Distributed Learning in Non-Convex World: Theory, Algorithms, and Applications

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE
UNIVERSITY OF MINNESOTA
BY

Haoran Sun

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Advisor: Mingyi Hong

May, 2021

Acknowledgements

I would like to first thank my advisor, Prof. Mingyi Hong, for all of his guidance and support during my graduate studies. Thanks to his professionalism, enthusiasm, patient, and endless support, I was able to go through all the difficulties during my studies and make as much progress and reward as I had hoped. I always feel honored and fortunate to have the chance to study and work with him, and none of my work would have been possible without his knowledgeable guidance and support along the way.

I would also like to thank my committee members: Prof. Georgios Giannakis, Prof. Andrew Lamperski, and Prof. Steven Wu. I appreciate that they being my committee members and reviewing this dissertation, and grateful for their encouragement, constructive feedback, and meaningful discussions.

My collaborators also deserve recognition for making me a better researcher. They are Prof. Nikolaos Sidiropoulos from the University of Virginia, Prof. Xiao Fu from Oregon State University, Prof. Qingjiang Shi from Tongji University, Prof. Meisam Razaviyayn from University of Southern California, Prof. Tsung-Hui Chang from the Chinese University of Hong Kong (Shenzhen), Dr. Harish Viswanathan from Nokia Bell Labs, and Dr. Aliye Özge Kaya from Nokia Bell Labs. I am grateful for their insightful suggestions and professional help.

I am also thankful for my current and former colleagues: Dr. Songtao Lu (now at IBM Thomas J. Watson Research Center), Xiangyi Chen, Dr. Ziping Zhao (now at ShanghaiTech University), Xinwei Zhang, Wenqiang Pu, Bingqing Song, Ioannis Tsaknakis, and many other colleagues and collaborators who have worked with me during the last five years. They provided me with unforgettable and meaningful doctoral experiences and enjoyable school life.

Finally, I would like to thank my parents, for their endless love, unconditional support, and warmest encouragement. They are my strength and pleasure to go forward.

Haoran Sun
Minneapolis, MN
May 2021

Dedication

To my beloved parents,

for their endless love, support and encouragement.

Abstract

The world we live in is extremely connected, and it will become even more so in a decade. It is projected that by 2030, there will be 125 billion interconnected smart devices and objects worldwide. These devices are capable of collecting huge amounts of data, performing complex computational tasks, and providing vital services and information to significantly improve our quality of life.

My research develops theories and methods for distributed machine learning and computation, so that future applications can effectively utilize vast of distributed resources such as data, computational power, and storage to become faster, smarter, and more robust. Specifically, the main research objectives and innovative claims include:

1) Theoretical Foundations for Distributed Machine Learning

The first part of this thesis focuses on theoretical guarantees for distributed learning and majorly answering the following question: if the agents can only access the gradients of local functions, what are the *fastest* rates that any distributed algorithms can achieve, and how to achieve those rates?

First, we show that there exist difficult problem instances, such that it takes a class of deterministic distributed first-order methods at least $\mathcal{O}(1/\sqrt{\xi(\mathcal{G})} \times \bar{L}/\epsilon)$ communication rounds to achieve certain ϵ-solution [1]. Then, we propose (near) optimal methods whose rates match the developed lower rate bound (up to a ploylog factor). To the best of our knowledge, this is the first time that lower rate bounds and optimal methods have been developed for distributed non-convex optimization problems. Next, we propose stochastic algorithms to further minimize the number of local data samples needed to be accessed, using the novel variance reduction type of ideas. We show that the proposed algorithm significantly improves upon the existing works and achives the best known sample complexity.

2) Machine Learning Advances for Wireless Communication

In the second part of the thesis, we design a novel machine learning-based strategy

[1] $\xi(\mathcal{G})$ denotes the spectral gap of the graph Laplacian matrix of the underlaying network, and $\bar{L}$ is some Lipschitz constant.

and apply the theory and algorithms developed in Part 1, to improve the real-time performance of modern 5G wireless systems. This task is strongly motivated by the urgent need to design revolutionary network infrastructure and advanced wireless resource allocation strategies, so that future generations of a wireless network can cope with the exponentially growing demand for wireless data.

