla f_i(y_i(k)) \tag{1.26}$$

Denoting

$$\bar{x}(k) = \frac{1}{n}\sum_{i=1}^{n} x_i(k) \tag{1.27}$$

then, we can rewrite (1.26) as follows:

$$\bar{x}(k+1) = \bar{x}(k) - \alpha(k)\frac{1}{n}\sum_{i=1}^{n}\nabla f_i(y_i(k)) \tag{1.28}$$

As can be seen, the only difference between the term (1.28) and a centralized gradient method for solving the problem (1.18) is that the term (1.28) uses $\nabla f_i(y_i(k))$ rather than $\nabla f_i(\bar{x}(k))$. Hence, by adding an error $\epsilon(k)$ to the equation (1.28), we can have the centralized gradient method as follows

$$\bar{x}(k+1) = \bar{x}(k) - \alpha(k)\frac{1}{n}\sum_{i=1}^{n}\nabla f_i(\bar{x}(k)) + \epsilon(k) \tag{1.29}$$

where

$$\epsilon(k) = \frac{\alpha(k)}{n}\Big(\sum_{i=1}^{n}\nabla f_i(y_i(k)) - \sum_{i=1}^{n}\nabla f_i(\bar{x}(k))\Big) \tag{1.30}$$

Based on some additional assumptions on the objective function and step-size $\alpha(k)$, it can be proved that the error $\epsilon(k)$ can be small, and the average iteration $\bar{x}(k+1)$ in (1.28) follows the centralized gradient method, i.e. the average iteration $\bar{x}(k)$ converges to the optimal solution and $\|x_i(k) - \bar{x}(k)\|$ goes to zero for each agent i. The consensus method, the matrix W and the gradient of the objective function have an important role in forcing each agent to converge to the same optimal solution x^*, for more details see [13–17]. In the paper [16], the authors utilized a subgradient of the objective function instead of the gradient as follows:

$$x_i(k+1) = \sum_{j=1}^{n} w_{ij}x_j(k) - \alpha(k)g_i(k) \tag{1.31}$$

where $g_i(k)$ is subgradient of $f_i(x_i(k))$ which is convex and possibly non-differentiable.

For the distributed method (1.24), convergence is guaranteed if α is a diminishing step-size satisfying the following conditions:

$$\sum_{k=0}^{\infty}\alpha(k) = \infty, \qquad \sum_{k=0}^{\infty}\alpha^2(k) < \infty \tag{1.32}$$

Since there is a difference between the convergence rates, the distributed algorithm (1.24), which solves convex optimization problems, has a slower convergence rate compared to a centralized method with diminishing step size; the distributed method has the converge rate of the order of $\mathcal{O}(\frac{\log(k)}{k})$, but the centralized one converges in the order of $\mathcal{O}(\frac{1}{k})$.

The order of equations in the distributed method in (1.24) can be exchanged as follows:

$$\begin{aligned} y_i(k) &= x_i(k) - \alpha(k)\nabla f_i(y_i(k)) \\ x_i(k+1) &= \sum_{j=1}^{n} w_{ij} x_j(k) \end{aligned} \tag{1.33}$$

In this case, the algorithm in (1.33) is known as adapt-then-combine because it first adjusts the gradient step and then combines the local information, as contrasted to the algorithm in (1.24), which is referred to as combine-then-adapt [17].

The distributed methods in (1.24) and (1.33) solve the problem (1.18) in a fully distributed way, and the use of diminishing step-size has advantages over constant step-size in certain circumstances, such as when solving problems involving noises caused by link failure. However, slow convergence or lack of linear convergence [1] is an evident disadvantage of the algorithm with a diminishing step-size. In [18], algorithms based on dual averaging of subgradients are studied for solving (1.18). In [19], under the assumption that the gradients are bounded and Lipschitz, an improved convergence rate is obtained by employing Nesterov acceleration. In the following section, we present discrete-time algorithms with linear convergence rates by eliminating the diminishing step-size and substituting it with a constant step-size.

1.3.1.2 The distributed methods with a linear convergence

In this section, we present a variety of distributed algorithms with constant step-size that converge linearly to the optimal solution.

EXTRA [20]. The authors proposed EXTRA (exact first order algorithm), a distributed optimization algorithm with a constant step size that converges linearly to the global optimum when the global objective function is strongly convex and has Lipschitz continuous gradients. EXTRA contains two consecutive steps with constant step-size α and two different matrices W and $\tilde{W}$ which is updated as follows

$$x(k+2) = Wx(k+1) - \alpha\nabla f(x(k+1)) \tag{1.34}$$

$$x(k+1) = \tilde{W}x(k) - \alpha\nabla f(x(k)) \tag{1.35}$$

[1] Assume that a sequence $x(k)$ converges to x^* in a certain norm $\|\cdot\|$. We say that the sequence converges linearly if $0 < \lambda < 1$ exist such that $\lim_{k\to\infty} \frac{\|x(k+1)-x^*\|}{\|x(k)-x^*\|} \leq \lambda$.

where

$$\tilde{W} = \frac{I_n + W}{2} \tag{1.36}$$

and W is double stochastic matrix and its entries w_{ij} might be chosen as a Metropolis matrix:

$$w_{ij} = \begin{cases} \frac{1}{\max\{d_j, d_i\}+\epsilon} & \text{if } (i,j) \in \mathcal{E}. \\ 0 & \text{if } (i,j) \notin \mathcal{E} \quad i \neq j \\ 1 - \sum\limits_{k \in \mathcal{V}} w_{ik} & \text{if } i = j. \end{cases} \tag{1.37}$$

where $\epsilon > 0$ and d_i is the degree of agent i, see [20].

Notice that by taking the difference of the two steps (1.34) and (1.35), we have the following second-order recursion:

$$x(k+2) = (I_n + W)x(k+1) - \tilde{W}x(k) - \alpha\Big(\nabla f(x(k+1)) - \nabla f(x(k))\Big) \tag{1.38}$$

where $x(0)$ is the initial condition and $x(1)$ is given as

$$x(1) = Wx(0) - \alpha\nabla f(x(0)) \tag{1.39}$$

Compared to the distributed algorithm in (1.24), which uses the gradient at the previous iteration, EXTRA utilizes the difference of the gradients at the previous two iterations. The gradient difference scheme gets rid of the steady-state error that occurs in the algorithm in (1.24), and when the objective function is strongly convex, EXTRA reaches a geometric convergence; when the objective function is only convex, the convergence rate is $\mathcal{O}(\frac{1}{k})$. In [21], the authors interpreted EXTRA as a primal-dual algorithm, and a proximal gradient exact first-order algorithm (PG-EXTRA) has been proposed when the local objective functions have the nonsmooth composite convex [22]. In [23], a distributed augmented Lagrangian algorithm along with a double-loop structure was proposed. Furthermore, the authors in [24] developed a distributed proximal gradient algorithm with only a single-loop; see also [25, 26] for more proximal algorithms for composite optimization.

Gradient Tracking. Unlike EXTRA, which corrects the fixed point of the iterations directly, gradient tracking methods try to manage a way to apply the following iteration:

$$x(k+1) = Wx(k) - \frac{\alpha}{n} 1_n 1_n^\top \nabla f(x(k)) \tag{1.40}$$

Here, the term $1_n 1_n^\top \nabla f(x(k))$ cannot be computed in a distributed way, hence, instead of the average gradient, gradient tracking methods utilize a variable y as follows:

$$y(k+1) = Wy(k) + \nabla f(x(k+1)) - \nabla f(x(k)) \tag{1.41}$$

In [27, 28], the authors independently introduced the AUG-DGM algorithm and the NEXT algorithm for the first time using the gradient tracking technique. In [29], the authors address the case of time-varying networks, by combining the distributed inexact gradient method and the gradient tracking technique. The iterative equations, called the Diging method, are of the form

$$x(k+1) = Wx(k) - \alpha y(k) \tag{1.42}$$

$$y(k+1) = Wy(k) + \nabla f(x(k+1)) - \nabla f(x(k)) \tag{1.43}$$

The Diging method is initialized with a arbitrary vector $x(0)$ and $y(0) = \nabla f(x(0))$. In (1.42), $1_n 1_n^\top \nabla f(x(k))$ in (1.40) is replaced by the variable y, and a track of average gradient is used in (1.43) by employing dynamic average consensus.

When the local objective functions are smooth and strongly convex, the Diging algorithm for undirected graphs achieves linear convergence. In [29], the authors provide an explanation of a primal-dual interpretation of Diging based on the augmented Lagrangian function. By selecting the appropriate two mixing matrices, they also demonstrated how Diging was derived from EXTRA.

In [30], the authors extended the Diging method under uncoordinated step-sizes with a Adapt-Then-Combine structure of the DIGing algorithm, which is given as

$$x(k+1) = W\Big(x(k) - \alpha(k)y(k)\Big) \tag{1.44}$$

$$y(k+1) = W\Big(y(k) + \nabla f(x(k+1)) - \nabla f(x(k))\Big) \tag{1.45}$$

where $\alpha(k)$ is the uncoordinated step-size. In addition, the works in [31, 32] improved convergence rates over an undirected graph.

The distributed methods over undirected networks require the use of a doubly stochastic matrix W to solve the problem (1.17). Due to nonuniform communication among agents or unexpected loss of communication between the agents, the information exchange may be unidirectional and the network topology may be (time-varying) directed; hence, ensuring a structure of a double stochastic matrix is not an easy task [33],

and therefore the previous algorithms in section (1.3.1) can not be directly implemented. To handle this issue, we next present distributed methods, which work without a double stochastic matrix.

1.3.2 Distributed optimization methods for directed graphs

In this section, we briefly present two classes of the most relevant existing discrete-time distributed methods over networks in which the communications among agents are (potentially time-varying) directed. In Chapter 3, we will present the distributed optimization methods for directed graphs in greater detail.

Gradient–push method. We explained in the previous section that distributed methods for undirected graphs require a doubly stochastic matrix. However, the implementation of a doubly stochastic matrix is computationally expensive when graphs are directed. In [34], a different consensus algorithm, called the *push-sum method*, that utilized a column stochastic matrix was introduced for the first time to help circumvent this issue. In the push-sum method, each agent simultaneously implements two linear iterations with a column stochastic matrix, and the ratio between the two states converges to the initial average for strongly connected directed graphs. The push-sum is formulated as follows,

$$x_i(k+1) = \sum_{j \in \mathcal{N}_i^{in}} b_{ij} x_j(k) \tag{1.46}$$

$$y_i(k+1) = \sum_{j \in \mathcal{N}_i^{in}} b_{ij} y_j(k) \tag{1.47}$$

$$v_i(k+1) = \frac{x_i(k+1)}{y_i(k+1)} \tag{1.48}$$

where $\mathcal{N}_i^{in}$ is in-neighbors of agent i, that is,

$$\mathcal{N}_i^{in} = \{j \in \mathcal{V} : \ (j,i) \in \mathcal{E}\} \cup \{i\} \tag{1.49}$$

and b_{ij} are the entries of the a column stochastic matrix B, which have the following properties:

$$\sum_{i \in \mathcal{N}_j^{out}} b_{ij} = 1, \quad b_{ij} > 0 \tag{1.50}$$

$$b_{ij} = 0 \qquad i \notin \mathcal{N}_j^{out} \tag{1.51}$$

where $\mathcal{N}_i^{out}$ is out-neighbors of agent i, that is,

$$\mathcal{N}_i^{out} = \{j \in \mathcal{V} : \ (i,j) \in \mathcal{E}\} \cup \{i\} \tag{1.52}$$

In (1.46) and (1.47), the variables x, y are initialized with an arbitrary vector $x(0)$ and a scalar $y(0) = 1$, and in (1.48), the ratio of x and y, reaches a consensus at each agent when the directed graph is strongly connected. The algorithm, also called *ratio consensus*, has been developed for time-varying graphs in [35–38]. The push-sum method and dual averaging distributed method are used in [39] to address a distributed optimization method for fixed-directed graphs. In addition, the push-sum method for general cases over time-varying directed graphs has been developed in [40] to solve the problem (1.17). It is referred to as the *gradient–push method* and its formulation is as follows:

$$\begin{aligned}
z_i(k+1) &= \sum_{j \in \mathcal{N}_i^{in}} b_{ij} x_j(k) \\
y_i(k+1) &= \sum_{j \in \mathcal{N}_i^{in}} b_{ij} y_j(k) \\
v_i(k+1) &= \frac{z_i(k+1)}{y_i(k+1)} \\
x_i(k+1) &= z_i(k+1) - \alpha(k) g_i(k+1)
\end{aligned} \tag{1.53}$$

where $\alpha(k)$ is a diminishing step-size and $g_i(k+1)$ is subgradient of $f_i(v_i(k+1))$.

The gradient–push method, like the undirected distributed methods which used a diminishing step-size, has a slow convergence rate. In addition, when the step-size is constant, the algorithm (1.53) fails to converge. Next, we describe the algorithms based on gradient tracking with a constant step-size that have a better convergence rate.

Gradient Tracking. Using the gradient tracking method (1.42) and (1.43) and the gradient push-sum method (1.53), *Push-Diging*, a gradient-based distributed optimization method, was proposed in [29],

$$\begin{aligned}
z_i(k+1) &= \sum_{j \in \mathcal{N}_i^{in}} b_{ij} \Big(z_j(k) - \alpha y_j \Big) \\
v_i(k+1) &= \sum_{j \in \mathcal{N}_i^{in}} b_{ij} v_j(k) \\
x_i(k+1) &= \frac{z_i(k+1)}{v_i(k+1)} \\
y_i(k+1) &= \sum_{j \in \mathcal{N}_i^{in}} b_{ij} y_i(k) + \nabla f_i(x_i(k+1)) - \nabla f_i(x_i(k))
\end{aligned} \tag{1.54}$$

where $B = [b_{ij}]$ is a column stochastic matrix, and the objective function f_i is strongly convex and Lipschitz continuous of the gradients.In addition, the authors in [41] proposed the same type of algorithm, *ADD-OPT*, with only using combine-then-adapt variant,

$$z_i(k+1) = \sum_{j \in \mathcal{N}_i^{in}} b_{ij} z_j(k) - \alpha y_i$$

In addition, algorithms based on the gradient tracking technique for directed graphs have been developed and will be explained in chapter 3.

1.4 Challenges

As discussed in the preceding section, there are a variety of distributed optimization algorithms that perform properly on both undirected and directed networks. The research challenge that this thesis aims to tackle is the improvement of distributed optimization algorithms for undirected and directed graphs. To be more specific, the key research question addressed is the following:
Given the current state-of-the-art, is it possible to design and demonstrate with mathematical guarantees or at least through numerical computations by comparing with the existing methods, distributed algorithms that require less storage and communication for each node per iteration while achieving a higher convergence rate?

To address this challenge, we first present a distributed optimization algorithm for solving an economic dispatch problem in a distributed way and indicate the method's fast convergence in practice. In the second part, we develop a fast row-stochastic decentralized algorithm for solving distributed consensus optimization problems on directed graphs. The proposed method reduces the data stored and the data broadcast for each node by employing a novel momentum term. We also show under which conditions the method converges to the optimal solution at a linear rate. Moreover, in the numerical tests, we empirically show that the method is competitive against the other state-of-the-art methods.

1.5 Thesis outline and contributions

This section highlights the contributions of this thesis.

In chapter 2, we propose a distributed optimization method, which enjoys the advantages of both the primal and the dual domain methods, to address the distributed energy management problem of the economic dispatch of grids in order to balance the power demand and supply. By manipulating the primal problem, we show that the resulting dual problem can be solved by using a decentralized linearized alternating direction method of multipliers. Contrary to the Alternating Direction Method of Multipliers (ADMM), the proposed algorithm exhibits a lower computational cost not requiring at each main step iteration an explicit resolution of a local optimization problem. Moreover, we show through computer simulations that for the IEEE 14-bus and the IEEE 39-bus distribution test system the proposed algorithm achieves a significantly higher convergence rate when compared with recent distributed approaches to solve the proposed economic dispatch problem.

In chapter 3, 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. The research developed over this thesis has been published in the following journals.

[1]. Diyako Ghaderyan , Fernando Lobo Pereira, and A Pedro Aguiar. A fully distributed method for distributed multiagent system in a microgrid. Energy Reports, volume 7, pages 2294–2301. Elsevier, 2021. DOI: 10.1109/TAC.2023.3275927.

[2]. Diyako Ghaderyan , Necdet Serhat Aybat, A Pedro Aguiar, and Fernando

Lobo Pereira. A Fast Row-Stochastic Decentralized Method for Distributed Optimization Over Directed Graphs. IEEE Transactions on Automatic Control, 2023. DOI: 10.1016/j.egyr.2021.03.035

Chapter 2

Distributed Optimization for Power systems

2.1 Introduction

In recent years, intelligent electrical devices and advanced communication contributed to the existence of electrical power system configurations composed of strongly distributed components in terms of generation, storage, and load. As a consequence, the management of these types of networks grew in complexity and cannot be done through the traditional centralized approach because it would require complete information about the system, which is difficult to handle, even, with very high computational cost and time. Moreover, there are also limitations of flexibility and disregard of privacy. Hence, to deal with these issues, the management of the electrical power system should avoid centralized solutions also because communication among subsystems is more robust in distributed approaches than in centralized cases, requiring lower computational complexity.

Microgrids also play a crucial role in future grids. A microgrid can be disconnected from the main grid and supply the local grids, see more details in [42] and[43]. The main challenge in this type of systems is how to meet the balance between demand and generation efficiently. As mentioned before, the approach for the energy management system can be centralized and decentralized. In the centralized approach, the main grid and other buses are connected to the central entity in order to solve an optimization problem [44]. In the decentralized approach, agents can communicate with some neighbor agents, and the operational information is not shared among all agents. Hence, it is resilient to possible failures of one or more agents. One of the essential research priorities on microgrids is to

investigate frameworks to operate on a multi-agent system in which their participation is coordinated in a distributed way [45].

In this chapter, we present a fully distributed algorithm that does not require any central coordinators or leader agents. In the literature, one can find several works addressing the same goal, usually relying on distributed consensus approaches. In [46], the authors consider a leader node in order to meet a balance between the power generation and the load in a decentralized dispatch. In [47], an incremental cost consensus algorithm was proposed to solve the conventional centralized economic dispatch problem in a distributed manner. It is however not fully distributed because it needs a leader agent to collect the power requests from all agents to determine the mismatch. The authors in [6] proposed a fully distributed consensus algorithm to maintain the balance of the demand and the supply without considering transmission loss. In [48, 49], the Consensus + Innovations method was proposed to solve a single or a multistep economic dispatch problem in a fully distributed way for an undirected system. In [50], the author proposed a minimum-time consensus-based algorithm to solve the economic dispatch problem in a minimum number of time steps. The authors in [51] describe a distributed algorithm based on the push-sum and gradient method for solving the optimal distributed energy resources problem where directed networks are both time-varying and communication time delays, and in [52], an extension was done for lossy communication networks. The authors in [53], accelerate the convergence of the algorithm of [51], by proposing the distributed algorithm with a fixed step size based on the push-sum and gradient method to solve the optimal distributed energy resources over a time-varying directed network. In [54], a consensus-based algorithm was proposed to solve the energy management problem over undirected graphs with a noticeable convergence rate, and some recent papers on economic dispatch in a distributed way were presented in [55–57] .

In this chapter, by the decentralized linearized alternating direction method DLM [58], we propose an approach to solve the economic dispatch problem that achieves a significantly higher convergence rate when compared with the mentioned ones in the literature. Simulation results are based on the IEEE 14-bus and the IEEE 39-bus distribution test system, which consists 14 and 39 components respectively. The networks include dispatchable generators, wind generators, storage device, and loads, and the connections show communication lines between components in the microgrid.

Main Contributions of this chapter: We propose a fully distributed algorithm, which

is novel for power system applications, for solving an economic dispatch problem in an undirected communication network. Comprehensive numerical experiments are performed to compare state-of-the-art methods with the proposed algorithm in terms of computational effort. In particular, for the IEEE 14-bus and the IEEE 39-bus distribution test systems, we show that the proposed algorithm has a fast convergence rate in practice when compared with other existing algorithms with fixed step-sizes or diminishing step-sizes, which illustrates the effectiveness of the proposed approach.

2.2 Problem Formulation

We consider an electrical microgrid with $L > 1$ buses (agents) formed by nondispatchable and dispatchable generators, storage devices, and flexible loads. We formulate this network as a bidirectional connected graph $G = (V, E)$, where V is the set of L nodes and $E = \{e_1, ..., e_m\}$ is the set of edges which is required to not contain self-loops. We define $\mathcal{N}_i := \{j : (i, j) \in E\} \cup \{i\}$ as the collection of neighbors of agent i and its cardinality is represented by $d_i = |\mathcal{N}_i|$. From [48], we consider a power network with $L = N + M + Q$ Distributed Energy Resources (DERs) where N is the number of generators, M the number of energy devices and Q the number of loads. For a finite-time horizon T, the goal of the economic dispatch problem is to determine the optimal power of each agent that maximizes the economic benefits subject to the power balance and other constraints during the time periods $\mathcal{T} = \{1, 2, ..., T\}$. More precisely, we formulate the following optimization problem:

$$(P): \quad \max_{p_{i,t}} \sum_{t=1}^{T} \sum_{i=1}^{L} -C_i(p_{i,t}) \tag{2.1}$$

$$s.t. \quad \sum_{i=1}^{L} p_{i,t} = 0 \quad \forall t \in \mathcal{T} \tag{2.2}$$

$$p_i^{min} \leq p_{i,t} \leq p_i^{max} \quad \forall t \in \mathcal{T}, \quad i \in \mathcal{L} \tag{2.3}$$

$$\Delta \underline{p}_i \leq p_{i,t} - p_{i,t-1} \leq \Delta \overline{p}_i, \quad \forall t \in \mathcal{T} \quad i \in \mathcal{N} \tag{2.4}$$

$$E_{i,t} = E_{i,t-1} - p_{i,t}\Delta T \quad \forall t \in \mathcal{T}, \quad i \in \mathcal{M} \tag{2.5}$$

$$0 \leq E_{i,t} \leq \overline{E}_i \quad \forall t \in \mathcal{T}, \quad i \in \mathcal{M} \tag{2.6}$$

$$E_{i,T} = E_{i,0}, \quad \forall i \in \mathcal{M} \tag{2.7}$$

where for each agent i and period t, $C_i(p_{i,t})$ is the cost function, $p_{i,t}$ is the power from the loads, generators or storages and p_i^{max} and p_i^{min} are the upper and lower power limits of

the loads, generators, or storages, respectively. Constraint (2.2) corresponds to the power balance at each time step. Given the sets $\mathcal{N} = \{1, 2, ..., N\}$, $\mathcal{M} = \{N+1, N+2, ..., N+M\}$, $\mathcal{L} = \{1, 2, ..., N+M+Q\}$, $\Delta\overline{p}_i$ and $\Delta\underline{p}_i$ are the upper and lower bounds of ramping rates of generator i, respectively; $E_{i,t}$ is the energy stored in storage i at the end of time period t, ΔT is the period size, and $\overline{E}_i$ is the energy capacity of storage i. As in (2.7) we consider that the energy stored in storage i at the end of the period is equal to the initial energy state. Following [48], the cost function for each agent i is assumed to be quadratic $C_i(p_{i,t}) = \frac{1}{2}a_i {p_{i,t}}^2 + b_i p_{i,t} + c_i$ where $a_i > 0$, b_i, $c_i \geq 0$ are constants. In particular, $p_{i,t} > 0$ if the power is generated into the system and $p_{i,t} < 0$ if it is consumed from the system. For the case of a storage device, $p_{i,t} > 0$ if storage is discharging and $p_{i,t} < 0$ when it is charging. The constraints of the power balance (2.2) must satisfy the local capacity limits (2.3), where $p_i^{min}, p_i^{max} \leq 0$ for loads, $p_i^{min}, p_i^{max} \geq 0$ for generators and $p_i^{min} \leq 0, p_i^{max} \geq 0$ for the storage case.

The dispatch problem can also be expressed in a way that is slightly different from the problem (P). In this form, the formulation only takes into account the generators and the total loads. The objective function is to minimize the total generation cost while meeting the total demand. The problem is expressed as follows:

$$(P^{\top}): \quad \min_{p_{i,t}} \sum_{t=1}^{T} \sum_{i=1}^{N+M} C_i(p_{i,t}) \tag{2.8}$$

$$s.t. \quad \sum_{i=1}^{N+M} p_{i,t} = D_t \quad \forall t \in \mathcal{T} \tag{2.9}$$

$$p_i^{min} \leq p_{i,t} \leq p_i^{max} \quad \forall t \in \mathcal{T}, \quad i \in \mathcal{N} \cup \mathcal{M} \tag{2.10}$$

$$\Delta\underline{p}_i \leq p_{i,t} - p_{i,t-1} \leq \Delta\overline{p}_i, \quad \forall t \in \mathcal{T} \quad i \in \mathcal{N} \tag{2.11}$$

$$E_{i,t} = E_{i,t-1} - p_{i,t}\Delta T \quad \forall t \in \mathcal{T}, \quad i \in \mathcal{M} \tag{2.12}$$

$$0 \leq E_{i,t} \leq \overline{E}_i \quad \forall t \in \mathcal{T}, \quad i \in \mathcal{M} \tag{2.13}$$

$$E_{i,T} = E_{i,0}, \quad \forall i \in \mathcal{M} \tag{2.14}$$

where D_t is the total demand of the period t. The total demand of any period t is assumed to be

$$\sum_{i=1}^{N} p_i^{min} \leq D_t \leq \sum_{i=1}^{N} p_i^{max}$$

For more details on this optimization formulation, see [51, 53]. In this thesis, we focus only on the dispatch problem (P).

2.3 Distributed Scheduling

We describe some distributed methods that solve the problem (P) and (P') in a distributed way.

2.3.1 Consensus+Innovations

In this section, the dispatch problem (P) is solved using a fully distributed approach, where it is assumed that the cost function $C_i(p_{i,t})$ is quadratic. We derive the Lagrange function from (P) as follows:

$$L = \sum_{t=1}^{T}\sum_{i=1}^{L} C_i(p_{i,t}) - \sum_{t=1}^{T}\lambda_t \sum_{i=1}^{L} p_{i,t} \tag{2.15}$$

where $p_{i,t}$ satisfy the constraints (2.2)-(2.7). If none of the constraints (2.3)-(2.7) become binding[1], we then draw the following conclusion from the first order optimality conditions:

$$\lambda_t^* = \frac{dC_i(p_{i,t})}{dp_{i,t}} = a_i p_{i,t}^* + b_i \quad i = 1, ..., N, \quad t = 1, ..., T \tag{2.16}$$

i.e., at the optimal solution, marginal costs for all components i have to be equal to the same value, that is the price λ_t^*, and the optimal solution is also required to satisfy the power balance constraint (2.2). If the component maximum constraint is binding, then its marginal cost is lower than λ_t^*, and If the component minimum constraint is binding, then its marginal cost is greater than λ_t^*, i.e., The value of the Lagrange multipliers for the corresponding binding constraints affects the marginal cost from λ^*. Let use define $\lambda^* = (\lambda_1^*, ..., \lambda_T^*)^\top$ and the power outputs $P_i = (p_{i,1}, ..., p_{i,T})$, then the optimal setting P_i^* is given by

$$P_i^* = \mathcal{P}_i\left[\frac{\lambda^* - B_i}{a_i}\right] = \arg\min_{P_i \in \Omega_i} \left\| P_i - \frac{\lambda^* - B_i}{a_i} \right\| \tag{2.17}$$

where $\mathcal{P}_i$ is a projection operation that projects the argument into the feasible sets Ω_i, $B_i = (b_i, .., b_i)^\top$, and Ω_i is the set of constraints $(2.3 - 2.7)$, where Ω_i is the constraint (2.3) if it is a load, (2.3) and (2.4) if it is a generator or (2.3)-(2.7) if it is a storage. The optimal power setting should also satisfy the power balanced (2.2) [49], which means that

$$\sum_{i=1}^{L} P_i^* = \sum_{i=1}^{L} \mathcal{P}_i\left[\frac{\lambda^* - B_i}{a_i}\right] = 0 \tag{2.18}$$

[1] A constraint is binding if the constraint function satisfies equality at the optimal solution.

The equality constraint (2.18), however, cannot be resolved in a directed way because a distributed approach is required, in which each agent only has access to local data.

Therefore, in [48, 49], the authors proposed a fully distributed method known as Consensus+Innovations for solving the dispatch problem (P) in a distributed way that reaches an agreement on λ^* that satisfies the equality condition (2.18). In this case, every generator, load, and storage connected to the microgrid is assumed to be an agent, and each agent is only aware of its local information and can only communicate with its neighbors to achieve a global solution to the problem. Denote $\lambda_i = (\lambda_{i,1}, ..., \lambda_{i,T})'$, then the Consensus+Innovations approach is updated in k-th step as follows:

$$\lambda_i(k+1) = \lambda_i(k) - \beta_i \sum_{j \in \mathcal{N}_i} (\lambda_i(k) - \lambda_j(k)) - \alpha_i \sum_{l \in \mathcal{N}_i} P_l(k) \tag{2.19}$$

$$P_i(k+1) = \mathcal{P}_i \left[\frac{\lambda_i(k+1) - B_i}{a_i} \right] \tag{2.20}$$

where $\mathcal{N}_i$ is the communication neighborhood of agent i. The step-sizes β_i and α_i are positive diminish step-sizes and satisfy the conditions:

$$\begin{aligned} &a) \quad \sum_i \alpha_i = \sum_i \beta_i = \infty, \\ &b) \quad \frac{\beta_i}{\alpha_i} \to \infty \qquad \text{if} \quad \alpha_i \to \infty \end{aligned} \tag{2.21}$$

For more details, see [49].

2.3.2 DLMC

This section describes the main contribution of this chapter. Following the approach in [53] and [51], we start with the derivation of the dual problem. Since the objective function in (2.1) is strictly concave, the constraint in (2.2) is affine, and (2.3)-(2.7) is a polyhedral set, one can conclude that the duality problem has a zero duality gap [2]. Moreover, the dual optimal set is nonempty. Therefore, solutions to the problem (P) can be obtained by solving its dual problem. To this end, we introduce the Lagrange multiplier $\lambda = (\lambda_1, ..., \lambda_T)^\top \geq 0$ and define the corresponding Lagrange dual function

$$(D) \qquad \min_{\lambda} \sum_{i=1}^{L} \Psi_i(\lambda) \tag{2.22}$$

where

$$\Psi_i(\lambda) = \max_{p_i} \sum_{t=1}^{T} -C_i(p_{i,t}) + \lambda^\top p_i \tag{2.23}$$

and $p_i = (p_{i,1}, ..., p_{i,T})^\top$ satisfies the constraint (2.3) if is a load, (2.3) and (2.4) if it is a generator or (2.3)-(2.7) if it is a storage. Due to the strict convexity of $C_i(p_{i,t})$, the right-hand side of (2.23) has a unique maximizer for every λ as follows [2].

$$p_i(\lambda) = \arg\min_{p_i} \sum_{t=1}^{T} C_i(p_{i,t}) - \lambda^\top p_i \tag{2.24}$$

Since the cost function is strictly convex, then following from [59], one can conclude that the function $\Psi_i(\lambda)$ is continuously differentiable and the gradient satisfies

$$\nabla\Psi_i(\lambda) = p_i(\lambda) = \mathcal{P}_i\left[\frac{\lambda - B_i}{a_i}\right] = \arg\min_{p_i \in \Omega_n} \left\| p_i - \frac{\lambda - B_i}{a_i} \right\| \tag{2.25}$$

where $B_i = (b_i, .., b_i)^\top$ and Ω_n is the set of constraints $(2.3 - 2.7)$.

In addition, there is at least one optimal solution to the dual problem (2.22), and the unique optimal solution to the primal problem (2.1) can be obtained by solving the local optimization problem (2.23), which is given by $p_i(\lambda^*)$ where λ^* is any dual optimal solution [2].

Note that the dual problem (2.22) is an unconstrained problem and can be solved using the Decentralized Linearized Alternating Direction Method (DLM) [58] which solves (2.22) in a fully distributed way.

To apply the DLM algorithm, we need first to conclude that the Assumptions 2.1 and 2.2 below hold, see [58].

Assumption 2.1. *The local cost functions Ψ_i are convex and differentiable.*

Assumption 2.2. *Each local cost function Ψ_i has Lipschitz continuous gradients. Moreover, there is a positive constant $M > 0$ such that for all $i \in \mathcal{L}$ and for any pair of points x and y , the following holds*

$$\left\|\nabla\Psi_i(x) - \nabla\Psi_i(y)\right\|_2 \leq M \left\|x - y\right\|_2 \tag{2.26}$$

Assumption 2.1 can be easily verified because the $C_i(P)$ is strictly convex, and therefore dual convex problem Ψ_i is differentiable. Assumption 2.2 is also satisfied as follows (see [60], Theorem 1.5.5 in [61]): given x and y

$$\left\| \mathcal{P}_i\left(\frac{x - b_i}{a_i}\right) - \mathcal{P}_i\left(\frac{y - b_i}{a_i}\right) \right\|_2 \leq \frac{1}{a_i} \left\|x - y\right\|_2 \tag{2.27}$$

We are now ready to apply the DLM algorithm [58] where each agent i only knows its local cost function and capacity limits as well as the ones from the neighbors, and obtain

the following iterative equations:

$$\lambda_i(t+1) = \lambda_i(t) - \frac{1}{\tilde{d}_i}\left[\mathcal{P}_i\left[\frac{\lambda_i(t)-b_i}{a_i}\right] + c\sum_{j\in\mathcal{N}_i}[\lambda_i(t)-\lambda_j(t)] + \phi_i(t)\right] \tag{2.28}$$

$$\phi_i(t+1) = \phi_i(t) + c\sum_{j\in\mathcal{N}_i}[\lambda_i(t+1)-\lambda_j(t+1)] \tag{2.29}$$

where $\tilde{d}_i = 2cd_i + \rho$ with c, ρ being positive constant, and d_i is such that $D = diag(d_1, ...d_N)$. Algorithm 1 shows the pseudo-code of the distributed proposed algo-

Algorithm 1 Distributed algorithm (DLMC)

Input: The bidirectional graph $G = (V, E)$, the step size c, $\tilde{d}$, and picks the initial variables $\lambda_i(0)$ and $\phi_i(0) = 0$.
Ouput: The optimal incremental cost λ^* and the optimal power $p_i^\star$.

1: **for all** $t = 0, 1, ...$ **do**
2: **for all** $i = 0, 1, ..., N$ **do**
3: Update the variable $\lambda_i(t+1)$ according to (2.28)
4: Update the variable $\phi_i(t+1)$ according to (2.29)
5: Update the variable $p_i = \mathcal{P}_i\left[\frac{\lambda - b_i}{a_i}\right]$
6: **end for**
7: **end for**

rithm, which consists of two steps: In the first step, each agent updates the marginal cost by communicating with the neighbors and using the variable ϕ_i that plays a role as a dual variable for the problem (2.22); In the second step, the dual variable ϕ_i is updated by the integration of the past integrating $c\sum_{j\in\mathcal{N}_i}[\lambda_i(t+1)-\lambda_j(t+1)]$ between local variable λ_i and its neighborhood λ_j.

The convergence of the proposed algorithm follows directly from [58] and equalities (2.22)-(2.25). As stated in [58], there is a guarantee of convergence, if $c\gamma_u + \rho > 2$, where γ_u is the smallest eigenvalue of matrix L_u.

Remark 2.3. The authors of [48] and [54] solve the optimization problem (P) (2.1)-(2.7) by converting it to a restricted agreement problem that arises from the first-order optimality conditions along with the power balance constraint. We show that by exploiting the dual problem and applying the DLM algorithm we obtain considerable performance advantages because the DLM approach enjoys the advantages of both the primal and the dual methods, i.e., low computation burden and fast convergence.

2.3.3 Push-Sum

In [51], the author proposed a distributed optimization method called Push-Sum from the method in [40]. In this paper, the problem $(P^\top)$ (2.8)- (2.14) has been solved in a distributed way instead of the problem (P). In order to make the Lagrange dual function in a sum of distributed functions, they assumed virtual local demands $D_{i,t}$ such that

$$\sum_{i=1}^{N+M} D_{i,t} = D_t \qquad t = 1, ..., T$$

The corresponding Lagrange dual function can be written

$$(D) \qquad \max_{\lambda} \sum_{i=1}^{L} \Psi_i(\lambda) \tag{2.30}$$

where

$$\Psi_i(\lambda) = \min_{p_i} \sum_{t=1}^{T} C_i(p_{i,t}) + \lambda^\top (p_i - D^i) \tag{2.31}$$

and $D^i = (D_{i,1}, ..., D_{i,T})^\top$. However, we adjust Push-Sum to solve the problem (P) and we can write the method as follows

$$w_i(t+1) = \frac{v_i(t)}{d_i + 1} + \sum_{j \in \mathcal{N}_i} \frac{v_j(t)}{d_j + 1} \tag{2.32}$$

$$y_i(t+1) = \frac{v_i(t)}{d_i + 1} + \sum_{j \in \mathcal{N}_i} \frac{y_j(t)}{d_j + 1} \tag{2.33}$$

$$\lambda_i(t+1) = \frac{w_i(t+1)}{y_i(t+1)} \tag{2.34}$$

$$v_i(t+1) = w_i(t+1) - \alpha \mathcal{P}_i \left[\frac{\lambda_i(t+1) - b_i}{a_i} \right] \tag{2.35}$$

2.3.4 EXTRAC

In [54], the authors proposed a consensus-based method from [20] to solve the problem (P) in a distributed fully way. The approach is a first-order decentralized algorithm written as follows:

$$\begin{aligned} \lambda_i(t+1) =& \lambda_i(t) + \sum_{j=1}^{L} w_{ij} \lambda_j(t) - \sum_{j=1}^{L} \hat{w}_{ij} \lambda_j(t-1) \\ & - \alpha \left(\mathcal{P}_i \left[\frac{\lambda_i(t) - b_i}{a_i} \right] - \mathcal{P}_i \left[\frac{\lambda_i(t-1) - b_i}{a_i} \right] \right) \end{aligned}$$

where $W = (w_{ij}) \in \mathcal{R}^{L\times L}$ is a the topology of the graph which $w_{ij} = w_{ji} > 0$ if the node i connect to the node j. The entries of the matrix W satisfy $\sum_{j=1,j\neq i}^{L} w_{ij} \leq 1, w_{ii} = 1 - \sum_{j=1,j\neq i}^{L} w_{ij}$ and $\hat{W} = (\hat{w}_{ij}) = \frac{1+W}{2}$.

2.4 Performance Assessment

This section illustrates through computer simulations the performance of the proposed approach. In particular, we compare its convergence with that of four methods presented in the literature in two case studies: single time step ($T = 1$) and multiple time steps.

2.4.1 Single Time Step

We compare the rate of convergence of the proposed algorithm for the following two systems: an IEEE 14-bus system and an IEEE 39-bus system. In order to facilitate the reference to the algorithms, we name our approach by DLMC, and the algorithm in [54] EXTRAC which is from the EXTRA algorithm in [20].

2.4.1.1 Case Study 1

In this case study, we test the algorithms for the IEEE 14-bus system displayed in Fig.2.1 that is formed by five dispatchable generators located at buses 1,2,3,6 and 8; a storage device at bus 7 and a wind generator at bus 9. The parameters of generators, loads, and ES are given in Table 2.1 - 2.3, respectively. The remaining buses are load buses. In this case study, the communication network is the same as the physical grid.

We consider $c = 5.4$, $c = 6$ and $\rho = 4$ in DLMC approach, $\alpha = 0.015$ in EXTRAC approach [54], $\alpha_i = (0.055/i^{0.98})$, $\beta_i = (0.2/i^{0.001})$ in Consensus + Innovation approach [48], $\gamma(i) = \frac{a}{i+b}$ in Push-Sum approach in [51]. These parameters are held for all case studies unless we mention the changes.

Fig.2.2 shows the output of the computation of the active power of the proposed algorithm along the iterations. It can be seen that the proposed approach converges to the final value after 25 iterations. [t]

In Fig.2.3, we show the evolution of the residual, which is defined as $\frac{\|p(t)-p^*\|_2}{\|p(0)-p^*\|_2}$ where $p(t) = [p_1(t), ..., p_{14}(t)]$ and $p^* = [p_1^*, ..., p_{14}^*]$ is the global optimal solution. From

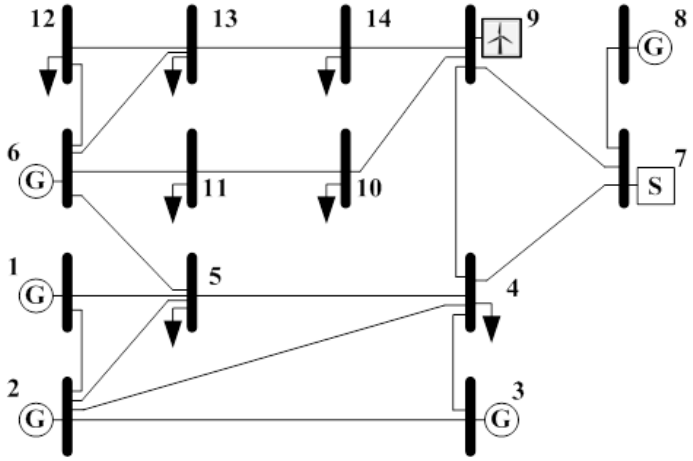


FIGURE 2.1: IEEE 14-bus System

TABLE 2.1: Generator Parameters (mu = Monetary Units)

Bus	$a_i[mu/pu^2]$	$b_i[mu/pu]$	$\Delta\overline{p}_i[mu/5min]$	$p_i^{max}[pu]$
1	0.084	2	20	100
2	0.056	3	5	105
3	0.070	4	10	100
6	0.060	4	15	90
8	0.080	2.5	15	80

TABLE 2.2: Load Parameters (mu = Monetary Units)

Bus	$a_i[mu/pu^2]$	$b_i[mu/pu]$	$p_i^{min}[pu]$	$p_i^{max}[pu]$
4	0.070	8	-80	-6
5	0.064	6.5	-30	-2
10	0.076	7	-40	-5
11	0.070	7.5	-25	-2
12	0.080	8	-90	-8
13	0.070	7	-30	-2
14	0.084	8	-80	-10

TABLE 2.3: Storage Parameters (mu = Monetary Units)

Bus	$a_i[mu/pu^2]$	$\Delta\overline{p}_i[mu/5min]$	$p_i^{max}[pu]$	$\overline{E}_i[puh]$
7	0.02	50	50	20

the figure, it can be seen that for an accuracy level $\epsilon = 10^{-2}$(green horizontal line), our approach has a faster convergence rate than the other two approaches, and in the accuracy level $\epsilon = 10^{-4}$, the convergence rate of DLMC approach is faster than EXTRAC algorithm. It is important to stress that reducing such numbers of iterations in the convergence of a problem, especially for large systems is quite significant. In Fig. 2.4, we show the convergence of the marginal cost of the four approaches. As shown, EXTRAC requires

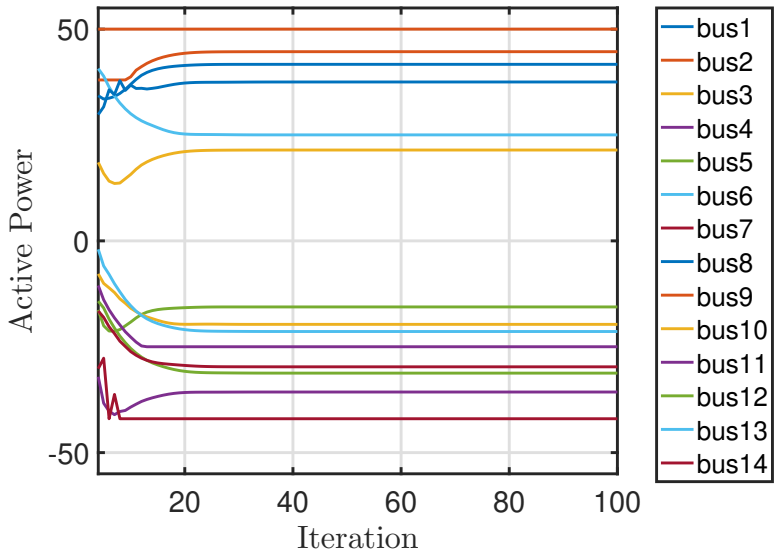


FIGURE 2.2: Active power of 14 bus

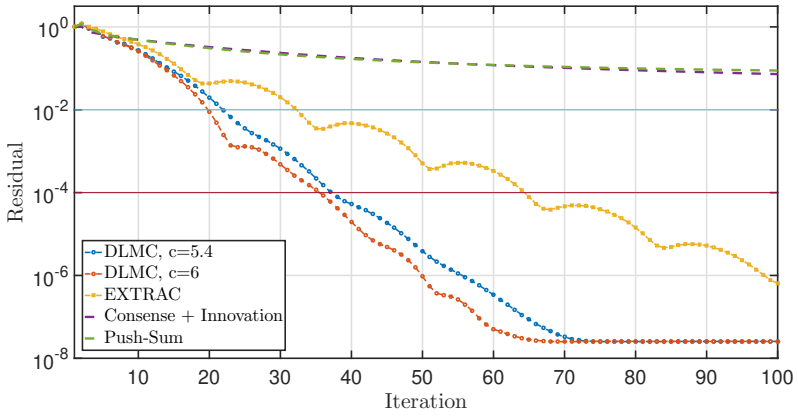


FIGURE 2.3: Residual for the four approaches

more the number of iterations to reach an equal marginal cost value rather than DLMC.