In specific, we develop a new approach that enables data-driven methods to continuously learn and optimize in a dynamic environment. We propose to build the notion of continual learning (CL) into the modeling process of learning wireless systems, so that the learning model can incrementally adapt to the new environments, *without forgetting* knowledge learned from the previous environments. Our design is based on a novel bilevel optimization formulation which ensures certain "fairness" across different data samples. We demonstrate the effectiveness of the CL approach by integrating it with two popular DNN based models for power control and beamforming, respectively, and testing using both synthetic and ray-tracing based data sets. Numerical results show that the proposed CL approach is not only able to adapt to the new scenarios quickly and seamlessly, but importantly, it also maintains high performance over the previously encountered scenarios as well.

To advocate the reproducible research, the code and implementation detail of this thesis is available online at `https://github.com/Haoran-S/Thesis`.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

We live in a world where virtually everything is connected. It is predicted that there will be over 125 billion smart devices connected via the internet worldwide by 2030 [1]. Many active and emerging applications will heavily rely on our ability to effectively utilize geographically distributed resources such as data, storage, and computational power. For example, consider self-driving cars at an intersection, given the fact that each vehicle generates approximately 40 Gbit/sec of vital data (for positioning, accelerating, braking, steering, and so on) [2], even the fastest cellular network would be overwhelmed, making real-time central processing impossible. These and other examples from small and ordinary (e.g. coordinating multiple smart features in a home) to large and vitally important (e.g. regional or national power distribution) show how important fast distributed optimization, information processing, and peer-to-peer coordination will be to the well-being of, quite literally, billions of people in the future. They reveal that in the near future we will need a large number of distributed devices at the edge of the network that can act as mini cloud servers, capable of processing huge local data and to perform complex computational tasks such as real-time learning and prediction.

Challenges. However, emerging applications arising in the future smart and connected world encounter significant challenges when trying to effectively utilize such

[1] `https://cdn.ihs.com/www/pdf/IoT_ebook.pdf`
[2] `https://www.tuxera.com/blog/autonomous-cars-300-tb-of-data-per-year/`

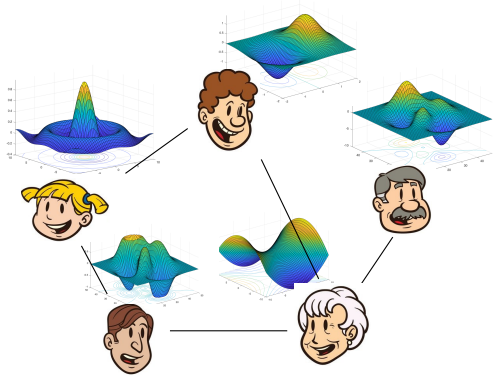


Figure 1.1: Illustration of distributed learning with multiple agents; each agent has part of the data, and they want to utilize the data across all the agents to learn a highly non-linear and non-convex model (such as neural networks).

distributed resources. For example, to build sophisticated intelligence into the devices and networks, distributed processing tasks become increasingly complex. For emerging applications such as distributed dictionary learning, collaborative deep learning, and drone coordination and networking, the interaction pattern among the nodes, and their local cost functions can only be modeled by highly nonlinear and non-convex functions, see illustration in Fig. 1.1. Unfortunately, traditional distributed algorithms based on convex models and linear interactions are no longer applicable. Further, coordinating a large population of distributed nodes requires building efficient and flexible algorithms that guarantee performance, while adaptive to various practical situations such as heterogeneity of the nodes and network failure. How to design new classes of algorithms to meet the stringent requirements on bandwidth, latency and power consumption? What is the best solution accuracy that can be achieved under system resource limitation?

1.2 Research Goals and Contributions

Motivated by the huge potential of decentralized information and computational resources, as well as the related algorithmic and theoretical challenges, my research mainly focuses on developing theoretical foundations and effective algorithms to enable large-scale distributed learning and optimization. Specifically, I am interested in understanding how to best design algorithms and accurately predict solution qualities, so that

distributed agents can jointly accomplish highly complex tasks as efficient as possible. Towards this end, my research has been, and will continue to be, focused on developing fundamental theoretical characterization of decentralized learning, designing efficient algorithms with performance guarantees, as well as finding applications that best illustrate the superiority of large-scale learning and optimization. More specifically, most significant research contributions of this thesis are summarized below.

1.2.1 Theoretical Foundations for Distributed Machine Learning

The first part of this thesis is to investigate various issues in decentralized learning from a theoretical standpoint. Our theoretical analysis then leads to the development of a unifying algorithmic framework, capable of achieving the optimal performance predicted by theory, in a fully distributed manner. Specifically, we pose and address the following open research question:

(Q1) How to identify and provably achieve the best possible performance, measured by rounds and bits of communication, the effort of computation, and quality of the solution, for distributed optimization and machine learning?

Unlike the majority of existing works that develop specific algorithms, our work offers a fresh perspective by investigating the performance limits of any distributed algorithms, and by providing some understanding about the minimum amount of resources required to attain a given solution accuracy. These bounds are significant because they help identify performance gaps in existing methods and provide key insights and standards to guide practical algorithm design. Importantly, such a theoretical investigation leads to an algorithmic framework that settles the open question (Q1).

The major contributions we have made in the first part of the thesis are summarized below.

In Chapter 2, for a class of distributed non-convex optimization problems, we analyze fundamental tradeoffs for key metrics such as computation, communication, and optimality, so as to recognize the best achievable performance for any distributed first-order algorithm. In particular, we show that there exist difficult problem instances, such that it takes a class of distributed first-order methods at least $\mathcal{O}(1/\sqrt{\xi(\mathcal{G})} \times \bar{L}/\epsilon)$ communication rounds to achieve certain ϵ-solution, where $\xi(\mathcal{G})$ denotes the spectral

gap of the graph Laplacian matrix of the underlaying network, and $\bar{L}$ is some Lipschitz constant.

In Chapter 3, we further develop an algorithmic framework called xFILTER, by combining classical signal processing and modern optimization techniques so that optimal performance bounds developed in Chapter 2 can be achieved via fully distributed algorithms. The key in the algorithm design is to properly embed the classical polynomial filtering techniques into modern first-order algorithms.

In Chapter 4, consider the distributed finite sum and stochastic problems, we show that classical methods suffer significant sampling complexity and we propose algorithms that only require the minimum number of local data samples, which significantly improve upon the best existing bounds. In specific, for a distributed system with m agents and each agent has a large number of samples (denoted as n), we show that, to achieve certain ϵ stationary solution of the distributed finite sum problem, the proposed algorithm achieves an $\mathcal{O}(mn^{1/2}\epsilon^{-1})$ sample complexity and an $\mathcal{O}(\epsilon^{-1})$ communication complexity. These bounds significantly improve upon the best existing bounds of $\mathcal{O}(mn\epsilon^{-1})$ and $\mathcal{O}(\epsilon^{-1})$, respectively.

In summary, to the best of our knowledge, this is the first time that 1) the theoretical analysis on the best achievable complexity bounds for a class of distributed learning problems is established [1] and 2) the optimal algorithm that achieves the best possible convergence speed [1] or sample complexity [2] have been developed. We believe our result could offer a comprehensive understanding of the distributed learning problem and serve as a guideline for modern large-scale distributed system design.

Results of this direction led to publications such as *IEEE Transactions on Signal Processing* [1] and *International Conference on Machine Learning (ICML)* [2]. The proposed lower bound and the optimal algorithmic framework [3] has been awarded a Best Student Paper Award at the 52nd Asilomar Conference on Signals, Systems, and Computers.

1.2.2 Machine Learning Advances for Wireless Communication

In the second part of the thesis, we design a novel machine learning-based strategy and apply the theory and algorithms developed in Part I, for allocating wireless resources to

improve the real-time performance of 5G wireless systems. This task is strongly motivated by the urgent need to design revolutionary network infrastructure and advanced wireless resource allocation strategies, so that future generations of a wireless network can cope with the exponentially growing demand for wireless data. In particular, we have developed efficient learning models and optimization algorithms that answer the following generic question:

(Q2)	Can we leverage Deep Neural Networks (DNN) and optimization methods to help significantly improve wireless tasks in modern 5G systems?