2.4.1.2 Case Study 2

In this case study, as in [54], we show that the convergence rate improves when additional edges are added to the communication network. In this particular experiment, we include the following bidirectional communication links buses 3 and 7, buses 4 and 10, buses 5 and 10, buses 8 and 9, and bus 11 and 14. In Fig.2.5, it can be seen that the convergence rate DLMC is faster than EXTRAC. We consider $c = 4.5$ for DLMC approach in order to avoid the zigzag process.

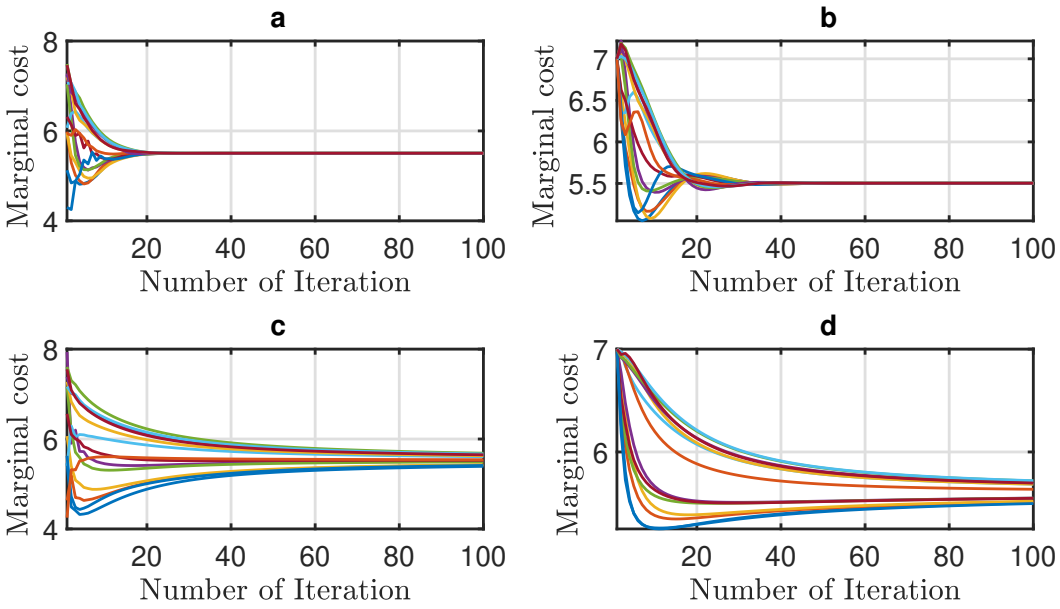


FIGURE 2.4: Marginal cost for all buses in IEEE 14-bus a) DLMC approach b) EXTRAC approach c) Consensus + Innovation approach d)Push-Sum approach

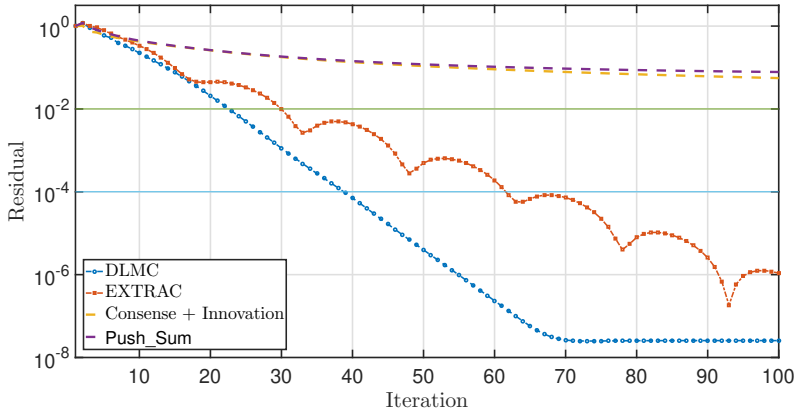


FIGURE 2.5: Residual for the topology which is added by some links

2.4.1.3 Case Study 3

In this section, we compare our method with a known distributed method to solve distributed optimization problem, called Alternating Direction Method of Multipliers (ADMM), which is applied in extensive applications, i.e. see [62–64]. In ADMM method, each agent n maintains a local estimate p_n, and the quantity y which is available to all

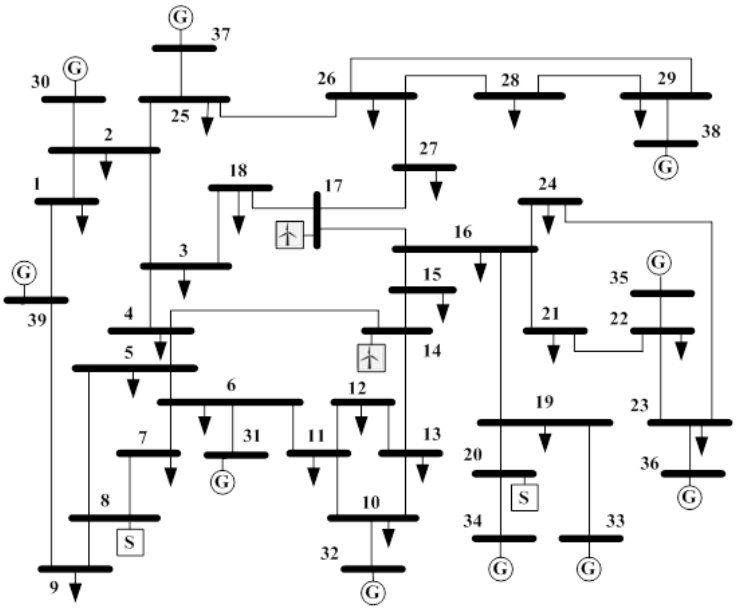


FIGURE 2.6: IEEE 39-bus

agents, and the algorithm is updated as follows:

$$p_n(t+1) = \arg\min_{p_n} C_n(p_n) + y(t)p_n + \frac{\rho}{2}\|p_n - \left(p_n(t) - \bar{p}_n(t)\right)\|^2$$
$$y(t+1) = y(t) + \rho\bar{p}_n(t+1)$$

where ρ is a step size, and $\bar{p}_n(t) = \frac{1}{14}\sum_{n=1}^{L} p_n$ is the average of p_n, see [64].
Unlike the proposed method, the ADMM is not a fully distributed method since the local variables include components of the global variable. Moreover, it also has a high computational burden since each agent needs to solve an optimization problem at each iteration. In contrast, the proposed method requires at each iteration a (sub)gradient step, and, hence, the computational cost is lower. Fig.2.7 and Fig.2.8 illustrate this aspect for the IEEE 14-bus system, where one can see the evolution of the residual with the number of iterations and the runtime in a single step, respectively. As shown, the ADMM has a high computational cost when compared with the proposed method.

2.4.1.4 Case Study 4

In this case study, we test the algorithm for a larger system, the IEEE 39-bus, which consists of 39 buses; 10 dispatchable generators, 2 wind generators, 2 storages and 25 loads as shown in Fig.2.6. Dispatchable generators are located in buses 30-39, wind generators are placed in buses 14 and 17, and storages are located in buses 8 and 20. As in the case of

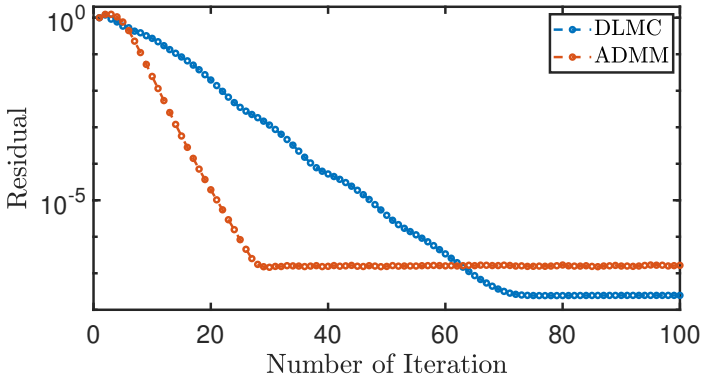


FIGURE 2.7: Residual by number of iterations

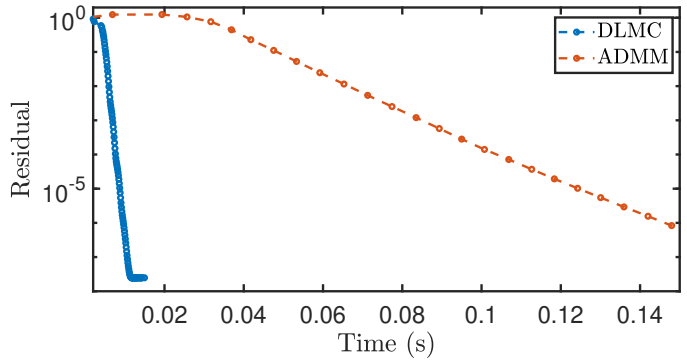


FIGURE 2.8: Residual by runtime

the IEEE 14-bus, we consider the physical electrical connections between buses, as shown in Fig.2.6. As case study 1, the simulation is performed for 5 min a single time step, and we have the ramp up and down limits for dispatchable generators and storages, and the ramp up and down limits are equal $\Delta\underline{p}_i$ = -$\Delta\overline{p}_i$. The parameters for generators, loads, and storages are in Table 2.4-2.6. The values of the upper bound of active power for wind generator buses 14 and 17 are 40 and 50$[pu]$ respectively. The rationale for the values of the upper and lower constraints for generators, loads, and storage given in Table 2.4 are in [48].

In this case study, we compare four approaches. The residual evolution along the number of iterations is shown in Fig. 2.9. We consider the best constants $c = 12.9$, $\rho = 4$ for DLMC, $\alpha = 0.010$ for EXTRAC, and the same α and β for Consensus + innovations approach, γ for Push-Sum approach in case study 1. One can see that DLMC and EXTRAC algorithm converge fast, but clearly, the convergence rate of DLMC is higher than EXTRAC. As shown in Fig. 2.10, the marginal cost of both algorithms converges to the same value, but EXTRAC algorithm requires 71 iterations to obtain the same value

TABLE 2.4: Load Parameters (mu = Monetary Units)

Bus	$a_i[mu/pu^2]$	$b_i[mu/pu]$	$p_i^{min}[pu]$	$p_i^{max}[pu]$
1	0.084	8	-80	-6
2	0.056	6.5	-30	-2
3	0.070	6.5	-50	-5
4	0.075	7	-55	-7
5	0.064	7.5	-40	-8
6	0.061	8	-25	-2
7	0.066	7	-90	-18
9	0.073	8	-75	-9
10	0.064	6.5	-80	-5
11	0.078	6.5	-30	-6
12	0.077	7	-60	-8
13	0.084	7.5	-50	-4
15	0.070	7	-35	-6
16	0.063	6.5	-95	-14
18	0.065	7	-80	-12
19	0.073	8.5	-45	-8
21	0.074	8	-30	-10
22	0.067	7	-100	-17
23	0.090	8	-50	-6
24	0.082	6.5	-60	-9
25	0.088	8	-40	-5
26	0.071	6.5	-65	-11
27	0.069	6.5	-100	-16
28	0.064	7	-25	-2
29	0.060	7.5	-85	-5

TABLE 2.5: Generator Parameters (mu = Monetary Units)

Bus	$a_i[mu/pu^2]$	$b_i[mu/pu]$	$\Delta\overline{p}_i[mu/5min]$	$p_i^{max}[pu]$
30	0.080	3	25	100
31	0.076	2	30	105
32	0.070	3	25	110
33	0.091	4	15	120
34	0.087	3	20	125
35	0.084	4	25	135
36	0.075	2.5	20	80
37	0.078	3	15	95
38	0.082	4	20	90
39	0.077	3.5	25	100

TABLE 2.6: Storage Parameters (mu = Monetary Units)

Bus	$a_i[mu/pu^2]$	$\Delta\overline{p}_i[mu/5min]$	$p_i^{max}[pu]$	$\overline{E}_i[puh]$
8	0.02	50	50	20
20	0.03	40	60	20

of our case at incremental cost 44. As it is clear, both algorithms are very efficient in solving a large system with a low number of iterations. In Fig.2.11, we show the output of the computation of the active power of the proposed algorithm along the iterations.

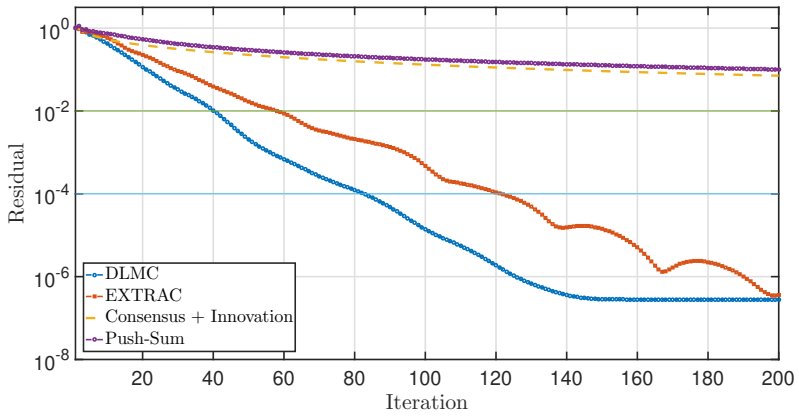


FIGURE 2.9: Residual for IEEE 39-bus

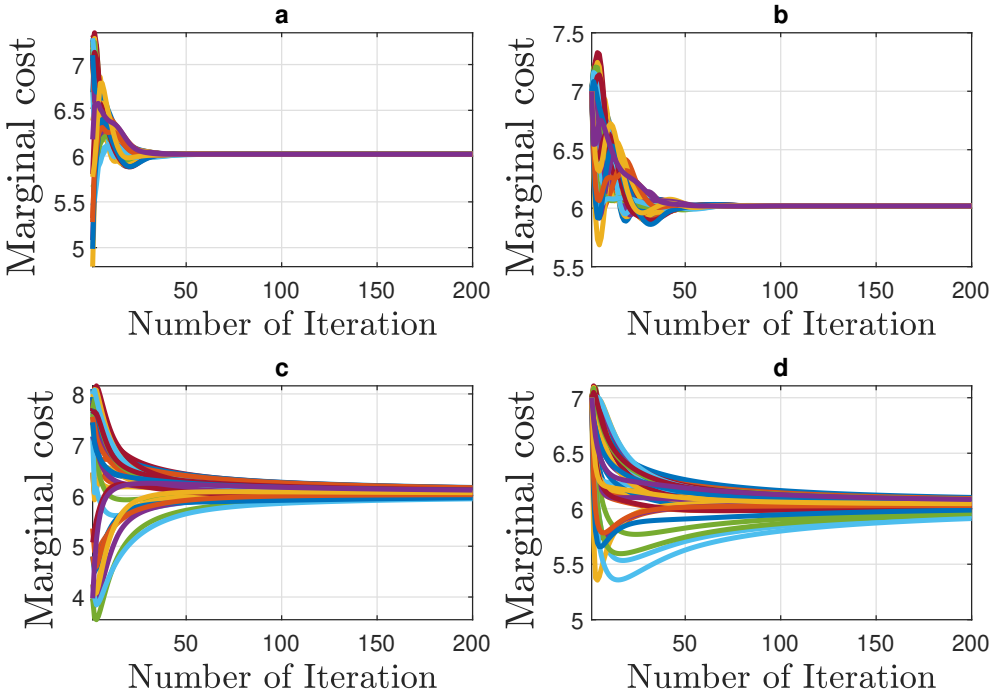


FIGURE 2.10: Marginal cost for all buses in IEEE 39-bus a) DLMC approach b) EXTRAC approach c) Consensus + Innovation approach d)Push-Sum approach

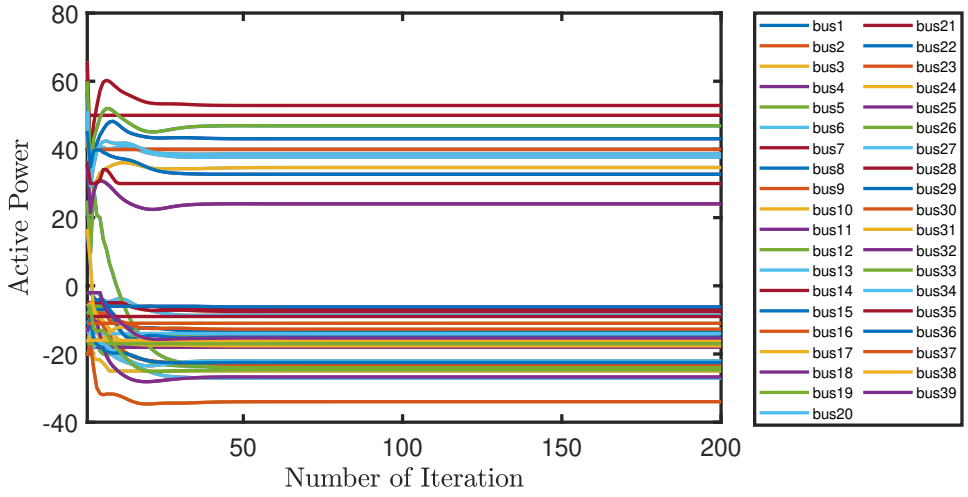


FIGURE 2.11: Active power of 39 bus

2.4.2 Multiple Time Steps

In this section, we consider an optimal distributed coordination of distributed generators with energy storage (ES) over multiple-time periods for a dispatch problem considered in the previous section. When we have storage in the system, in which not only power but also energy is limited, it is required to be optimized over multiple time steps simultaneously. In this case study, charging/discharging the ES are modeled efficiently, see [48]. In [51] and [65], they have also considered charging/discharging losess equation separately in the problem for multiple time steps.

2.4.2.1 Case Study 5

We use the same IEEE 14-bus network system in Fig. 2.1. The upper and lower power limits of generators and storage stay the same over a longer time period but the load's parameters change over a day. We use the data for one day with 30 minutes time-step for our simulation.

The parameters of generators, loads, and ES are given in Table 2.1 - 2.3 respectively. The ramp down and up is equal, $\Delta\overline{p}_i = -\Delta\underline{p}_i$. To solve the dual problem (D) in a distributed way over 24 hours, the value of parameters $c = 7$ and $\rho = 12$ are considered for DLMC algorithm, and $\alpha = 0.012$ for EXTRAC approach. All $\lambda_{i,T}$ converge to the optimal incremental cost and we show the $\lambda_{i,1}$, which converge to $\lambda_1^* = 4.8581$ in Fig. 2.12 for DLMC and EXTRAC algorithms.The marginal cost of both algorithms converges to the

same value, but EXTRAC algorithm requires 157 iterations to obtain the same value of our case at incremental cost 115. The storage works efficiently and the charging and discharging power. Note that for ES over 24 hours are shown in Fig.2.13, and we can see the state of charge is the same at the beginning and at the end of the scheduling period. In Fig. 2.14, we can see the optimal power for generators and storage over one day for DLMC approach. We also show the optimal loads in Fig. 2.15. It can be seen that both algorithms converge to the optimal generation in less 250 steps efficiently even though DLMC converges faster than EXTRAC. The simulation result of residual for both approaches at period 1 is shown in Fig. 2.16.