The key idea of the proposed approach is to treat the input and output of an existing physical layer optimization algorithm as an unknown nonlinear mapping, use a DNN to approximate it, and use the efficient distributed training algorithms developed in Part I to train such a neural network, as illustrated in [4, Fig. 1]. This design strategy has been applied to solve many complicated wireless resource allocation problems, including but not limited to power allocation [4], channel estimation [5], Time of Arrival (TOA) Estimation [6] and multi-antenna beamforming problems [7].

In Chapter 5, to further deal with the performance loss in the challenging dynamic environment of modern data-driven methods, we introduce the notion of continual learning (CL) to the above-mentioned data-driven framework for wireless system design and develop a CL formulation together with a training algorithm tailored for core tasks in wireless communications. Our framework incrementally adapts the DNN models by using the new incoming data as well as a limited but carefully selected subset of data from the previous experiences and ensures certain "fairness" across different data samples. Numerical results show that the proposed CL approach is not only able to adapt to the new scenarios quickly and seamlessly, but importantly, it maintains high performance over the previously encountered scenarios as well.

Our work in this direction [4] has been selected as the Best Readings in Machine Learning in Communications by the IEEE Communications Society [3], and been identified as being one of the IEEE Signal Processing Society's most downloaded articles from January 2018 to November 2020 for IEEE Transactions on Signal Processing on IEEE Xplore [4].

[3] https://www.comsoc.org/publications/best-readings/machine-learning-communications
[4] https://ieeexplore.ieee.org/document/9363504

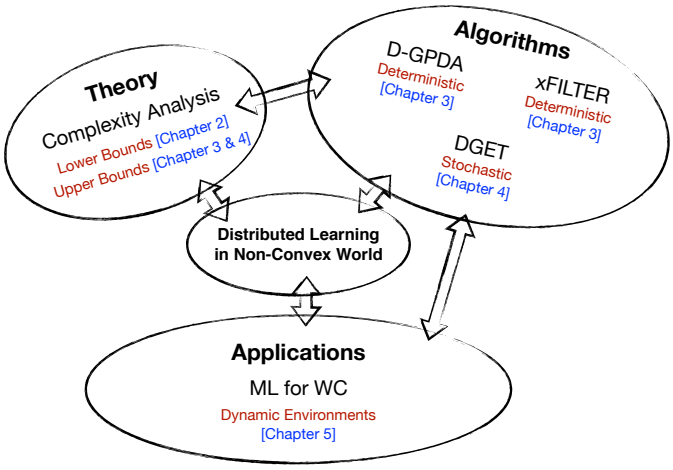


Figure 1.2: Thesis Organization

In Chapter 6, we draw some conclusions for this dissertation and discuss some interesting open problems for future exploration. The overall organization of this thesis is summarized in Fig. 1.2.

1.3 Notations.

Unless otherwise noted, for a given symmetric matrix B, we use $\lambda_{\max}(B)$, $\lambda_{\min}(B)$ and $\underline{\lambda}_{\min}(B)$ to denote the maximum, the minimum and the minimum *nonzero* eigenvalues; We use I_P to denote an identity matrix with size P, use $\otimes$ to denote the Kronecker product, and use $\circ$ to denote the Hadamard product. We use $[M]$ to denote the set $\{1, \cdots, M\}$. For a vector x we use $x[i]$ to denote its ith element. We use $\widetilde{\mathcal{O}}$ to denote $\mathcal{O}(\log(M))$ where M is the problem dimension. We use $i \sim j$ to denote two connected nodes i and j, i.e., for a graph $\mathcal{G} := \{\mathcal{V}, \mathcal{E}\}$, $i \sim j$ if $i \neq j$, and $(i, j) \in \mathcal{E}$.

Chapter 2

Theoretical Limit of Distributed Non-convex Learning

We consider a class of popular distributed non-convex optimization problems, in which agents connected by a network $\mathcal{G}$ collectively optimize a sum of smooth (possibly non-convex) local objective functions. We address the following question: if the agents can only access the gradients of local functions, what are the *fastest* rates that any distributed algorithms can achieve (cf. Chapter 2), and how to achieve those rates (cf. Chapter 3).