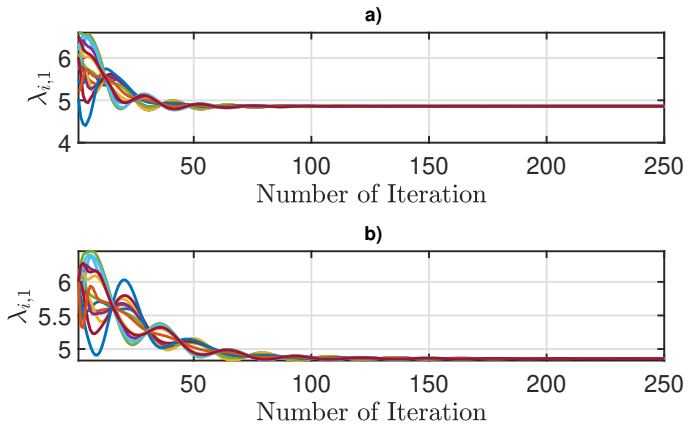


FIGURE 2.12: Marginal cost a)DLMC approach b)EXTRAC approach

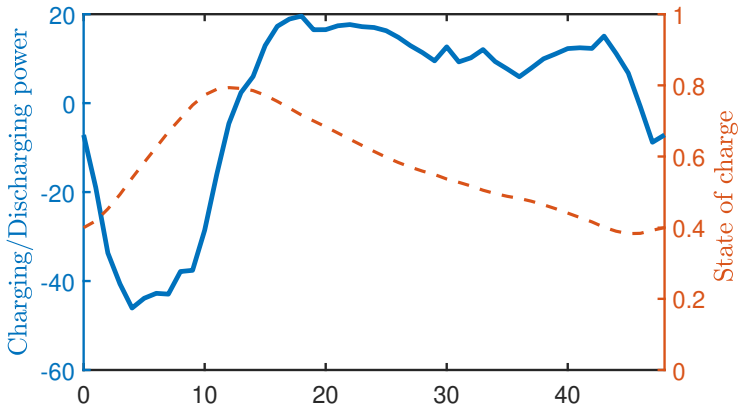


FIGURE 2.13: Charging/ Discharging Storage

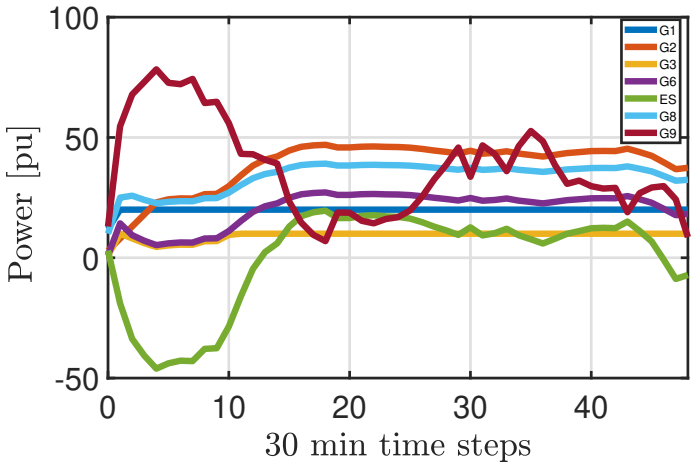


FIGURE 2.14: Power for Generations and Storage

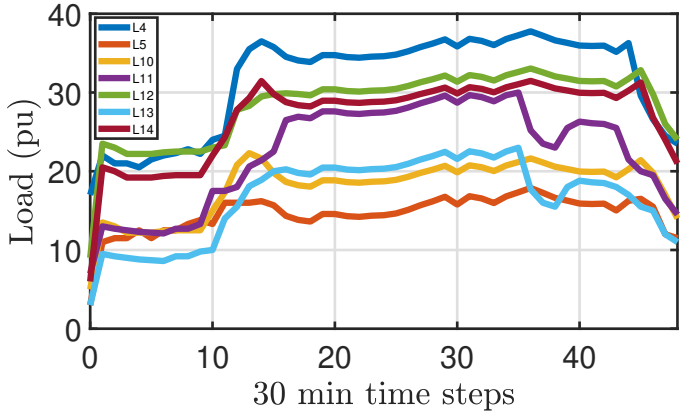


FIGURE 2.15: Consumption by Loads

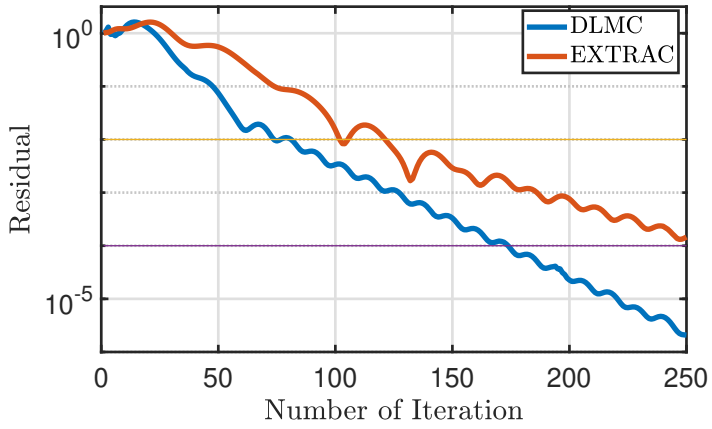


FIGURE 2.16: Residual for DLMC and EXTRAC approaches

Chapter 3

A Fast Row-Stochastic Decentralized Method for Distributed Optimization Over Directed Graphs

3.1 Introduction

In this chapter, we focus on the classic consensus optimization problem formulated in the form

$$x^* \in \arg\min_{x\in\mathbb{R}^p} \bar{f}(x) \triangleq \frac{1}{n}\sum_{i=1}^{n} f_i(x), \tag{3.1}$$

where the objective function $\bar{f}$ is the average of all individual cost functions $\{f_i\}_{i=1}^n$, $f_i : \mathbb{R}^p \to \mathbb{R}$ is the private function of agent i. This problem appears in a variety of applications, e.g., sensor networks [66, 67], distributed control [68], large-scale machine learning [39, 64, 69–73], distributed estimation [74].

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 $\tilde{x}_i$ denote a local estimate of the optimal decision for $i \in \mathcal{V}$. For merely convex objectives, we call $\tilde{\mathbf{x}} = [\tilde{x}_i]_{i\in\mathcal{V}}$ an ϵ-solution if $|\frac{1}{n}\sum_{i\in\mathcal{V}} f_i(\tilde{x}_i) - \bar{f}^*| \leq \epsilon$ and the consensus violation satisfies $\max\{\left\|\tilde{x}_i - \tilde{x}_j\right\| : (i,j) \in \mathcal{E}\} \leq \epsilon$ for $i \in \mathcal{V}$, where $\bar{f}^*$ denotes the optimal value of (3.1). On the other hand, for strongly convex problems, we say that $\tilde{\mathbf{x}}$ is ϵ-optimal, if $\|\tilde{x}_i - x^*\| \leq \epsilon$ for all $i \in \mathcal{V}$.

Inspired by the seminal work in [13], the authors in [16] proposed a distributed (sub)gradient method for solving (3.1). When each f_i is closed convex, the method in [16] 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. Note that 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 simply closed convex constraint sets, several distributed projected subgradient methods have been proposed for solving

$$\min\{\bar{f}(x) : \ x \in \mathcal{X}\}; \tag{3.2}$$

e.g., the method in [60] solves (3.2) employing exact subgradient evaluations, while the method in [75] can handle subgradients corrupted by stochastic noise. In [18], algorithms based on dual averaging of subgradients are studied for solving (3.2) assuming $\mathbf{0} \in \mathcal{X}$. In [75], 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 [18] 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 [76] 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 (3.2) such that each agent-i only knows f_i, $\mathcal{X}_i$ and $\mathcal{X}_0$. In [19], 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 [20], 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., [77–80] –some handling composite convex functions [23, 24], and some even handling complicated constraints [81, 82].

The works that we have discussed above are designed for undirected networks; hence, they correspond to balanced graphs [1] if we treat undirected networks as a special case of directed networks. However, directed networks may well be unbalanced; this situation

[1] A directed graph is a balanced graph when the incoming and outgoing edges at each vertex are equal.

arises especially for directed time-varying networks. For general directed networks, the first works to employ push-sum consensus protocol [83] (for computing an average over directed networks) within the distributed optimization framework (using the dual averaging method) are [39, 73]. A follow-up work in this direction is the subgradient-push method proposed in [40] that combines the push-sum protocol with the distributed subgradient method [16] (for minimization of convex functions). More precisely, the method in [40] applies to (3.1) 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 [84], which is a distributed method for directed graphs, achieves R-linear convergence rate [1]; it combines EXTRA [20] with push-sum approach [83]. 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 [29] and ADD-OPT [85] 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., $\mathcal{AB}$, $\mathcal{AB}$m and Push-Pull [86–88]. Later, $\mathcal{ABN}$ method is proposed in [89] 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 [41], 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 [41] is extended to handle uncoordinated step-sizes in the FROST algorithm [90] and to incorporate Nesterov acceleration leading to the FROZEN algorithm [89]. Finally, the D-DNGT algorithm proposed in [91] employs heavy-ball momentum and can handle nonuniform step-sizes. Some recent work extended the synchronous methods for directed

[1] Assume that a sequence $x(k)$ converges to x^* in a certain norm $\|\cdot\|$. We say that convergence is R-linear if $0 < \lambda < 1$ and some positive constant μ exist such that $\|x(k) - x^*\| \leq \mu\lambda^k$ for all k.

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., [92–94].

Main contributions of this chapter: In this chapter, 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 [41, 89–91] crucially depend on the gradient tracking technique to establish linear rate, in this chapter 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 $\nabla f_i(\boldsymbol{x}_i(k+1))$ and $\nabla f_i(\boldsymbol{x}_i(k))$ 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 chapter 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 $\boldsymbol{x}$ 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. Compared 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 3.1. In the numerical tests, we empirically show that FRSD is competitive against the other state-of-the-art methods: Xi-row [41], D-DNGT [91], FROZEN and $\mathcal{AB}\mathcal{N}$ [89], $\mathcal{AB}$ [86], $\mathcal{AB}$m [87], Push-DIGing [29] and Push-Pull [88].

Notation: In this chapter, the bold letters denote vectors, e.g., $\boldsymbol{x} \in \mathbb{R}^p$, and $[\boldsymbol{x}]_j$ denotes the j-th element of $\boldsymbol{x}$. 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 $X \in \mathbb{R}^{n\times n}$, $\mathbf{diag}(X) \in \mathbb{R}^{n\times n}$ denotes the diagonal matrix of which diagonal is equal to that of $X \in$

$\mathbb{R}^{n\times n}$. Moreover, given $v \in \mathbb{R}^n$, $\mathbf{diag}(v)$ is a diagonal matrix with its diagonal equal to v. $I_n = [e_1, \dots e_n]_{i=1}^n$ denotes the $n \times n$ identity matrix, where 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.

3.2 Design, Comparison and analysis of FRSD

Consider the consensus optimization problem (3.1) 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 $f_i : \mathbb{R}^p \to \mathbb{R}$, 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., $\mathcal{N}_i^{in} \triangleq \{j \in \mathcal{V} : \ (j,i) \in \mathcal{E}\} \cup \{i\}$. 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.

We make the following assumptions throughout this chapter.

Assumption 3.1. *$\mathcal{G}$ is directed and strongly connected.*

Assumption 3.2. *For every $i \in \mathcal{V}$, the local function f_i is L-smooth, i.e., it is differentiable with a Lipschitz gradient:*

$$\|\nabla f_i(x) - \nabla f_i(x')\| \leq L\|x - x'\|, \quad \forall\, x, x' \in \mathbb{R}^p. \tag{3.3}$$

Assumption 3.3. *For all $i \in \mathcal{V}$, f_i is μ-strongly convex, i.e.,*

$$f_i(x') \geq f_i(x) + \nabla f_i(x)^\top (x' - x) + \frac{\mu}{2} \| x' - x\|^2 \tag{3.4}$$

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

Remark 3.4. Under Assumption 3.3, the optimal solution to (3.1) is unique, denoted by x^*.

Definition 3.5. Define $\mathbf{x} \triangleq \left[x_1^\top, \dots, x_n^\top\right]^\top \in \mathbb{R}^{np}$ and $\mathbf{y} \triangleq \left[y_1{}^\top, \dots, y_n{}^\top\right]^\top \in \mathbb{R}^{np}$, where $x_i, y_i \in \mathbb{R}^p$ 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 $x_i(k)$ and $y_i(k)$ for $i \in \mathcal{V}$. Let $f : \mathbb{R}^{np} \to \mathbb{R}$ be a function of local variables $\{x_i\}_{i\in\mathcal{V}}$ such that $f(\mathbf{x}) \triangleq \sum_{i\in\mathcal{V}} f_i(x_i)$ for $\mathbf{x} \in \mathbb{R}^{np}$ and $\nabla f(\mathbf{x}) \triangleq \left[\nabla f_1(x_1)^\top, ..., \nabla f_n(x_n)^\top\right]^\top \in \mathbb{R}^{np}$, where $\nabla f_i(x_i) \in \mathbb{R}^p$ denotes the gradient of f_i at $x_i \in \mathbb{R}^p$.

We next propose our decentralized optimization algorithm FRSD to solve the consensus optimization problem in (3.1).

Algorithm 2 Fast row-stochastic decentralized algorithm (FRSD)

Input: $x_i(0) \in \mathbb{R}^p$, $\forall\ i \in \mathcal{V}$, $\alpha, \beta > 0$ such that $\alpha\beta < 1$, $\overline{R} = [r_{ij}] \in \mathbb{R}^{n\times n}$ satisfies (3.5).

1: $y_i(0) \leftarrow \mathbf{0}_p$, $v_i(0) \leftarrow e_i \in \mathbb{R}^n$ for $i \in \mathcal{V}$
2: **for all** $k = 0, 1, \ldots$ **do**
3: Each $i \in \mathcal{N}$ independently performs:
4: $\bar{x}_i(k) \leftarrow \sum_{j\in\mathcal{N}_i^{in}} r_{ij} x_j(k)$
5: **if** $k > 0$ **then**
6: $y_i(k) \leftarrow y_i(k-1) + \beta\left(x_i(k) - \bar{x}_i(k)\right)$
7: **end if**
8: $x_i(k+1) \leftarrow \bar{x}_i(k) - \alpha\left(\dfrac{\nabla f_i(x_i(k))}{[v_i(k)]_i} + y_i(k)\right)$
9: $v_i(k+1) \leftarrow \sum_{j\in\mathcal{N}_i^{in}} r_{ij} v_j(k)$
10: **end for**

3.2.1 FRSD Algorithm

Consider FRSD displayed in Algorithm 2, at each iteration $k \geq 0$, every agent $i \in \mathcal{V}$ updates three variables $x_i(k)$, $y_i(k) \in \mathbb{R}^p$ and $v_i(k) \in \mathbb{R}^n$, where $\alpha, \beta > 0$ and $\overline{R} = [r_{ij}] \in \mathbb{R}^{n\times n}$ are the parameters of the algorithm: α is the constant step-size and β is a momentum parameter such that $\alpha\beta < 1$, and $\overline{R}$ is a row-stochastic matrix such that

$$r_{ij} = \begin{cases} > 0, & j \in \mathcal{N}_i^{in}, \\ 0, & \text{otherwise;} \end{cases} \qquad \sum_{j\in\mathcal{V}} r_{ij} = 1, \quad \forall\, i \in \mathcal{V}. \tag{3.5}$$

Remark 3.6. Since $\mathcal{G}$ is strongly connected and has finitely many nodes, the Markov chain corresponding to the transition probability matrix $\overline{R}$ is irreducible and positive recurrent; moreover, since $\overline{R}$ has a positive diagonal, it is also aperiodic; therefore, there exists a stationary distribution $\pi \in \mathbb{R}^n$, i.e., $\pi > 0$ and $\mathbf{1}_n^\top \pi = 1$ such that $\pi^\top \overline{R} = \pi^\top$.

Definition 3.7. Each node $i \in \mathcal{V}$, initialized with $v_i(0) = e_i$, generates a sequence $\{v_i(k)\}_{k\geq 0}$ initialized from $v_i(0) = e_i$, using the recursion in `line 9` of the FRSD algorithm. Define $R = \overline{R} \otimes I_p$, $\overline{V}(k) \triangleq \left[v_1(k), \ldots, v_n(k)\right]^\top \in \mathbb{R}^{n\times n}$, i.e., $v_i(k)$ is the i-th row of $\overline{V}(k)$, set $V(k) \triangleq \overline{V}(k) \otimes I_p$ and $\widetilde{V}(k) \triangleq \mathbf{diag}(V(k))$.

Given arbitrary $\mathbf{x}(0) \in \mathbb{R}^{np}$, we initialize $\mathbf{y}(0) \in \mathbb{R}^{np}$ such that $\boldsymbol{y}_i(0) = \mathbf{0}_p$ for $i \in \mathcal{V}$ and $\overline{V}(0) = I_n$. We present FRSD stated in Algorithm 2 in a compact form as follows:

$$\mathbf{x}(k+1) = R\mathbf{x}(k) - \alpha \left(\mathbf{y}(k) + \widetilde{V}^{-1}(k)\nabla f(\mathbf{x}(k))\right), \tag{3.6a}$$

$$\mathbf{y}(k+1) = \mathbf{y}(k) + \beta \left(I_n - R\right)\mathbf{x}(k+1), \tag{3.6b}$$

$$\overline{V}(k+1) = \overline{R}\ \overline{V}(k). \tag{3.6c}$$

Remark 3.8. In the numerical section (Section 3.4), 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 (3.6a) with

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

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 (3.1) 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 comparing our method with other related ones and establishing asymptotic convergence guarantees for FRSD.

3.2.2 Related Methods

Next, we discuss the existing distributed optimization methods for a directed graph $\mathcal{G}$ satisfying Assumption 3.1. In particular, Push-DIGing using column-stochastic weights, $\mathcal{AB}$, $\mathcal{AB}$m, $\mathcal{ABN}$ 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.

3.2.2.1 Push-DIGing

The Push-DIGing algorithm, proposed in [29], achieves a linear convergence rate for solving (3.1) over directed graphs (possibly time-varying) with a constant step-size under Assumptions 3.1-3.3. Given $\mathcal{G}$, Push-DIGing updates four variables $\boldsymbol{x}_i(k), \boldsymbol{y}_i(k), \boldsymbol{z}_i(k) \in \mathbb{R}^p$ and $v_i(k) \in \mathbb{R}$ for each agent $i \in \mathcal{V}$ as follows:

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

where $\alpha > 0$ is the step size and $\overline{B} = \left[b_{ij} \right] \in \mathbb{R}^{n \times n}$ is a column-stochastic matrix compatible with $\mathcal{G}$. The Push-DIGing algorithm is initialized with $v_i(0) = 1$, $\boldsymbol{y}_i(0) = \nabla f_i(\boldsymbol{z}_i(0))$ and from an arbitrary $\boldsymbol{x}_i(0)$ 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 of information pertaining to its in-neighbors.

3.2.2.2 $\mathcal{AB}$/$\mathcal{AB}$m/Push-Pull

In contrast to Push-DIGing, $\mathcal{AB}$ [86] and $\mathcal{AB}$m [87] algorithms could get away with the nonlinear update due to eigenvector estimation. The $\mathcal{AB}$ and $\mathcal{AB}$m[1] methods use both row-stochastic and column-stochastic weights simultaneously. At each iteration $k \geq 0$, they update two variables $\boldsymbol{x}_i(k), \boldsymbol{y}_i(k) \in \mathbb{R}^p$ for each agent $i \in \mathcal{V}$:

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

[1] To present $\mathcal{AB}$ and $\mathcal{AB}$m in a unified manner, we use an adapt-then-combine update for $\boldsymbol{y}_i$ in $\mathcal{AB}$m similar to $\mathcal{AB}$ [86]. In [87], it is mentioned that the $\mathcal{AB}$m method also works with this update; but, the originally stated version of $\mathcal{AB}$m uses an combine-then-adapt update: $\boldsymbol{y}_i(k+1) = \sum_{j \in \mathcal{N}_i^{in}} b_{ij} \boldsymbol{y}_j(k) + \nabla f_j(\boldsymbol{x}_j(k+1)) - \nabla f_j(\boldsymbol{x}_j(k))$ –we also considered the original version for our numerical tests.

where $\alpha > 0$ is the step-size, $\beta \geq 0$ is the momentum parameter, $\overline{R} = [r_{ij}] \in \mathbb{R}^{n\times n}$ and $\overline{B} = \Big[b_{ij}\Big] \in \mathbb{R}^{n\times n}$ 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 $\boldsymbol{x}_i(0)$ and $\boldsymbol{y}_i(0) = \nabla f_i(\boldsymbol{x}_i(0))$ for each $i \in \mathcal{V}$, converges with a linear rate to the optimal solution under Assumptions 3.1-3.3. Setting $\beta > 0$, $\mathcal{AB}$m [87] 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 [88], is related to $\mathcal{AB}$, it is only different in its $\boldsymbol{x}_i(k+1)$ update:

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

while $\boldsymbol{y}_i(k+1)$ 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 [17].

3.2.2.3 Xi-row

The method proposed in [41], which we call as Xi-row in this thesis, can solve (3.1) 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 $\boldsymbol{x}_i(k), \boldsymbol{y}_i(k), \boldsymbol{v}_i(k) \in \mathbb{R}^p$ as follows:

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

where $\alpha > 0$ is the step-size and $\overline{R} = \Big[r_{ij}\Big] \in \mathbb{R}^{n\times n}$ is a row-stochastic matrix compatible with $\mathcal{G}$. The Xi-row iterates are initialized with arbitrary $\boldsymbol{x}_i(0)$, $\boldsymbol{v}_i(0) = \boldsymbol{e}_i$ and $\boldsymbol{y}_i(0) = \nabla f_i(\boldsymbol{x}_i(0))$ for each $i \in \mathcal{V}$. A variant of the Xi-row method, FROST [90] extends Xi-row to handle nonuniform step-sizes.

3.2.2.4 $\mathcal{ABN}$/FROZEN/D-DNGT

$\mathcal{ABN}$ and FROZEN, proposed in [89], 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 $\boldsymbol{x}_i(k), \boldsymbol{y}_i(k), \boldsymbol{s}_i(k) \in \mathbb{R}^p$ for each agent $i \in \mathcal{V}$:

$$\boldsymbol{s}_i(k+1) = \sum_{j \in \mathcal{N}_i^{in}} r_{ij} \boldsymbol{x}_j(k) - \alpha \boldsymbol{y}_i(k) \tag{3.7a}$$

$$\boldsymbol{x}_i(k+1) = \boldsymbol{s}_i(k+1) + \beta \left(\boldsymbol{s}_i(k+1) - \boldsymbol{s}_i(k) \right), \tag{3.7b}$$

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

and FROZEN updates $\boldsymbol{x}_i(k), \boldsymbol{y}_i(k), \boldsymbol{s}_i(k) \in \mathbb{R}^p$, and $\boldsymbol{v}_i(k) \in \mathbb{R}^n$, such that for each agent $i \in \mathcal{V}$, $\boldsymbol{s}_i(k+1)$ and $\boldsymbol{x}_i(k+1)$ are updated according to (3.7a) and (3.7b), respectively, $\boldsymbol{v}_i(k+1)$ and $\boldsymbol{y}_i(k+1)$ are updated according to

$$\boldsymbol{v}_i(k+1) = \sum_{j \in \mathcal{N}_i^{in}} r_{ij} \boldsymbol{v}_j(k),$$

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

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

Another momentum-based method is D-DNGT, proposed in [91]. D-DNGT is related to both FROZEN and $\mathcal{AB}$m:

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

$$\boldsymbol{x}_i(k+1) = \boldsymbol{s}_i(k+1) + \beta \left(\boldsymbol{s}_i(k+1) - \boldsymbol{s}_i(k) \right),$$

$$\boldsymbol{v}_i(k+1) = \sum_{j \in \mathcal{N}_i^{in}} r_{ij} \boldsymbol{v}_j(k),$$

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

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

3.2.2.5 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 $v_i \in \mathbb{R}^n$. 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 $\mathcal{AB}$m methods, which use both row- and column-stochastic matrices, and also with Xi-row which uses row-stochastic weights.

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

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

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

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

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

where for $k \geq 0$, $\widetilde{V}(k) \triangleq \mathbf{diag}(V(k))$ and $V(k) \triangleq \big[v_1(k), ..., v_n(k)\big]^\top \in \mathbb{R}^{n\times n}$ –see Definition 3.7. 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 $k \geq 0$,

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

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

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

where $\Delta_c(k) \triangleq (1-c)(I-R)\boldsymbol{x}(k+1) - R(I-R)\boldsymbol{x}(k)$ 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 $\mathcal{G}$, $\mathcal{AB}$, $\mathcal{AB}$m, Xi-row and FRSD are not the same, they generate distinct iterate sequences.

3.2.2.6 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-DIGing. Using an *implicit* gradient tracking mechanism, FRSD does not need to store the previous iterates for neither $\boldsymbol{x}_i$ nor $\boldsymbol{y}_i$ variables, i.e., each agent-i needs to store only $\boldsymbol{x}_i(k)$, $\boldsymbol{y}_i(k-1)$ and $\boldsymbol{v}_i(k)$ to be able to update these iterates to $\boldsymbol{x}_i(k+1)$, $\boldsymbol{y}_i(k)$ and $\boldsymbol{v}_i(k+1)$; 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 2), leading to *implicit* gradient-tracking, also results 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 $\boldsymbol{v}_j(k) \in \mathbb{R}^n$; 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-DIGing 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 3.1.

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

TABLE 3.1: Comparison of methods for directed graphs in terms of storage and communication requirements.

3.2.3 Primal-Dual Algorithm Motivation

In a similar spirit with the discussion in [95], 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 (3.1):

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

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

$$\min_{\mathbf{x}} \max_{\mathbf{y}\in\mathcal{C}^\perp} \sum_{i\in\mathcal{V}} f_i(\boldsymbol{x}_i) + \boldsymbol{y}_i^\top \boldsymbol{x}_i, \tag{3.8}$$

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 $\sum_{i\in\mathcal{V}} \boldsymbol{y}_i = \mathbf{0}_p$. After switching the roles of $\mathbf{x}$ and $\mathbf{y}$ through multiplying (3.8) with -1, if one naively implements a variant[1] of the primal-dual algorithm proposed by Chambolle & Pock [97], we get

$$\boldsymbol{m}_i(k) \leftarrow (1+\theta)\boldsymbol{y}_i(k) - \theta\boldsymbol{y}_i(k-1), \quad i \in \mathcal{V}, \tag{3.9a}$$

$$\boldsymbol{x}_i(k+1) \leftarrow \boldsymbol{x}_i(k) - \alpha\Big(\nabla f_i(\boldsymbol{x}_i(k)) + \boldsymbol{m}_i(k)\Big), \quad i \in \mathcal{V}, \tag{3.9b}$$

$$\mathbf{y}(k+1) \leftarrow \Pi_{\mathcal{C}^\perp}\Big(\mathbf{y}(k) + \beta\mathbf{x}(k+1)\Big), \tag{3.9c}$$

[1]The variant we discussed here is proposed in [96] – see Eq (5) therein.

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 $\mathbf{y}$, $\Pi_{\mathcal{C}^\perp}(\mathbf{y}) = \mathbf{y} - \mathbf{1}\mathbf{1}^\top\mathbf{y}/n$. This explicit form of $\Pi_{\mathcal{C}^\perp}(\cdot)$ and $\mathbf{y}(0) = \mathbf{0}_n$ initialization imply that (3.9c) is equivalent to $\mathbf{y}(k+1) \leftarrow \mathbf{y}(k) + \Pi_{\mathcal{C}^\perp}(\mathbf{x}(k+1))$. This method is known to converge with a linear rate for appropriately chosen $\alpha, \beta, \theta > 0$; however, due to $\Pi_{\mathcal{C}^\perp}(\cdot)$ in (3.9c), 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 $W_\infty \triangleq \lim_{k\to\infty} W^k = \mathbf{1}_n\mathbf{1}_n^\top/n$. Note that $\Pi_{\mathcal{C}^\perp}(\mathbf{y}) = (I - W_\infty)\mathbf{y}$. For decentralized implementation, consider approximating $\Pi_{\mathcal{C}^\perp}(\cdot)$ with $(I - W)(\cdot)$. Thus, we will approximate the update in (3.9c) with $\mathbf{y}(k+1) \leftarrow \mathbf{y}(k) + (I - W)\mathbf{x}(k+1)$, and with this approximation, the recursion in (3.9) takes the following form:

$$\mathbf{y}(k) \leftarrow \mathbf{y}(k-1) + \beta(I - W)\mathbf{x}(k), \tag{3.10a}$$

$$\mathbf{x}(k+1) \leftarrow \mathbf{x}(k) - \alpha\Big(\nabla f(\mathbf{x}(k)) + \mathbf{y}(k) + \theta\beta(I - W)\mathbf{x}(k)\Big), \tag{3.10b}$$

where $\nabla f(\mathbf{x}) = [\nabla f_1(\boldsymbol{x}_1)^\top, \ldots, \nabla f_n(\boldsymbol{x}_n)^\top]^\top$. Hence, adding and subtracting $W\mathbf{x}(k)$ to (3.10b), we get

$$\mathbf{x}(k+1) \leftarrow W\mathbf{x}(k) - \alpha\Big(\nabla f(\mathbf{x}(k)) + \mathbf{y}(k)\Big) + (1 - \alpha\beta\theta)(I - W)\mathbf{x}(k). \tag{3.11}$$

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 (3.11) disappears as $1 - \alpha\beta\theta = 0$, the primal-dual algorithm with approximate averaging in (3.10) reduces to

$$\mathbf{y}(k) \leftarrow \mathbf{y}(k-1) + \beta\Big(\mathbf{x}(k) - W\mathbf{x}(k)\Big), \tag{3.12a}$$

$$\mathbf{x}(k+1) \leftarrow W\mathbf{x}(k) - \alpha\Big(\nabla f(\mathbf{x}(k)) + \mathbf{y}(k)\Big). \tag{3.12b}$$

It can be clearly seen that the FRSD algorithm for directed networks can be obtained from (3.12) 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 $R_\infty = \lim_{k\to\infty} R^k = \mathbf{1}_n\boldsymbol{\pi}^\top$ for some $\boldsymbol{\pi} \in \mathbb{R}^n$ such that $\boldsymbol{\pi} > \mathbf{0}_n$ and $\mathbf{1}_n^\top\boldsymbol{\pi} = 1$.

3.3 Convergence Results

In this section, we will show that the iterate sequence generated by the algorithm FRSD as stated in (3.6) 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 thesis.

Remark 3.9. Assumptions 3.2 and 3.3 imply that f is L-smooth, i.e., $\left\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{x}')\right\| \leq L\left\|\mathbf{x} - \mathbf{x}'\right\|$, and μ-strongly convex.

Remark 3.10. Since $\overline{R}$ is row-stochastic, the spectral radius of $\overline{R}$ is 1, $\rho(\overline{R}) = 1$; thus, $\lim_{k\to\infty} \overline{R}^k$ exists. In particular, since $\overline{R}$ corresponds to an ergodic Markov chain, we get $\lim_{k\to\infty} \overline{R}^k = \mathbf{1}_n \boldsymbol{\pi}^\top$ – see Remark 3.6.

Definition 3.11. Define $V_\infty \triangleq \lim_{k\to\infty} V(k)$ and $\widetilde{V}_\infty \triangleq \mathbf{diag}(V_\infty)$. Furthermore, let $v \triangleq \sup_{k\geq 0} \|V(k)\|$ and $\tilde{v} \triangleq \sup_{k\geq 0} \|\widetilde{V}^{-1}(k)\|$.

Remark 3.12. Since $\overline{V}(0) = I_n$, $\lim_{k\to\infty} \overline{R}^k = \mathbf{1}_n \boldsymbol{\pi}^\top$, we get $V_\infty = (\mathbf{1}_n \boldsymbol{\pi}^\top) \otimes I_p$ and $\widetilde{V}_\infty = \mathbf{diag}(V_\infty) = \boldsymbol{\pi} \otimes \mathbf{1}_p$. Note $\{V(k)\}_k$ is convergent; hence, it is bounded, implying that $v \in \mathbb{R}$ exists. Furthermore, Remark 3.6 shows that $\boldsymbol{\pi} > 0$; therefore, $\tilde{v} \in \mathbb{R}$ also exists.

Remark 3.13. Since $\overline{R}$ corresponds to an Ergodic Markov chain, Remarks 3.6 and 3.10 imply that $V_\infty R = R V_\infty = V_\infty V_\infty = V_\infty$. Moreover, the spectral radius of $R - V_\infty$ satisfies $\rho(R - V_\infty) = \rho(\overline{R} - \mathbf{1}_n \boldsymbol{\pi}^\top) < 1$, which is proved as follow:
Spectral radius of $\overline{R}$ equal to 1 follows from [98, Lemma 3.4]. Assume that we have $\boldsymbol{v}^\top(\overline{R} - \mathbf{1}_n \boldsymbol{\pi}^\top) = \lambda \boldsymbol{v}^\top$ for some $\lambda, \boldsymbol{v} \neq 0$. Since $\mathbf{1}_n$ is a right eigenvector of $(\overline{R} - \mathbf{1}_n \boldsymbol{\pi}^\top)$ corresponding to eigenvalue 0, $\boldsymbol{v}^\top \mathbf{1}_n = 0$ according to [99, Theorem 1.4.7]. Therefore, λ is also an eigenvalue of $\overline{R}$ due to $\boldsymbol{v}^\top \overline{R} = \lambda \boldsymbol{v}^\top$. From remark 3.6, we have $\boldsymbol{\pi}^\top \mathbf{1}_n = 1$, and then $\boldsymbol{v}^\top \neq \boldsymbol{\pi}^\top$, hence $\lambda < 1$. As a consequence, $\rho(\overline{R} - \mathbf{1}_n \boldsymbol{\pi}^\top) < 1$.

Next, we define some auxiliary sequences that will be used in the analysis. For $k \geq 0$, let $\hat{\mathbf{x}}(k) \triangleq V_\infty \mathbf{x}(k) = (\mathbf{1}_n \otimes I_p)(\boldsymbol{\pi}^\top \otimes I_p)\mathbf{x}(k) = (\mathbf{1}_n \otimes I_p)\hat{\boldsymbol{x}}(k) \in \mathbb{R}^{np}$, where $\hat{\boldsymbol{x}}(k) \triangleq (\boldsymbol{\pi}^\top \otimes I_p)\mathbf{x}(k) \in \mathbb{R}^p$, i.e., $\hat{\mathbf{x}}(k) = \mathbf{1}_n \otimes \hat{\boldsymbol{x}}(k)$. Let $\mathbf{x}^* \triangleq \mathbf{1}_n \otimes \boldsymbol{x}^*$ where $\boldsymbol{x}^* \in \mathbb{R}^p$ is the unique optimal solution to (3.1). Thus, Definition 3.5 implies that $\nabla f(\hat{\mathbf{x}}(k)) = \left[\nabla f_1(\hat{\mathbf{x}}(k))^\top, \ldots, \nabla f_n(\hat{\mathbf{x}}(k))^\top\right]^\top \in \mathbb{R}^{np}$ and $\nabla f(\mathbf{x}^*) = \left[\nabla f_1(\boldsymbol{x}^*)^\top, \ldots, \nabla f_n(\boldsymbol{x}^*)^\top\right]^\top \in \mathbb{R}^{np}$.

Remark 3.14. From the optimality condition for (3.1), $(\mathbf{1}_n^\top \otimes I_p)\nabla f(\mathbf{x}^*) = 0$.

The structure of our proof was inspired by [41] and [31]. In particular, we construct a linear system of inequalities and use the deterministic version of the celebrated supermartingale convergence theorem [100] 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 [86, 88, 91].

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

Definition 3.15. Given $\alpha, \beta > 0$ such that $\alpha\beta \in (0,1)$, let $C = \overline{C} \otimes I_p$ and $\overline{C} \triangleq (1-\alpha\beta)I_n + \alpha\beta\overline{R}$, where $\overline{R} = [r_{ij}] \in \mathbb{R}^{n\times n}$ is the row-stochastic matrix as given in (3.5).

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

Lemma 3.16. *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*

$$\|R\mathbf{x} - \hat{\mathbf{x}}\|_R \leq \sigma_R \|\mathbf{x} - \hat{\mathbf{x}}\|_R, \tag{3.13}$$

$$\|C\mathbf{x} - \hat{\mathbf{x}}\|_C \leq \sigma_C \|\mathbf{x} - \hat{\mathbf{x}}\|_C, \tag{3.14}$$

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

Proof. Since $\overline{R}$ is irreducible, row-stochastic, and has positive diagonals, the Perron-Frobenius theorem states that $\rho(\overline{R}) = 1$, every eigenvalue of $\overline{R}$ other than 1 is strictly less than $\rho(\overline{R}) = 1$, and $\boldsymbol{\pi}$ is a strictly positive left eigenvector corresponding to the eigenvalue 1 where $\mathbf{1}_n^\top \boldsymbol{\pi} = 1$ obtains. Therefore, $\lim_{k\to\infty} \overline{R}^k = \mathbf{1}_n \boldsymbol{\pi}^\top$. Moreover, we have

$$\lim_{k\to\infty} R^k = \left(\lim_{k\to\infty} \overline{R}^k\right) \otimes I_p = \left(\mathbf{1}_n \boldsymbol{\pi}^\top\right) \otimes I_p = V_\infty$$

As a result,

$$RV_\infty = \left(\overline{R} \otimes I_p\right)\left(\left(\mathbf{1}_n \boldsymbol{\pi}^\top\right) \otimes I_p\right) = \left(\left(\mathbf{1}_n \boldsymbol{\pi}^\top\right) \otimes I_p\right) = V_\infty$$

$$V_\infty V_\infty = \left(\left(\mathbf{1}_n \boldsymbol{\pi}^\top\right) \otimes I_p\right)\left(\left(\mathbf{1}_n \boldsymbol{\pi}^\top\right) \otimes I_p\right) = \left(\left(\mathbf{1}_n \boldsymbol{\pi}^\top\right) \otimes I_p\right) = V_\infty$$

Therefore,

$$R\mathbf{x} - \hat{\mathbf{x}} = Rx - V_\infty \mathbf{x} = \left(R - V_\infty\right)\left(\mathbf{x} - V_\infty \mathbf{x}\right) \tag{3.15}$$

Indeed, since

$$\rho\Big(R - V_\infty\Big) = \rho\Big(R - \Big(\Big(\mathbf{1}_n\boldsymbol{\pi}^\top\Big) \otimes I_p\Big)\Big) = \rho\Big(\Big(\overline{R} - \Big(\mathbf{1}_n\boldsymbol{\pi}^\top\Big)\Big) \otimes I_p\Big) < 1,$$

according to Lemma 5.6.10 in [99], there exists a matrix norm, let's call it $|||\cdot|||$, such that

$$\sigma_R = |||R - V_\infty||| < 1$$

Moreover, according to Theorem 5.7.13 in [99], for any matrix norm, $|||\cdot|||$, there exists a compatible vector norm, $\|\cdot\|_R$, such that $\|B\mathbf{x}\|_R \leq |||B|||\,\|\mathbf{x}\|_R$ for all matrices, B, and all vectors, $\mathbf{x}$; therefore, Equation (3.15) leads to

$$\begin{aligned}\|R\mathbf{x} - \hat{\mathbf{x}}\|_R &= \Big\|\Big(R - V_\infty\Big)\Big(\mathbf{x} - V_\infty\mathbf{x}\Big)\Big\|_R \\ &\leq |||R - V_\infty|||\,\|\mathbf{x} - V_\infty\mathbf{x}\|_R \\ &= \sigma_R\,\|\mathbf{x} - \hat{\mathbf{x}}\|_R\end{aligned}$$

,and (3.14) can be shown similarly since $\lim_{k\to\infty} C^k = \lim_{k\to\infty} R^k = (\mathbf{1}_n\boldsymbol{\pi}^\top) \otimes I_p$, and same arguments can be used for showing (3.14) as we also have $\rho(C - V_\infty) < 1$. □

Remark 3.17. Let $|||\cdot|||$ represent the matrix norm induced by $\|\cdot\|_R$. According to [99, Lemma 5.6.10], the constant $\sigma_R \in (0,1)$ in Lemma 3.16 has an explicit form, $\sigma_R = |||R - V_\infty|||$.

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

$$\begin{aligned}\|\cdot\|_R \leq \kappa_1\|\cdot\|_C, \qquad \|\cdot\|_C \leq \kappa_2\|\cdot\|_R, \\ \|\cdot\|_R \leq \kappa_3\|\cdot\|, \qquad \|\cdot\|_C \leq \kappa_4\|\cdot\|.\end{aligned} \tag{3.16}$$

Similar to the results in [40], 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 3.18. *Let $V(k) = R^k$ for $k \geq 0$ and $V_\infty = \lim_{k\to\infty} R^k$. Then, for $\kappa_3 > 0$ defined in (3.16) and $\sigma_R \in (0,1)$ given in Remark 3.17, the following bound holds:*

$$\|\overline{V}(k) - \mathbf{1}_n\boldsymbol{\pi}^\top\| = \|V(k) - V_\infty\| \leq \kappa_3\sigma_R^k, \quad \forall\, k \geq 0. \tag{3.17}$$

Proof. It immediately follows from Remark 3.13 that

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

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

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

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 (3.17) follows from the fact that singular values of $\overline{V}(k) - \mathbf{1}_n \boldsymbol{\pi}^\top$ and $V(k) - V_\infty$ are the same. □

Lemma 3.19. *The following inequalities hold for all $k \geq 0$:*

$$\|\widetilde{V}^{-1}(k) - \widetilde{V}_\infty^{-1}\| \leq \tilde{v}^2 \sqrt{n} \kappa_3 \sigma_R^k \tag{3.18a}$$

$$\|\widetilde{V}^{-1}(k) - \widetilde{V}^{-1}(k-1)\| \leq 2\tilde{v}^2 \sqrt{n} \kappa_3 \sigma_R^k. \tag{3.18b}$$

Proof. The proof follows from [41, Lemma 3]. Indeed, note that $\widetilde{V}^{-1}(k) - \widetilde{V}_\infty^{-1} = \widetilde{V}^{-1}(k)(\widetilde{V}(k) - \widetilde{V}_\infty)\widetilde{V}_\infty^{-1}$; hence, $\left\|\widetilde{V}^{-1}(k) - \widetilde{V}_\infty^{-1}\right\| \leq \tilde{v}^2 \left\|\mathbf{diag}\left(V(k) - V_\infty\right)\right\| \leq \tilde{v}^2 \sqrt{n} \|\bar{V}(k) - \mathbf{1}_n \boldsymbol{\pi}^\top\|$, 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 3.18. □

Lemma 3.20. *The following inequality holds for all $k \geq 0$:*

(a) $\|V_\infty \widetilde{V}^{-1}(k) \nabla f(\mathbf{x}(k))\| \leq v\tilde{v}^2 \sqrt{n} \kappa_3 \sigma_R^k \|\nabla f(\mathbf{x}(k))\| + nL\|\mathbf{x}(k) - \hat{\mathbf{x}}(k)\|_C + nL\|\hat{\mathbf{x}}(k) - \mathbf{x}^*\|$

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

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

(d) $\|\widetilde{V}^{-1}(k) \nabla f(\mathbf{x}(k)) - \widetilde{V}^{-1}(k-1) \nabla f(\mathbf{x}(k-1))\| \leq \tilde{v} L \|\mathbf{x}(k) - \mathbf{x}(k-1)\|_R + 2\tilde{v}^2 \sqrt{n} \kappa_3 \sigma_R^k \|\nabla f(\mathbf{x}(k))\|.$

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

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

which together with triangular inequality implies (a), where $\widetilde{V}_\infty = \mathbf{diag}(V_\infty)$ –see Definition 3.11. In the second inequality, we use Remark 3.14, and the third inequality

follows from (3.18a) in Lemma 3.19 and we also use Remark 3.9 along with $V_\infty \widetilde{V}_\infty^{-1} = (\mathbf{1}_n \otimes \boldsymbol{I_p})(\mathbf{1}_n^\top \otimes \boldsymbol{I_p}) = (\mathbf{1}_n \mathbf{1}_n^\top) \otimes I_p$; hence, $\left\| (\mathbf{1}_n \mathbf{1}_n^\top) \otimes I_p \right\| = n$. Next, we prove part (b):

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

hence, the part (b) follows from (3.18b) in Lemma 3.19 and from part (a) of Lemma 3.20.

Now we consider part (c). Since $\mathbf{y}(0) = \mathbf{0}_{np}$, it follows from (3.6b) that $\mathbf{y}(k) = \beta(I_{np} - R)\sum_{\ell=1}^{k} \mathbf{x}(\ell)$. Since $V_\infty R = V_\infty$ – see Remark 3.13, we have $V_\infty \mathbf{y}(k) = \mathbf{0}_{np}$ for all $k \geq 0$ 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 (3.6a), we get

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

Therefore, the part (c) immediately follows from using Remark 3.9 and the part (b) of Lemma 3.20 on

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

Finally, we consider part (d).

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

Hence, the part (d) follows from (3.18b) of Lemma 3.19 and Remark 3.9. □

For the sake of completeness we provide another technical result –for its proof, see [31, Lemma 10].

Lemma 3.21. *Given $\alpha \in (0, \frac{2}{nL})$, let $\eta \triangleq \max\{|1 - nL\alpha|, |1 - n\mu\alpha|\}$. If Assumptions 3.2 and 3.3 hold, then for all $\boldsymbol{x} \in \mathbb{R}^p$, one has*

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

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

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

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

Proof. Using (3.6a) 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:

$$\begin{aligned}
&\|\mathbf{x}(k+1)-\hat{\mathbf{x}}(k+1)\|_C \\
=\ & \|R\mathbf{x}(k)-\alpha\mathbf{y}(k)-\alpha\widetilde{V}^{-1}(k)\nabla f(\mathbf{x}(k))-\hat{\mathbf{x}}(k)+\alpha V_\infty\widetilde{V}^{-1}(k)\nabla f(\mathbf{x}(k))\|_C \\
=\ & \|R\mathbf{x}(k)-R\mathbf{x}(k-1)+\mathbf{x}(k)-\hat{\mathbf{x}}(k)-\alpha\beta(I_n-R)\mathbf{x}(k)+\alpha\widetilde{V}^{-1}(k-1)\nabla f(\mathbf{x}(k-1)) \\
&-\alpha\widetilde{V}^{-1}(k)\nabla f(\mathbf{x}(k))+\alpha V_\infty\widetilde{V}^{-1}(k)\nabla f(\mathbf{x}(k))\|_C \\
\leq\ & \|\Big((1-\alpha\beta)I_n+\alpha\beta R\Big)\mathbf{x}(k)-\hat{\mathbf{x}}(k)\|_C+\kappa_2\|R\mathbf{x}(k)-R\mathbf{x}(k-1)\|_R \\
&+\alpha\kappa_4\|\widetilde{V}^{-1}(k)\nabla f(\mathbf{x}(k))-\widetilde{V}^{-1}(k-1)\nabla f(\mathbf{x}(k-1))\|+\alpha\kappa_4\|V_\infty\widetilde{V}^{-1}(k)\nabla f(\mathbf{x}(k))\|,
\end{aligned} \tag{3.19}$$

where in the second equality we first use (3.6b) to represent $\mathbf{y}(k)$ in terms of $\mathbf{x}(k)$ and $\mathbf{y}(k-1)$, and next we use (3.6a) to get rid of the term $-\alpha\mathbf{y}(k-1)$.