In this Chapter, we show that there exist difficult problem instances, such that it takes a class of distributed first-order methods at least $\mathcal{O}(1/\sqrt{\xi(\mathcal{G})} \times \bar{L}/\epsilon)$ communication rounds to achieve certain ϵ-solution [where $\xi(\mathcal{G})$ denotes the spectral gap of the graph Laplacian matrix, and $\bar{L}$ is some Lipschitz constant]. To the best of our knowledge, this is the first time that lower rate bounds have been developed for distributed non-convex optimization problems.

2.1 Introduction

2.1.1 Problem and motivation

In this work, we consider the following distributed optimization problem over a network

$$\min_{y\in\mathbb{R}^S} \ \bar{f}(y) := \frac{1}{M}\sum_{i=1}^{M} f_i(y), \tag{2.1}$$

where $f_i(y) : \mathbb{R}^S \to \mathbb{R}$ is a smooth and possibly non-convex function accessible to agent i. There is no central controller, and the M agents are connected by a network defined by an *undirected* and *unweighted* graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, with $|\mathcal{V}| = M$ vertices and $|\mathcal{E}| = E$ edges. Each agent i can only communicate with its immediate neighbors, and it can access one component function f_i (by "access" we meant that it will be able to query the function and obtain its values and gradients; this notion will be defined precisely shortly).

A common way to reformulate problem (2.1) in the distributed setting is given below. Introduce M local variables $x_1, \cdots, x_M \in \mathbb{R}^S$ and a concatenation of M variables $x := [x_1; \cdots; x_M] \in \mathbb{R}^{SM\times 1}$, and suppose the graph $\{\mathcal{V}, \mathcal{E}\}$ is connected, then the following formulation is equivalent to the global consensus problem

$$\min_{x\in\mathbb{R}^{SM}} \ f(x) := \frac{1}{M}\sum_{i=1}^{M} f_i(x_i), \quad \text{s.t. } x_i = x_j, \forall\ (i,j) \in \mathcal{E}. \tag{2.2}$$

The main benefit of the above formulation is that the objective function is now separable, and the linear constraint encodes the network connectivity pattern.

2.1.2 Lower and upper rate bounds analysis

Distributed non-convex optimization has gained considerable attention recently and many popular algorithms has been developed, see [8–23]. However, despite all the recent interests and contributions in this field, one major question remains open:

(Q) What is the *best* convergence rate achievable by *any* distributed algorithms for the non-convex problem (2.1)?

Question (**Q**) seeks to find a "best convergence rate", which is a characterization of the smallest number of iterations required to achieve certain *high-quality solutions*, among all distributed algorithms. Clearly, understanding (**Q**) provides fundamental insights to distributed optimization and information processing. For example, the answer to (**Q**) offers meaningful optimal estimates on the total amount of communication and computation effort required to achieve a given level of accuracy. Further, the identified optimal strategies capable of attaining the best convergence rates will also help guide the practical design of distributed information processing algorithms.

Question (**Q**) is easy to state, but formulating it rigorously is quite involved and a number of delicate issues have to be clarified. Below we provide a high level discussion on some of these issues.

(1) Fix Problem and Network Classes. A class of problems $\mathcal{P}$ and networks $\mathcal{N}$ of interest should be fixed. Roughly speaking, in this work, we will fix $\mathcal{P}$ to be the family of smooth unconstrained problem (2.1), and $\mathcal{N}$ to be the set of *connected* and *unweighted* graphs with finite number of nodes.

(2) Characterize High-Quality Solutions. For a properly defined error constant $\epsilon > 0$, one needs to define a *high-quality solution* in distributed and non-convex setting. Differently from the centralized case, the following questions have to be addressed: Should the solution quality be evaluated based on the *averaged* iterates among all the agents, or on the individual iterates? Shall we include some *consensus measure* in the solution characterization? Different solution notion could potentially lead to different lower and upper rate bounds.

(3) Fix Algorithm Classes. A class of algorithms $\mathcal{A}$ has to be fixed. In the classical complexity analysis in (centralized) optimization, it is common to define the class of algorithms by the information structures that they utilize [24]. In the distributed and non-convex setting, it is necessary to specify *both* the function information that can be used by individual nodes, as well as the communication protocols that are allowed.