Next, using (3.14) of Lemma 3.1, we can bound the first term on the right-hand-side of (3.19) as follows:

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

where C is given in Definition 3.15. Clearly, we can also bound the second term in (3.19) with $|||R|||\,\|\mathbf{x}(k)-\mathbf{x}(k-1)\|_R$. Finally, using parts (d) and (a) of Lemma 3.20 for the third and the fourth terms, respectively, we get the desired result. □

Remark 3.23. 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 $\mathcal{AB}$, Push-DIGing, Xi-row methods.

Lemma 3.24. *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 (3.6a) 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

$$\begin{aligned}&\|\hat{\mathbf{x}}(k+1) - \mathbf{x}^*\| \\ =\ & \|\hat{\mathbf{x}}(k) - \alpha V_\infty \widetilde{V}^{-1}(k)\nabla f(\mathbf{x}(k)) - \mathbf{x}^*\| \\ \leq\ & \alpha\|\Big((\mathbf{1}_n\mathbf{1}_n^\top)\otimes I_p\Big)\nabla f(\hat{\mathbf{x}}(k)) - V_\infty\widetilde{V}^{-1}(k)\nabla f(\mathbf{x}(k))\| \\ & + \|\hat{\mathbf{x}}(k) - \alpha\Big((\mathbf{1}_n\mathbf{1}_n^\top)\otimes I_p\Big)\nabla f(\hat{\mathbf{x}}(k)) - \mathbf{x}^*\|. \end{aligned} \tag{3.20}$$

Now, we bound the first term on the right-hand-side of (3.20):

$$\begin{aligned}&\|\Big((\mathbf{1}_n\mathbf{1}_n^\top)\otimes I_p\Big)\nabla f(\hat{\mathbf{x}}(k)) - V_\infty\widetilde{V}^{-1}(k)\nabla f(\mathbf{x}(k))\| \\ \leq\ & \|\Big((\mathbf{1}_n\mathbf{1}_n^\top)\otimes I_p\Big)\nabla f(\hat{\mathbf{x}}(k)) - V_\infty\widetilde{V}_\infty^{-1}\nabla f(\mathbf{x}(k))\| \\ & + \|V_\infty\widetilde{V}^{-1}(k)\nabla f(\mathbf{x}(k)) - V_\infty\widetilde{V}_\infty^{-1}\nabla f(\mathbf{x}(k))\| \\ \leq\ & nL\|\mathbf{x}(k) - \hat{\mathbf{x}}(k)\|_C + v\tilde{v}^2\sqrt{n}\kappa_3\sigma_R^k\|\nabla f(\mathbf{x}(k))\|, \end{aligned} \tag{3.21}$$

where we used $V_\infty\widetilde{V}_\infty^{-1} = (\mathbf{1}_n\mathbf{1}_n^\top)\otimes I_p$, Assumption 3.2 and Lemma 3.19. Next, the second term on the right-hand-side of (3.20) can be bounded using Lemma 3.21:

$$\begin{aligned}\|\hat{\mathbf{x}}(k) - \alpha\Big((\mathbf{1}_n\mathbf{1}_n^\top)\otimes I_p\Big)\nabla f(\hat{\mathbf{x}}(k)) - \mathbf{x}^*\| =& \|\mathbf{1}_n\otimes\Big(\hat{x}(k) - x^* - \alpha\sum_{i=1}^n\nabla f_i(\hat{x}(k))\Big)\| \\ \leq& \eta\|\mathbf{1}_n\otimes\Big(\hat{x}(k) - x^*\Big)\| = \eta\|\hat{\mathbf{x}}(k) - \mathbf{x}^*\|, \end{aligned} \tag{3.22}$$

where $\eta = \max\{|1 - nL\alpha|, |1 - n\mu\alpha|\}$, $\mathbf{x}^* = \mathbf{1}_n\otimes x^*$, and $\hat{x}(k) \triangleq (\boldsymbol{\pi}^\top\otimes I_p)\mathbf{x}(k)$, i.e., $\hat{\mathbf{x}}(k) = \mathbf{1}_n\otimes\hat{x}(k)$. Finally, Lemma 3.24 follows from (3.20)-(3.22). □

Lemma 3.25. *The following inequality holds for all $k \geq 0$:*

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

Proof. We use (3.6a) and (3.6b) for rewriting $\mathbf{x}(k+1)$ and $\mathbf{y}(k)$ respectively, to derive the first two equations:

$$\begin{aligned}
& \|\mathbf{x}(k+1)-\mathbf{x}(k)\|_R \qquad (3.23)\\
= \; & \|R\mathbf{x}(k)-\alpha\mathbf{y}(k)-\alpha\widetilde{V}^{-1}(k)\nabla f(\mathbf{x}(k))-\mathbf{x}(k)\|_R\\
= \; & \|R\mathbf{x}(k)-\alpha\mathbf{y}(k-1)-\alpha\beta(I_n-R)\mathbf{x}(k)-\alpha\widetilde{V}^{-1}(k)\nabla f(\mathbf{x}(k))-\mathbf{x}(k)\|_R\\
= \; & \|R(\mathbf{x}(k)-\mathbf{x}(k-1))+\alpha\widetilde{V}^{-1}(k-1)\nabla f(\mathbf{x}(k-1))-\alpha\beta(I_n-R)\mathbf{x}(k)-\alpha\widetilde{V}^{-1}(k)\nabla f(\mathbf{x}(k))\|_R\\
\leq \; & \|R\mathbf{x}(k)-R\mathbf{x}(k-1)-\hat{\mathbf{x}}(k)+\hat{\mathbf{x}}(k-1)\|_R+\kappa_3\|\hat{\mathbf{x}}(k)-\hat{\mathbf{x}}(k-1)\|+\alpha\beta\|(I_n-R)\mathbf{x}(k)\|_R\\
& +\alpha\kappa_3\|\widetilde{V}^{-1}(k)\nabla f(\mathbf{x}(k))-\widetilde{V}^{-1}(k-1)\nabla f(\mathbf{x}(k-1))\|
\end{aligned}$$

where in the third equation, we use (3.6a) to get rid of the term $-\alpha\mathbf{y}(k-1)$ as we did previously to derive (3.19). We bound the first term above using Remark 3.17, i.e.,

$$\begin{aligned}
& \|R\mathbf{x}(k)-R\mathbf{x}(k-1)-\hat{\mathbf{x}}(k)+\hat{\mathbf{x}}(k-1)\|_R\\
= \; & \|(R-V_\infty)(\mathbf{x}(k)-\mathbf{x}(k-1))\|_R\leq\sigma_R\|\mathbf{x}(k)-\mathbf{x}(k-1)\|_R. \qquad (3.24)
\end{aligned}$$

We can use Lemma 3.20 (c) to bound $\|\hat{\mathbf{x}}(k)-\hat{\mathbf{x}}(k-1)\|$, and Lemma 3.20 (d) to bound the fourth term. Then, the remaining third term in (3.23) 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 fist 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. □

Combining the results of Lemmas 3.22, 3.24 and 3.25, we will construct a linear dynamical system to prove the linear convergence of the proposed algorithm. For the sake of notational simplicity, we define some constants below:

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

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:

$$\theta_{k+1} \leq \mathrm{Y}_{\alpha,\beta}\, \theta_k + \Phi_k \Psi_k, \quad \forall\, k \geq 0, \tag{3.25}$$

where θ_k, Φ_k, Ψ_k and $\mathrm{Y}_{\alpha,\beta} \triangleq \mathrm{Y}_1(\alpha,\beta) + \alpha \mathrm{Y}_2$ are defined as

$$\theta_k = \begin{bmatrix} \|\mathbf{x}(k) - \hat{\mathbf{x}}(k)\|_C \\ \|\hat{\mathbf{x}}(k) - \boldsymbol{x}^*\| \\ \|\mathbf{x}(k) - \mathbf{x}(k-1)\|_R \end{bmatrix}, \qquad \Phi_k = \sigma_R^k \alpha \begin{bmatrix} p_1 & 0 & 0 \\ p_2 & 0 & 0 \\ p_3 & 0 & 0 \end{bmatrix}, \qquad \Psi_k = \begin{bmatrix} \|\nabla f(\mathbf{x}(k))\| \\ 0 \\ 0 \end{bmatrix},$$

$$\mathrm{Y}_1(\alpha,\beta) = \begin{bmatrix} \sigma_C & 0 & \kappa_2 \|\!|R|\!\| \\ 0 & 1 & 0 \\ \alpha s(\beta) & 0 & \sigma_R \end{bmatrix}, \qquad \mathrm{Y}_2 = \begin{bmatrix} s_1 & s_1 & s_2 \\ s_3 & -n\mu & 0 \\ s_4 & s_4 & s_5 \end{bmatrix}.$$

Theorem 3.26. *Suppose Assumptions 3.1-3.3 holds. Let $\alpha, \beta > 0$ such that $\alpha \in (0, \bar{\alpha})$ and $\alpha\beta < 1$, where $\bar{\alpha} > 0$ is defined as*

$$\begin{aligned} \bar{\alpha} \triangleq \sup_{\delta_1,\delta_2} \min & \left\{ \frac{(1-\sigma_C) - \kappa_2 \|\!|R|\!\| \delta_2}{s_1(1+\delta_1) + s_2\delta_2}, \frac{(1-\sigma_R)\delta_2}{s_4(1+\delta_1) + s_5\delta_2 + s(\beta)}, \frac{1}{nL} \right\} \\ s.t. \quad & \frac{L}{\mu} < \delta_1, \qquad 0 < \delta_2 < \frac{1-\sigma_c}{\kappa_2 \|\!|R|\!\|}. \end{aligned} \tag{3.26}$$

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

Proof. Given $\alpha \in (0, \frac{2}{nL})$ and $\beta > 0$ such that $\alpha\beta < 1$, it follows from Lemmas 3.22-3.25 that (3.25) holds for $k \geq 0$. Next, we show $\rho(\mathrm{Y}_{\alpha,\beta}) < 1$. Since $\mathrm{Y}_{\alpha,\beta}$ has all non-negative entries, it is sufficient to show that $\mathrm{Y}_{\alpha,\beta}\, \gamma < \gamma$ for some positive $\gamma = [\gamma_1, \gamma_2, \gamma_3]^\top \in \mathbb{R}^3_+$ –see [99, Corollary 8.1.29]. Since $L \geq \mu$, according to the definition of η in Lemma 3.21, $\eta = 1 - \alpha n \mu$ for $\alpha \in (0, \frac{1}{nL})$. Hence, $\mathrm{Y}_{\alpha,\beta}\, \gamma < \gamma$ is equivalent to

$$(s_1\gamma_1 + s_1\gamma_2 + s_2\gamma_3)\alpha < \gamma_1(1-\sigma_C) - \kappa_2 \|\!|R|\!\| \gamma_3, \tag{3.27a}$$

$$s_3\gamma_1\alpha - \gamma_2 n\mu\alpha < 0, \tag{3.27b}$$

$$\Big((s_4 + s(\beta))\gamma_1 + s_4\gamma_2 + s_5\gamma_3\Big)\alpha < \gamma_3(1-\sigma_R). \tag{3.27c}$$

Clearly, (3.27) 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 (3.26); thus, we get $\rho(\mathrm{Y}_{\alpha,\beta}) < 1$. □

Remark 3.27. Note δ_1 and δ_2 are only required to satisfy (3.26). 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 (3.26). The supremum over δ_1 subject to (3.26) 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

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

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 [100, 101].

Lemma 3.28. *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 3.29. *Let Assumptions 3.1-3.3 hold. For any step-size $\alpha \in (0, \bar{\alpha})$, the sequence $\{\mathbf{x}(k)\}$ converges with a linear rate to $\mathbf{x}^* = \mathbf{1}_n \otimes \boldsymbol{x}^*$ with a rate arbitrarily close to $\varphi_{\alpha,\beta} \triangleq \max\{\rho(\mathrm{Y}_{\alpha,\beta}), \sigma_R\} \in (0,1)$, where $\bar{\alpha}$ is defined in the Theorem 3.26, and $\sigma_R = \left|\left|\left| \left(\overline{R} - (\mathbf{1}_n \boldsymbol{\pi}^\top)\right) \otimes I_p \right|\right|\right| < 1$.*

Proof. The proof follows from similar arguments as in the proof of [41, Lemma 5]. Theorem 3.26 shows that $\rho(\mathrm{Y}_{\alpha,\beta}) < 1$; hence, from [99, Lemma 5.6.10], given any positive $\zeta < 1 - \varphi_{\alpha,\beta}$, there exists a matrix norm $\|\cdot\|_{(\zeta)}$ such that $\left\|\mathrm{Y}_{\alpha,\beta}\right\|_{(\zeta)} \leq \rho(\mathrm{Y}_{\alpha,\beta}) + \frac{\zeta}{2}$. Since norms are equivalent in finite-dimensional spaces, we conclude that there exists some $\Gamma > 0$ such that we have

$$\|\mathrm{Y}_{\alpha,\beta}^k\| \leq \Gamma \tilde{\lambda}^k, \qquad \|\mathrm{Y}_{\alpha,\beta}^{k-j-1}\Phi_j\| \leq \Gamma \tilde{\lambda}^k, \tag{3.28}$$

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 (3.25) recursively, we get, for all $k \geq 0$,

$$\theta_k \leq \Upsilon_{\alpha,\beta}^k \theta_0 + \sum_{j=0}^{k-1} \Upsilon_{\alpha,\beta}^{k-j-1} \Phi_j \Psi_j. \tag{3.29}$$

Since all the terms in (3.29) have non-negative entries, using (3.28), we get for all $k \geq 0$,

$$\|\theta_k\| \quad \leq \quad \|\Upsilon_{\alpha,\beta}^k\| \|\theta_0\| + \sum_{j=0}^{k-1} \|\Upsilon_{\alpha,\beta}^{k-j-1} \Phi_j\| \|\Psi_j\| \leq \quad \Gamma \tilde{\lambda}^k \Big(\|\theta_0\| + \sum_{j=0}^{k-1} \|\Psi_j\| \Big). \tag{3.30}$$

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

$$\begin{aligned} \|\Psi_k\| \quad &\leq \quad \|\nabla f(\mathbf{x}(k)) - \nabla f(\mathbf{x}^*)\| + \|\nabla f(\mathbf{x}^*)\| \\ &\leq \quad L\|\mathbf{x}(k) - \hat{\mathbf{x}}(k)\| + L\|\hat{\mathbf{x}}(k) - \mathbf{x}^*\| + \|\nabla f(\mathbf{x}^*)\| \\ &\leq \quad 2L\|\theta_k\| + \|\nabla f(\mathbf{x}^*)\|. \end{aligned} \tag{3.31}$$

Thus, for all $k \geq 0$, combining (3.30) and (3.31) 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

$$\|\theta_k\| = a_{k+1} - a_k \leq (\tilde{c} a_k + \tilde{d}_k)\tilde{\lambda}^k, \quad \forall\, k \geq 0. \tag{3.32}$$

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 3.28 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, (3.32) leads to

$$\lim_{k\to\infty} \frac{\|\theta_k\|}{(\tilde{\lambda} + \frac{\zeta}{2})^k} \leq \frac{(\tilde{c} a_k + \tilde{d}_k)\tilde{\lambda}^k}{(\tilde{\lambda} + \frac{\zeta}{2})^k} = 0. \tag{3.33}$$

Thus, there exist $c > 0$ such that

$$\|\theta_k\| \leq c(\tilde{\lambda} + \tfrac{\zeta}{2})^k, \quad \forall k \geq 0, \tag{3.34}$$

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

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

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

3.4 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 (3.1) to test FRSD method: *(a)* regression with Huber loss, *(b)* the logistic regression, which are described in Sections 3.4.1 and 3.4.2, respectively. Throughout the experiments, we use the uniform weights for the row-stochastic matrix defined in (3.5), i.e., $r_{ij} = 1/|\mathcal{N}_i^{in}|$ for all $i \in \mathcal{V}$, and we use coordinated step-size and momentum parameters for $\mathcal{AB}$m, $\mathcal{ABN}$, FROZEN and D-DNGT to have a fair comparison with other methods.

For both distributed regression problems, we compare FRSD with Xi-row [41], FROZEN [89] and D-DNGT [91], which use only *row-stochastic* weights similar to our method, with Push-DIGing [29], which utilizes *column-stochastic* weights, and also with $\mathcal{AB}$ [86], $\mathcal{AB}$m [87], $\mathcal{ABN}$ [89] and Push-Pull [88], 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 3.1.

Furthermore, in Section 3.4.2.2, 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.

3.4.1 Distributed Regression with Huber Loss

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

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

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

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

$$\boldsymbol{x}^* \in \arg\min_{\boldsymbol{x}\in\mathbb{R}^p} \bar{f}(\boldsymbol{x}) \triangleq \frac{1}{n}\sum_{i\in\mathcal{V}} \mathbf{H}_{\xi}(M_i\boldsymbol{x} - \boldsymbol{b}_i). \tag{3.35}$$

In the experiments, we solve (3.35) over the set of directed graphs shown in Figure 3.1. We generate data as in [29] 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 $f_i(\boldsymbol{x}) = \mathbf{H}_{\xi}(M_i\boldsymbol{x} - \boldsymbol{b}_i)$ as described in [29, Sec. 6] such that $L_i = 1$. Moreover, we also initialized all the methods we tested from $\boldsymbol{x}_i(0) = \mathbf{0}$ 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. 3.2, we plot the residual sequence $\{r(k)\}_{k\geq 0}$ for all the methods

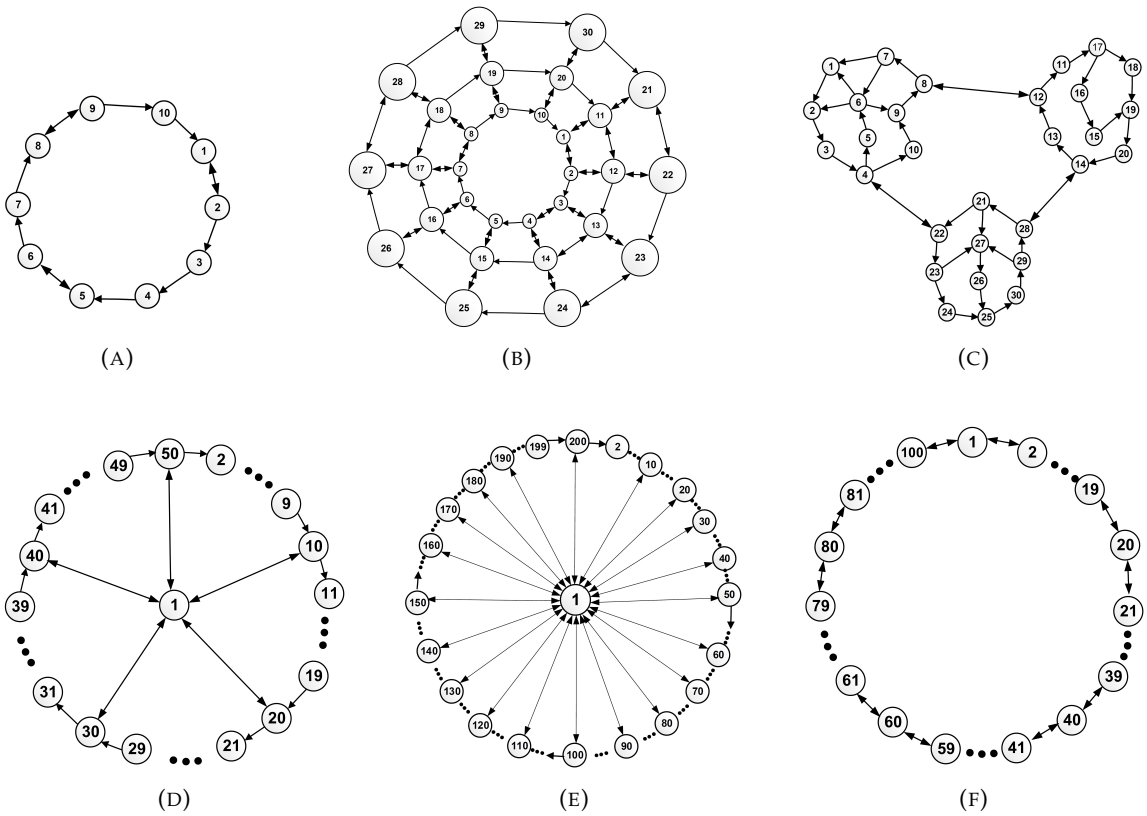
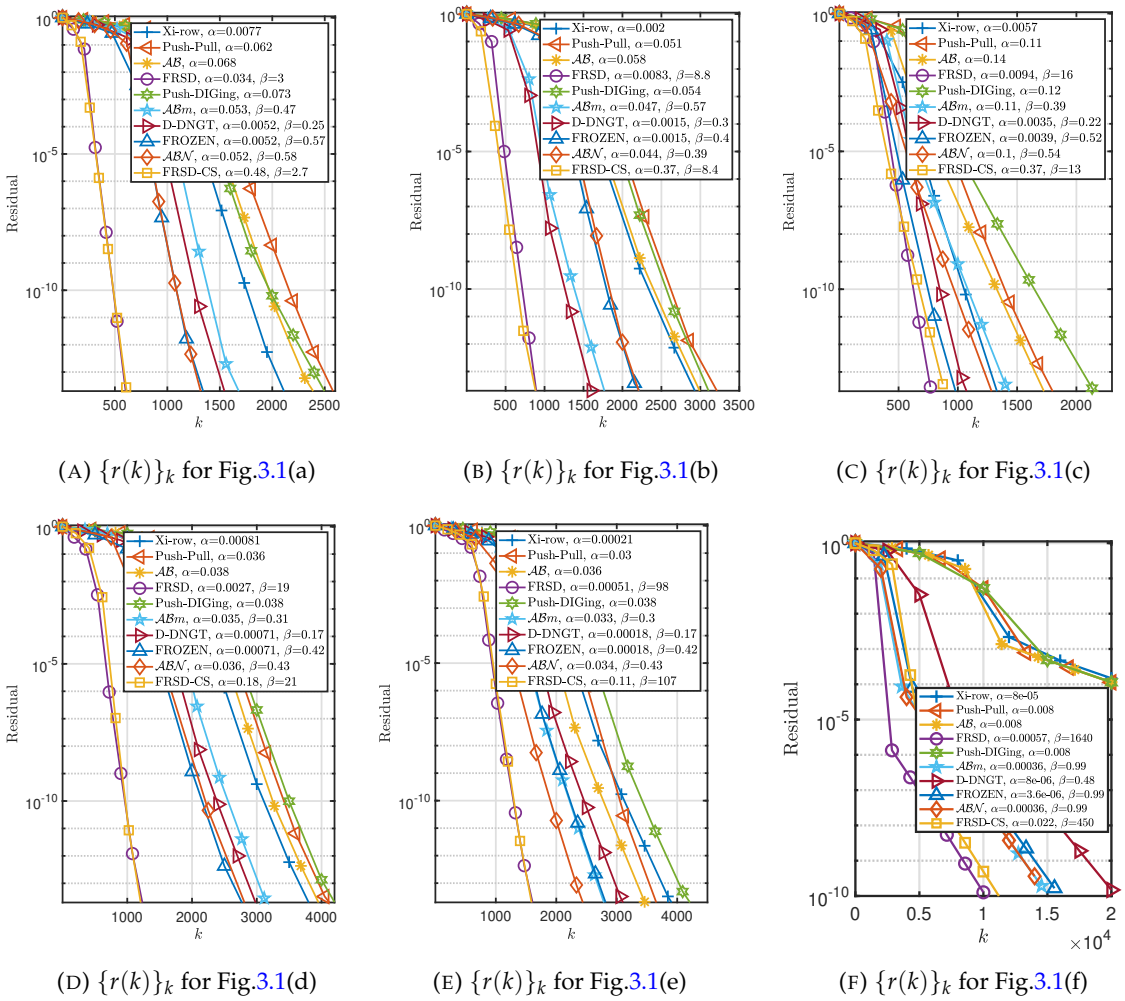


FIGURE 3.1: Strongly-connected digraphs tested in our experiments.

where $r(k) \triangleq \dfrac{\|\mathbf{x}(k) - \mathbf{x}^*\|}{\|\mathbf{x}(0) - \mathbf{x}^*\|}$. To optimize the convergence rate, we tuned parameters for all algorithms. In the appendix A, we present more details about how the step-sizes have been chosen for the methods.



(A) $\{r(k)\}_k$ for Fig.3.1(a) (B) $\{r(k)\}_k$ for Fig.3.1(b) (C) $\{r(k)\}_k$ for Fig.3.1(c)

(D) $\{r(k)\}_k$ for Fig.3.1(d) (E) $\{r(k)\}_k$ for Fig.3.1(e) (F) $\{r(k)\}_k$ for Fig.3.1(f)

FIGURE 3.2: Distributed Regression with Huber Loss

3.4.2 Distributed Logistic Regression

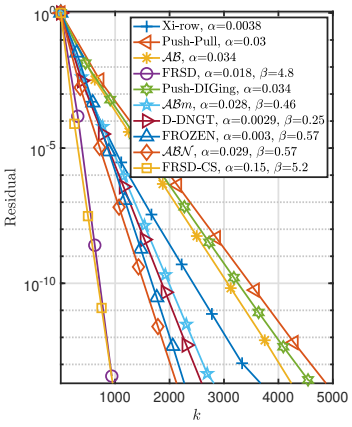
We now consider the distributed binary classification problem using the logistic regression to train a linear classifier. Suppose each node (agent) $i \in \mathcal{V}$ has access to $(M_i, \boldsymbol{b}_i) \in \mathbb{R}^{m_i \times p} \times \{-1,+1\}^{m_i}$. Let $L : \mathbb{R} \times \{-1,1\} \to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

$$\boldsymbol{x}^* = \arg\min_{x \in \mathbb{R}^p} \bar{f}(x) \triangleq \frac{1}{n} \sum_{i \in \mathcal{V}} \left(\mathbf{L}(M_i \boldsymbol{x}, \boldsymbol{b}_i) + \frac{\lambda}{2} \|x\|_2^2 \right). \tag{3.36}$$

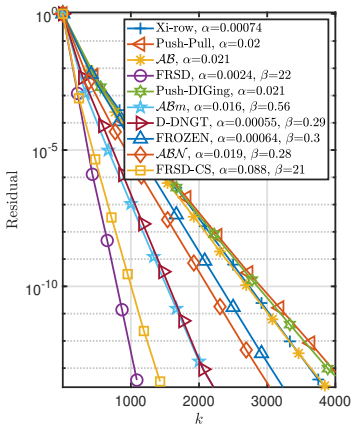
where using regularization parameter $\lambda > 0$ improves the ststistical properties of $\boldsymbol{x}^*$ – see [102].

3.4.2.1 Tests on Specific Graph Topologies

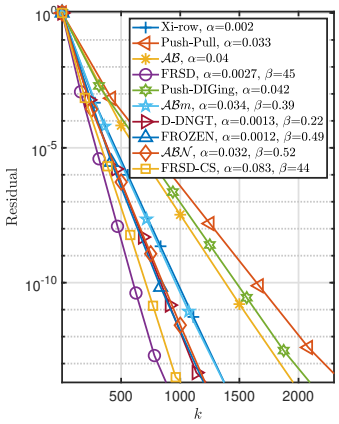
In the first set of experiments, we use the Australian-scale data set [103] with 790 data points where each data point consists of a 14-dimensional feature vector, i.e., $p = 15$ to model the intercept, and the corresponding binary label. Suppose each agent $i \in \mathcal{V}$ samples $m_i = 10$ data 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 3.4.1. The residual sequence $\{r(k)\}_{k\geq 1}$ for all the methods are shown in Fig. 3.3, where $r(k)$ is defined in Section 3.4.1.



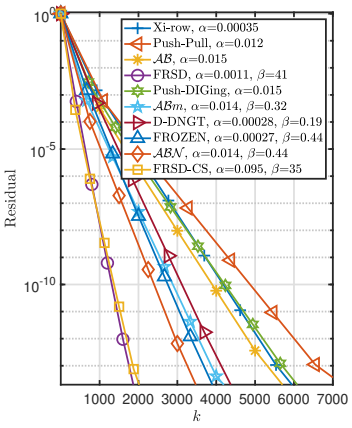
(A) $\{r(k)\}_k$ for Fig.3.1(a)



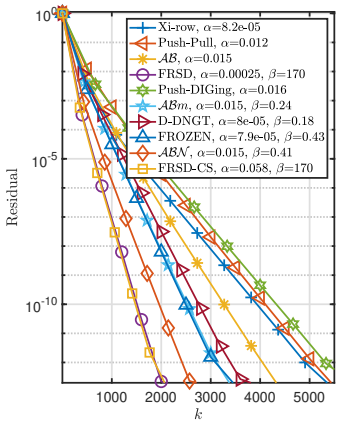
(B) $\{r(k)\}_k$ for Fig.3.1(b)



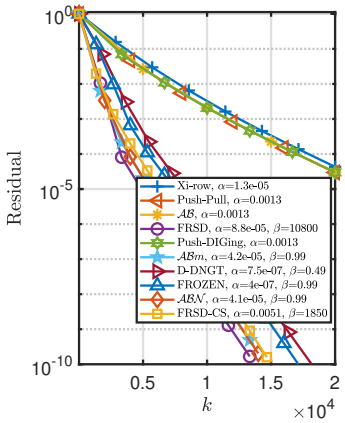
(C) $\{r(k)\}_k$ for Fig.3.1(c)



(D) $\{r(k)\}_k$ for Fig.3.1(d)



(E) $\{r(k)\}_k$ for Fig.3.1(e)



(F) $\{r(k)\}_k$ for Fig.3.1(f)

FIGURE 3.3: Distributed Logistic Regression

3.4.2.2 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 3.4.2. 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 possibles 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 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.

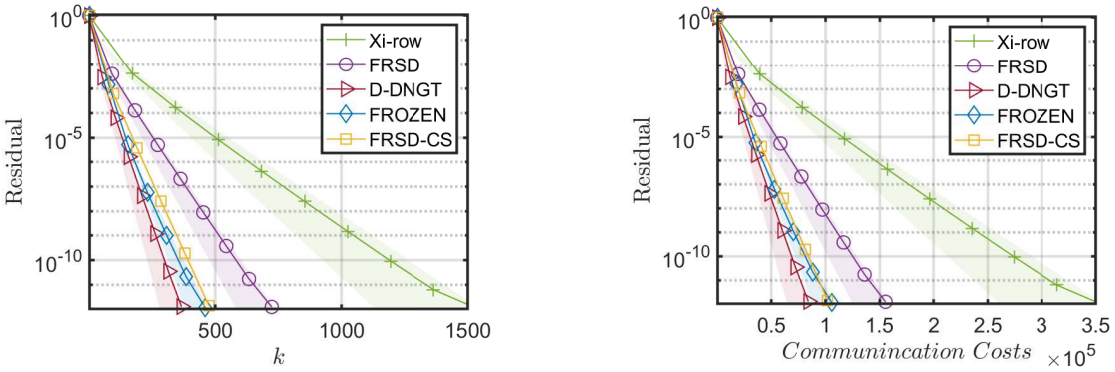
Random Graph Generation We used `DGen` code[1] to generate strongly connected random graphs. The algorithm `DGen`, implemented in MATLAB, receives two inputs: the number of nodes n, and the connectivity ratio $\phi \in (0,1]$. Given n and ϕ, `DGen` generates a strongly connected random graph with $|\mathcal{E}| = \lceil \phi n(n-1) \rceil \triangleq m_{n,\phi}$ 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

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

the remaining $m_{n,\phi} - \tilde{m}$ edges are randomly added to obtain a strongly connected graph with connectivity ratio ϕ.

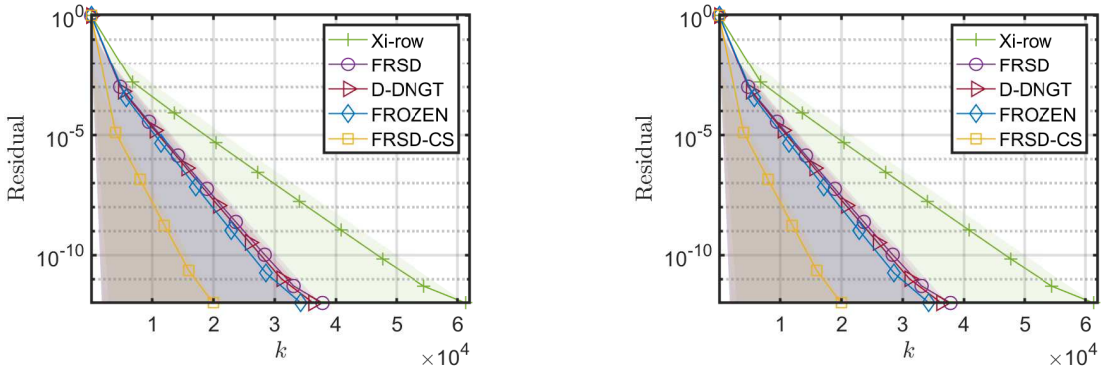
In the experiments with randomly generated strongly connected graphs, we only tested row-stochastic methods. Indeed, being able to get away with the eigenvector estimation through employing both row- and column-stochastic mixing matrices, AB-type methods, e.g., $\mathcal{AB}$ [86], $\mathcal{AB}$m [87], Push-Pull [88] and $\mathcal{ABN}$ [89], 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 3.4.2.1, i.e., $p = 15$ and the number of data points $m_i = 10$ for all $i \in \mathcal{V}$. In Figures 3.4 and 3.5, 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 an increase in either iteration counter or the amount of data broadcast per node.



(A) (B)

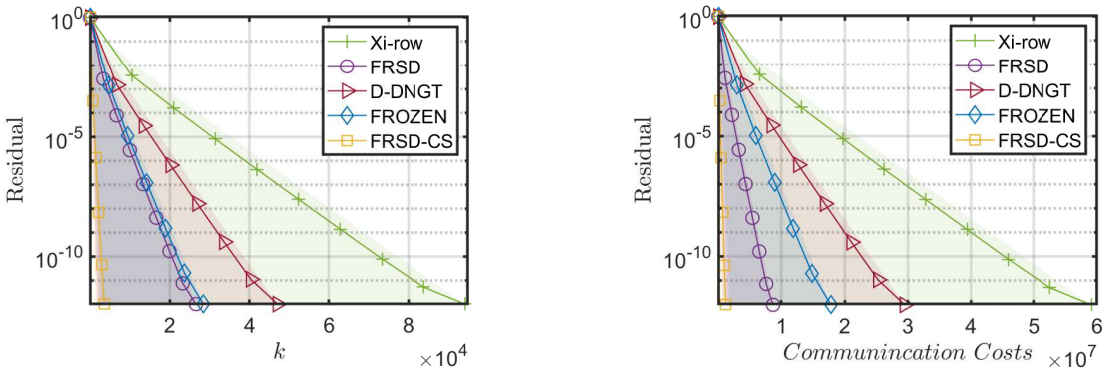
FIGURE 3.4: Distributed logistic regression problem ($n = 200$, $p = 15$) over 20 randomly directed graphs with a high connectivity ratio $\phi = 0.15$. Solid curves represent the average and the shaded region represents the range statistics.



(A) (B)

FIGURE 3.5: Distributed logistic regression problem ($n = 200$, $p = 15$) over 20 randomly 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 [103] 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 3.4.2.1 with $p = 301$ and $m_i = 400$ for all $i \in \mathcal{V}$. In Figure 3.6, 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-stocastic 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.



(A) (B)

FIGURE 3.6: Distributed logistic regression problem ($n = 25$, $p = 301$) over 20 randomly directed graphs with a low connectivity ratio $\phi = 0.1$. Solid curves represent the average and the shaded region represents the range statistics.

Chapter 4

Conclusions and Future Work

This thesis contains two types of contributions in the field of distributed optimization with applications to large-scale networks.

In chapter 2, we proposed a distributed algorithm for microgrid, where the information is only shared between local neighborhoods of the agents. The proposed algorithm, which enjoys the advantages of both the primal and the dual domain methods, provides the optimal solution to the distributed energy management problem of the economic dispatch of grids in order to balance the power demand and supply. All of the marginal costs of the agents converge to the common value while fulfilling the power balance and achieving usage of the available storage in a multistep economic dispatch. The advantage of the proposed approach is the fast convergence rate while no centralized communication is required. The performances of the proposed algorithms have been tested on the 14-bus and the 39-bus system for single and multiple time steps.

In chapter 3, 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.

We consider extending our results to the asynchronous computation setting over directed communication graphs.

Future work. There are some distributed optimization methods, which use a column-stochastic matrix such as Push-DIGing and the method proposed in [104] over time-vary directed networks. However, there is no algorithm, that utilizes only row-stochastic weights and converges linearly over a time-varying network. Note that the algorithms Xi-row and FRSD do not converge linearly. Hence, it would be interesting to obtain a method that only utilizes row-stochastic weights, and would work over time-varying directed networks.

Another direction of research is to investigate second-order distributed optimization methods for directed networks. To the best of our knowledge, presently there are no second-order distributed optimization methods which has a fast convergence compared to the first-order method in general.

Appendix A

In this appendix, we present more details about how the step-sizes have been chosen for all methods and where FRSD may have a better convergence compared to the existing methods. In our experiments, we have also observed that the row-stochastic methods tend to converge faster when the graph is sparse and there are some communication bottlenecks due to graph topology, e.g., in Fig. 3.1a, for node 3 to ping node 1, the message bit should pass through all the nodes. Given a strongly connected di-graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, let d_{ij} be the length of the shortest directed path from $i \in \mathcal{N}$ to $j \in \mathcal{N}$, and we define $\bar{d} \triangleq \sum_{i,j\in\mathcal{N}:\, i\neq j} d_{ij}/(n^2 - n)$, i.e., the average of all shortest directed path lengths over all node pairs. We conjecture that for *sparse* graphs with *high* $\bar{d}$, the performance of row-stochastic methods compared to AB-type methods improves relative to their performances on randomly generated graphs as in Section 3.4.2.2. The random di-graphs, generated as described in Section 3.4.2.2 are similar to Erdos-Renyi graphs, and $\bar{d}$ value turns out to be small for these random graphs compared to those in Fig. 3.1. This is why we think our row-stochastic method, FRSD, performs well for the experiments illustrated in Figure 3.2 and Figure 3.3 as all these numerical tests are conducted on specific graph topologies shown in Figure 3.1. Most importantly, the step sizes for all methods have been chosen very carefully using grid search. For instance, for each method using 2 parameters (α, β), i.e., FRSD, $\mathcal{AB}$m, $\mathcal{ABN}$, D-DNGT and FROZEN, we carefully chose the limits of search range $\bar{\alpha}, \bar{\beta} > 0$ such that the algorithm behaves badly for (α, β) chosen outside the parameter range $[0, \bar{\alpha}] \times [0, \bar{\beta}]$. After we grid the 2-dimensional search space $[0, \bar{\alpha}] \times [0, \bar{\beta}]$, we ran the algorithm for each grid point, i.e., for specific (α, β) values, then we picked the best performing (α, β) within the grid, and reported the results for these parameters. We did this for each problem (Huber Loss and Logistic Regression), for each graph in Fig. 3.1, and for each algorithm individually. Moreover, the chosen algorithm parameter values for all methods in all figures so that the readers can simulate all experiments easily.

To give some intuition on how $\mathcal{AB}m$ behaves for different parameter values, we plot $\mathcal{AB}m$ for 36 different (α, β) choices including the best one we found based on the grid-search (this one is indicated with " $*$ "), and we also plot FRSD as benchmark for all 6 graph topology given in Fig. 3.1. For illustrative purposes, we only provide residual-vs-iteration curves for 36 best performing parameter choices within the grid, which is a tiny fraction of all the grid points tested. Our experiment showed that the behavior of $\mathcal{AB}m$ with respect to parameter choice is stable: as parameters slightly vary, its behavior slightly changes as demonstrated in the plots below. Therefore, FRSD indeed performs better than AB-type methods on specific graph topologies stated in Fig. 3.1.

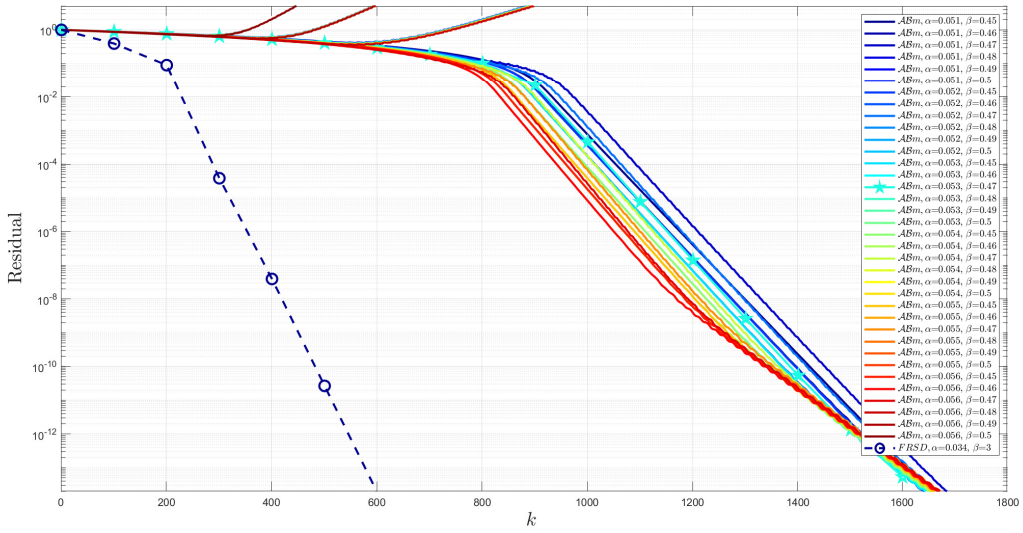


FIGURE A.1: Distributed Regression with Huber Loss for Fig. 3.1a

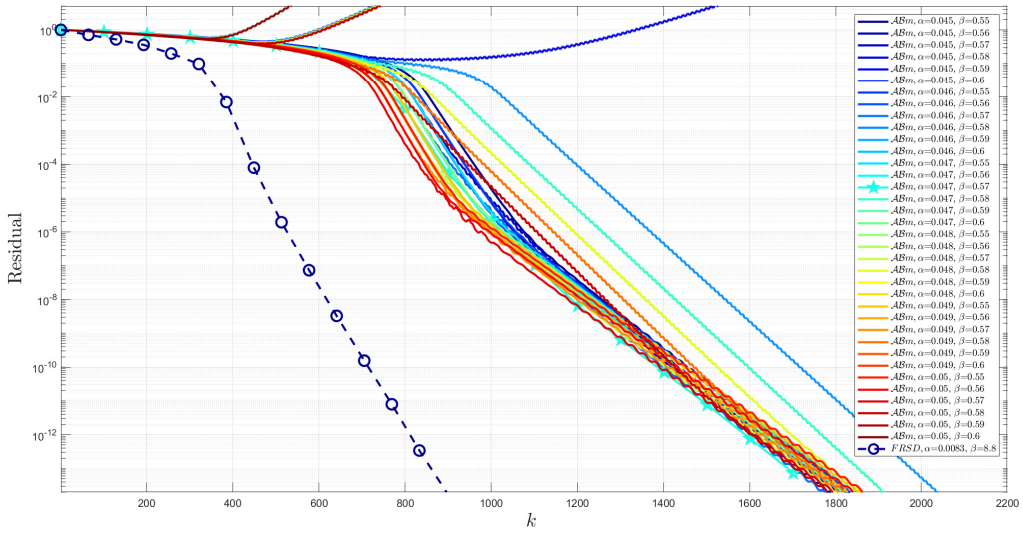


FIGURE A.2: Distributed Regression with Huber Loss for Fig. 3.1b

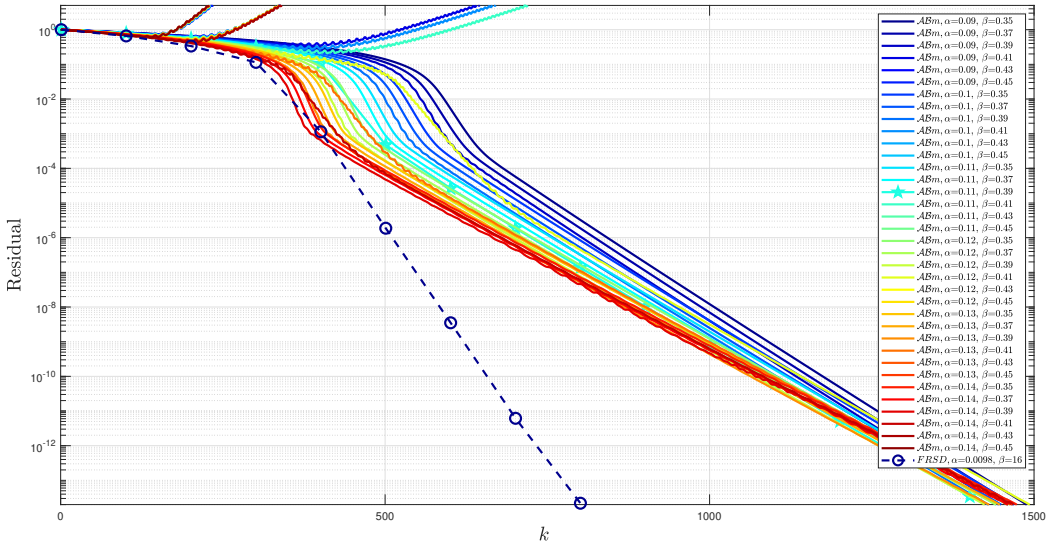


FIGURE A.3: Distributed Regression with Huber Loss for Fig. 3.1c

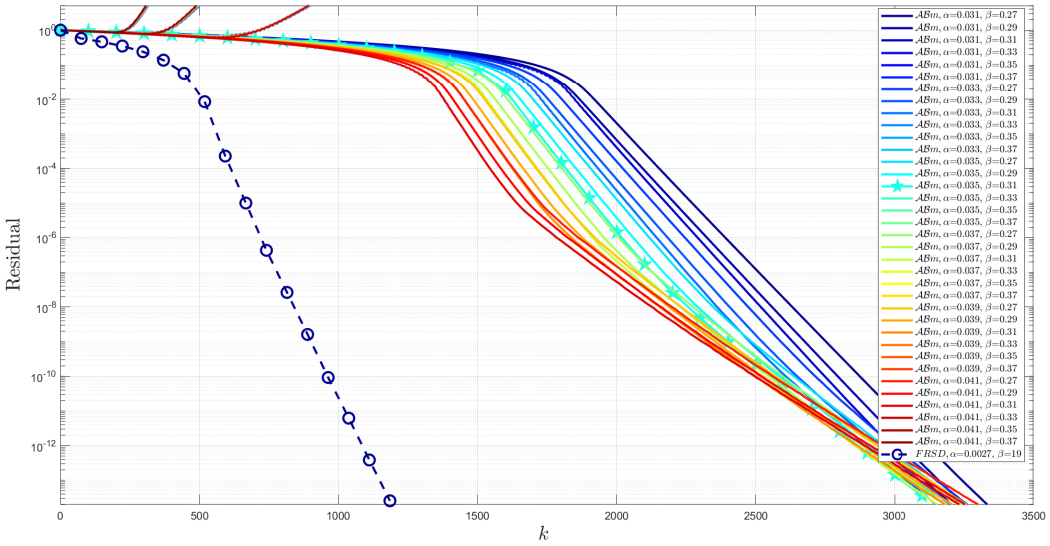


FIGURE A.4: Distributed Regression with Huber Loss for Fig. 3.1d

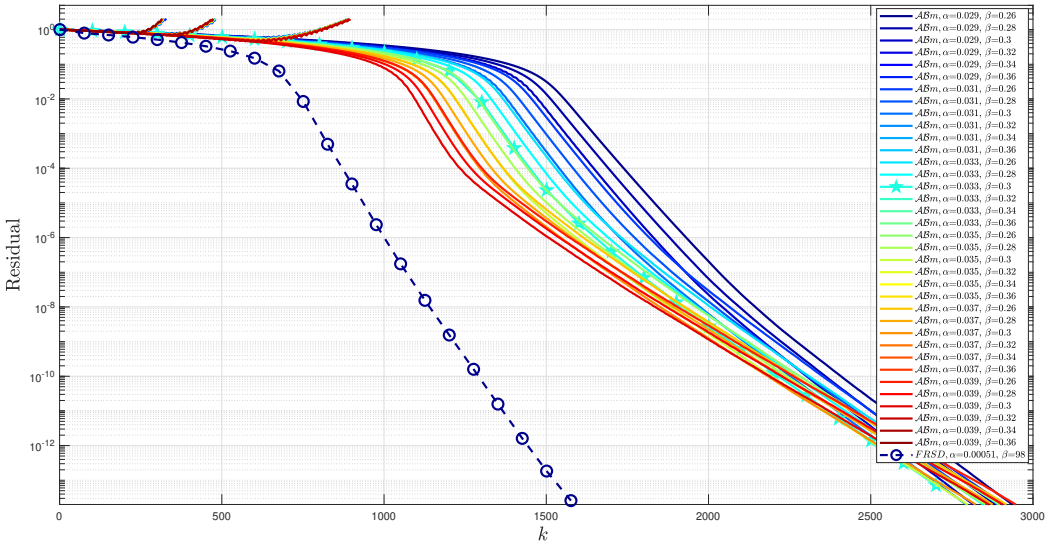


FIGURE A.5: Distributed Regression with Huber Loss for Fig. 3.1e

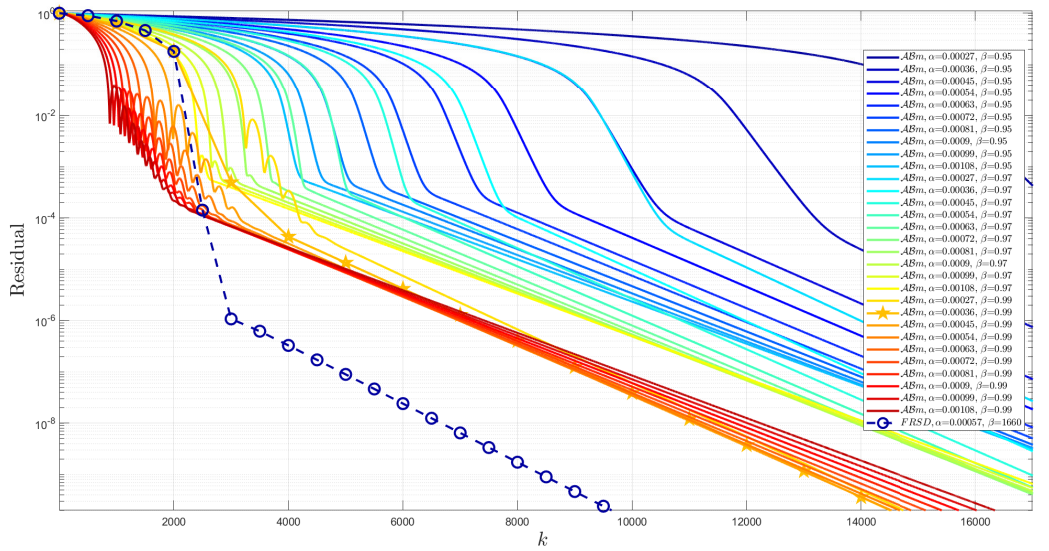


FIGURE A.6: Distributed Regression with Huber Loss for Fig. 3.1f

Bibliography

[1] Y. Nesterov, *Introductory lectures on convex optimization: A basic course*. Springer Science & Business Media, 2003, vol. 87. [Cited on page 3.]

[2] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004. [Cited on pages 3, 26, and 27.]

[3] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *Ussr computational mathematics and mathematical physics*, vol. 4, no. 5, pp. 1–17, 1964. [Cited on page 3.]

[4] A. d'Aspremont, D. Scieur, A. Taylor *et al.*, "Acceleration methods," *Foundations and Trends® in Optimization*, vol. 5, no. 1-2, pp. 1–245, 2021. [Cited on page 3.]

[5] H. Muhsen, A. Alkhraibat, and A. Al-Halhouli, "Real-time simulation and energy management attainment of microgrids," *Sustainability*, vol. 15, no. 3, p. 2696, 2023. [Cited on pages xiii and 4.]

[6] S. Yang, S. Tan, and J.-X. Xu, "Consensus based approach for economic dispatch problem in a smart grid," *IEEE Transactions on Power Systems*, vol. 28, no. 4, pp. 4416–4426, 2013. [Cited on pages 4 and 22.]

[7] H. Pourbabak, J. Luo, T. Chen, and W. Su, "A novel consensus-based distributed algorithm for economic dispatch based on local estimation of power mismatch," *IEEE Transactions on Smart Grid*, vol. 9, no. 6, pp. 5930–5942, 2017. [Cited on page 4.]

[8] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers*. Springer, 2010, pp. 177–186. [Cited on page 5.]

[9] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *Siam Review*, vol. 60, no. 2, pp. 223–311, 2018. [Cited on page 5.]

[10] M. H. DeGroot, "Reaching a consensus," *Journal of the American Statistical association*, vol. 69, no. 345, pp. 118–121, 1974. [Cited on page 10.]

[11] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on automatic control*, vol. 48, no. 6, pp. 988–1001, 2003.

[12] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007. [Cited on page 10.]

[13] 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. [Cited on pages 12 and 42.]

[14] A. H. Sayed and C. G. Lopes, "Adaptive processing over distributed networks," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 90, no. 8, pp. 1504–1510, 2007.

[15] B. Johansson, T. Keviczky, M. Johansson, and K. H. Johansson, "Subgradient methods and consensus algorithms for solving convex optimization problems," in *2008 47th IEEE Conference on Decision and Control*. IEEE, 2008, pp. 4185–4190.

[16] 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. [Cited on pages 12, 42, and 43.]

[17] A. H. Sayed *et al.*, "Adaptation, learning, and optimization over networks," *Foundations and Trends® in Machine Learning*, vol. 7, no. 4-5, pp. 311–801, 2014. [Cited on pages 12, 13, and 49.]

[18] 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. [Cited on pages 13 and 42.]

[19] 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. [Cited on pages 13 and 42.]

[20] 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. [Cited on pages 13, 14, 29, 30, 42, and 43.]

[21] A. Mokhtari and A. Ribeiro, "Dsa: Decentralized double stochastic averaging gradient algorithm," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2165–2199, 2016. [Cited on page 14.]

[22] W. Shi, Q. Ling, G. Wu, and W. Yin, "A proximal gradient algorithm for decentralized composite optimization," *IEEE Transactions on Signal Processing*, vol. 63, no. 22, pp. 6013–6023, 2015. [Cited on page 14.]

[23] 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. [Cited on pages 14 and 42.]

[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. [Cited on pages 14 and 42.]

[25] N. K. Dhingra, S. Z. Khong, and M. R. Jovanović, "The proximal augmented lagrangian method for nonsmooth composite optimization," *IEEE Transactions on Automatic Control*, vol. 64, no. 7, pp. 2861–2868, 2018. [Cited on page 14.]

[26] J. Xu, S. Zhu, Y. C. Soh, and L. Xie, "A bregman splitting scheme for distributed optimization over networks," *IEEE Transactions on Automatic Control*, vol. 63, no. 11, pp. 3809–3824, 2018. [Cited on page 14.]

[27] ——, "Augmented distributed gradient methods for multi-agent optimization under uncoordinated constant stepsizes," in *2015 54th IEEE Conference on Decision and Control (CDC)*. IEEE, 2015, pp. 2055–2060. [Cited on page 15.]

[28] P. Di Lorenzo and G. Scutari, "Distributed nonconvex optimization over networks," in *2015 IEEE 6th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*. IEEE, 2015, pp. 229–232. [Cited on page 15.]

[29] 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. [Cited on pages 15, 17, 43, 44, 48, 66, and 67.]

[30] A. Nedić, A. Olshevsky, W. Shi, and C. A. Uribe, "Geometrically convergent distributed optimization with uncoordinated step-sizes," in *2017 American Control Conference (ACC)*. IEEE, 2017, pp. 3950–3955. [Cited on page 15.]

[31] 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. [Cited on pages 15, 56, and 59.]

[32] ——, "Accelerated distributed nesterov gradient descent," *IEEE Transactions on Automatic Control*, vol. 65, no. 6, pp. 2566–2581, 2019. [Cited on page 15.]

[33] B. Gharesifard and J. Cortés, "Distributed strategies for generating weight-balanced and doubly stochastic digraphs," *European Journal of Control*, vol. 18, no. 6, pp. 539–557, 2012. [Cited on page 15.]

[34] D. Kempe, A. Dobra, J. Gehrke *et al.*, "Computing aggregate information using gossip," in *Proceedings of FOCS'03*, 2003, pp. 482–491. [Cited on page 16.]

[35] A. D. Domínguez-García and C. N. Hadjicostis, "Distributed strategies for average consensus in directed graphs," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*. IEEE, 2011, pp. 2124–2129. [Cited on page 17.]

[36] C. N. Hadjicostis and T. Charalambous, "Average consensus in the presence of delays in directed graph topologies," *IEEE Transactions on Automatic Control*, vol. 59, no. 3, pp. 763–768, 2013.

[37] F. Bénézit, V. Blondel, P. Thiran, J. Tsitsiklis, and M. Vetterli, "Weighted gossip: Distributed averaging using non-doubly stochastic matrices," in *2010 ieee international symposium on information theory*. IEEE, 2010, pp. 1753–1757.

[38] C. N. Hadjicostis, N. H. Vaidya, and A. D. Domínguez-García, "Robust distributed average consensus via exchange of running sums," *IEEE Transactions on Automatic Control*, vol. 61, no. 6, pp. 1492–1507, 2015. [Cited on page 17.]

[39] 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. [Cited on pages 17, 41, and 43.]

[40] 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. [Cited on pages 17, 29, 43, and 57.]

[41] 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. [Cited on pages 18, 43, 44, 49, 56, 58, 64, and 66.]

[42] F. Katiraei, R. Iravani, N. Hatziargyriou, and A. Dimeas, "Microgrids management," *IEEE power and energy magazine*, vol. 6, no. 3, pp. 54–65, 2008. [Cited on page 21.]

[43] R. H. Lasseter, "Microgrids," in *2002 IEEE Power Engineering Society Winter Meeting. Conference Proceedings (Cat. No. 02CH37309)*, vol. 1. IEEE, 2002, pp. 305–308. [Cited on page 21.]

[44] S. Conti, R. Nicolosi, S. Rizzo, and H. Zeineldin, "Optimal dispatching of distributed generators and storage systems for mv islanded microgrids," *IEEE Transactions on Power Delivery*, vol. 27, no. 3, pp. 1243–1251, 2012. [Cited on page 21.]

[45] C. M. Colson and M. H. Nehrir, "Algorithms for distributed decision-making for multi-agent microgrid power management," in *2011 IEEE Power and Energy Society General Meeting*. IEEE, 2011, pp. 1–8. [Cited on page 22.]

[46] Z. Zhang and M.-Y. Chow, "Incremental cost consensus algorithm in a smart grid environment," in *2011 IEEE Power and Energy Society General Meeting*. IEEE, 2011, pp. 1–6. [Cited on page 22.]

[47] ——, "Convergence analysis of the incremental cost consensus algorithm under different communication network topologies in a smart grid," *IEEE Transactions on Power Systems*, vol. 27, no. 4, pp. 1761–1768, 2012. [Cited on page 22.]

[48] G. Hug, S. Kar, and C. Wu, "Consensus+ innovations approach for distributed multiagent coordination in a microgrid," *IEEE Transactions on Smart Grid*, vol. 6, no. 4, pp. 1893–1903, 2015. [Cited on pages 22, 23, 24, 26, 28, 30, 35, and 38.]

[49] S. Kar and G. Hug, "Distributed robust economic dispatch in power systems: A consensus+ innovations approach," in *2012 IEEE Power and Energy Society General Meeting*. IEEE, 2012, pp. 1–8. [Cited on pages 22, 25, and 26.]

[50] T. Yang, D. Wu, Y. Sun, and J. Lian, "Minimum-time consensus-based approach for power system applications," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 2, pp. 1318–1328, 2015. [Cited on page 22.]

[51] T. Yang, J. Lu, D. Wu, J. Wu, G. Shi, Z. Meng, and K. H. Johansson, "A distributed algorithm for economic dispatch over time-varying directed networks with delays," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 6, pp. 5095–5106, 2016. [Cited on pages 22, 24, 26, 29, 30, and 38.]

[52] J. Wu, T. Yang, D. Wu, K. Kalsi, and K. H. Johansson, "Distributed optimal dispatch of distributed energy resources over lossy communication networks," *IEEE Transactions on Smart Grid*, vol. 8, no. 6, pp. 3125–3137, 2017. [Cited on page 22.]

[53] T. Yang, D. Wu, H. Fang, W. Ren, H. Wang, Y. Hong, and K. H. Johansson, "Distributed energy resource coordination over time-varying directed communication networks," *IEEE Transactions on Control of Network Systems*, vol. 6, no. 3, pp. 1124–1134, 2019. [Cited on pages 22, 24, and 26.]

[54] Z. Tang, D. J. Hill, and T. Liu, "A novel consensus-based economic dispatch for microgrids," *IEEE Transactions on Smart Grid*, vol. 9, no. 4, pp. 3920–3922, 2018. [Cited on pages 22, 28, 29, 30, and 32.]

[55] Q. Yang, G. Chen, and T. Wang, "Admm-based distributed algorithm for economic dispatch in power systems with both packet drops and communication delays," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 3, pp. 842–852, 2020. [Cited on page 22.]

[56] S. Mao, Y. Tang, Z. Dong, K. Meng, Z. Y. Dong, and F. Qian, "A privacy preserving distributed optimization algorithm for economic dispatch over time-varying directed networks," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 3, pp. 1689–1701, 2020.

[57] H. Zhong, X. Yan, and Z. Tan, "Real-time distributed economic dispatch adapted to general convex cost functions: a secant approximation-based method," *IEEE Transactions on Smart Grid*, 2021. [Cited on page 22.]

[58] Q. Ling, W. Shi, G. Wu, and A. Ribeiro, "Dlm: Decentralized linearized alternating direction method of multipliers," *IEEE Transactions on Signal Processing*, vol. 63, no. 15, pp. 4051–4064, 2015. [Cited on pages 22, 27, and 28.]

[59] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*. Prentice hall Englewood Cliffs, NJ, 1989, vol. 23. [Cited on page 27.]

[60] 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. [Cited on pages 27 and 42.]

[61] F. Facchinei and J.-S. Pang, *Finite-dimensional variational inequalities and complementarity problems*. Springer Science & Business Media, 2007. [Cited on page 27.]

[62] T.-L. Nguyen, Q.-T. Tran, R. Caire, Y. Wang, Y. Besanger, and N.-A. Luu, "Distributed optimal power flow and the multi-agent system for the realization in cyber-physical system," *Electric Power Systems Research*, vol. 192, p. 107007, 2021. [Cited on page 33.]

[63] G. Chen and Q. Yang, "An admm-based distributed algorithm for economic dispatch in islanded microgrids," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 9, pp. 3892–3903, 2017.

[64] S. Boyd, N. Parikh, and E. Chu, *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011. [Cited on pages 33, 34, and 41.]

[65] D. Wu, T. Yang, A. A. Stoorvogel, and J. Stoustrup, "Distributed optimal coordination for distributed energy resources in power systems," *IEEE transactions on automation science and engineering*, vol. 14, no. 2, pp. 414–424, 2016. [Cited on page 38.]

[66] 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. [Cited on page 41.]

[67] 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. [Cited on page 41.]

[68] 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. [Cited on page 41.]

[69] 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. [Cited on page 41.]

[70] R. Bekkerman, M. Bilenko, and J. Langford, *Scaling up machine learning: Parallel and distributed approaches*. Cambridge University Press, 2011.

[71] 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.

[72] 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.

[73] K. I. Tsianos, S. Lawlor, and M. G. Rabbat, "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. [Cited on pages 41 and 43.]

[74] 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. [Cited on page 41.]

[75] 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. [Cited on page 42.]

[76] 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. [Cited on page 42.]

[77] 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. [Cited on page 42.]

[78] 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.

[79] 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.

[80] ——, "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. [Cited on page 42.]

[81] 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. [Cited on page 42.]

[82] 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. [Cited on page 42.]

[83] 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. [Cited on page 43.]

[84] 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. [Cited on page 43.]

[85] 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. [Cited on page 43.]

[86] 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. [Cited on pages 43, 44, 48, 56, 66, and 71.]

[87] ——, "Distributed heavy-ball: A generalization and acceleration of first-order methods with gradient tracking," *IEEE Transactions on Automatic Control*, 2019. [Cited on pages 44, 48, 49, 66, and 71.]

[88] S. Pu, W. Shi, J. Xu, and A. Nedic, "Push-pull gradient methods for distributed optimization in networks," *IEEE Transactions on Automatic Control*, 2020. [Cited on pages 43, 44, 49, 56, 66, and 71.]

[89] 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. [Cited on pages 43, 44, 49, 66, and 71.]

[90] 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. [Cited on pages 43 and 49.]

[91] 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. [Cited on pages 43, 44, 50, 56, and 66.]

[92] 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. [Cited on page 44.]

[93] 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.

[94] M. S. Assran and M. G. Rabbat, "Asynchronous gradient push," *IEEE Transactions on Automatic Control*, vol. 66, no. 1, pp. 168–183, 2020. [Cited on page 44.]

[95] 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. [Cited on page 53.]

[96] 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*, 2021. [Cited on page 53.]

[97] 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. [Cited on page 53.]

[98] W. Ren and R. W. Beard, "Consensus seeking in multiagent systems under dynamically changing interaction topologies," *IEEE Transactions on automatic control*, vol. 50, no. 5, pp. 655–661, 2005. [Cited on page 55.]

[99] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge university press, 2012. [Cited on pages 55, 57, 63, and 64.]

[100] 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. [Cited on pages 56 and 64.]

[101] B. T. Polyak, "Introduction to optimization. optimization software," *Inc., Publications Division, New York*, vol. 1, 1987. [Cited on page 64.]

[102] 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. [Cited on page 68.]

[103] 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. [Cited on pages 69 and 72.]

[104] F. Saadatniaki, R. Xin, and U. A. Khan, "Decentralized optimization over time-varying directed graphs with row and column-stochastic matrices," *IEEE Transactions on Automatic Control*, vol. 65, no. 11, pp. 4769–4780, 2020. [Cited on page 76.]