(4) Develop Sharp Upper Bounds. It is necessary to develop algorithms within class $\mathcal{A}$, which possess provable and sharp global convergence rate for problem/network class $(\mathcal{P}, \mathcal{N})$. These algorithms provide achievable *upper bounds* on the global convergence rates.

(5) Identify Lower Bounds. It is important to characterize the *worst* rates achievable

by *any* algorithm in class $\mathcal{A}$ for problem/network class $(\mathcal{P}, \mathcal{N})$. This task involves identifying instances in $(\mathcal{P}, \mathcal{N})$ that are difficult for algorithm class $\mathcal{A}$.

(6) Match Lower and Upper Bounds. The key task is to investigate whether the developed algorithms are *rate optimal*, in the sense that rate upper bounds derived in **(4)** match the *worst-case* lower bounds developed in **(5)**. Roughly speaking, matching two bounds requires that for the class of problem and networks $(\mathcal{P}, \mathcal{N})$, the following quantities should be matched between the lower and upper bounds: *i)* the order of the error constants ϵ; *ii)* the order of problem parameters such as M, or that of network parameters such as the spectral gap, diameter, etc.

Convergence rate analysis (aka iteration complexity analysis) for convex problems dates back to Nesterov, Nemirovsky and Yudin [25, 26], in which lower bounds and optimal *first-order* algorithms have been developed; also see [27]. In recent years, many accelerated *first-order* algorithms achieving those lower bounds for different kinds of convex problems have been derived; see e.g., [28–30], including those developed for distributed convex optimization [31]. In those works, the problem is to optimize $\min_x f(x)$ with convex f, the optimality measure used is $f(x) - f(x^*)$, and the lower bound can be expressed as [27, Theorem 2.2.2]

$$f(x^t) - f(x^*) \leq \frac{\|x^0 - x^*\| L}{(t+2)^2}, \tag{2.3}$$

where L is the Lipschitz constant for ∇f; x^* (resp. x^0) is the global optimal solution (resp. the initial solution); t is the iteration index. Therefore to achieve ϵ-optimal solution in which $f(x^t) - f(x^*) \leq \epsilon$, one needs $\sqrt{\frac{\|x^* - x^0\| L}{\epsilon}}$ iterations. Recently the above approach has been extended to distributed strongly convex optimization in [32]. In particular, the authors consider problem (2.1) in which each f_i is strongly convex, and they provide lower and upper rate bounds for a class of algorithms in which the local agents can utilize both $\nabla f_i(x)$ and its Fenchel conjugate $\nabla^* f_i(x)$. We note that this result is not directly related to the class of "first-order" method, since beyond the first-order gradient information, the Fenchel conjugate $\nabla^* f_i(x)$ is also needed, but computing this quantity requires performing certain exact minimization, which itself involves solving a strongly convex optimization problem. Other related works in this direction also include [33] and [34]. In particular, the work [34] is a non-smooth extension

of [32], where the lower complexity bound under the Lipschitz continuity of the global and local objective function are discussed and the optimal algorithm is proposed.

When the problem becomes non-convex, the size of the gradient function can be used as a measure of solution quality. In particular, let $h_T^* := \min_{0\leq t\leq T} \|\nabla f(x^t)\|^2$, then it has been shown that the classical (centralized) gradient descent (GD) method achieves the following rate [27, page 28]

$$h_T^* \leq \frac{c_0 L(f(x^0) - f(x^*))}{T+1}, \text{ where } c_0 > 0 \text{ is some constant.}$$

It has been shown in [35] that the above rate is (almost) tight for GD. Recently, [36] has further shown that the above rate is optimal for *any* first-order methods that only utilize the gradient information, when applied to problems with Lipschitz gradient. However, no lower bound analysis has been developed for distributed non-convex problem (2.19); there are even not many algorithms that provide achievable upper rate bounds (except for the recent works [10, 13, 37, 38]), not to mention any analysis on the tightness/sharpness of these upper bounds.

2.1.3 Contribution of this work

In this Chapter, we address various issues that arise in answering (**Q**). Our main contributions are given below:

1) We identify a class of non-convex problems and networks $(\mathcal{P}, \mathcal{N})$, a class of distributed first-order algorithms $\mathcal{A}$, and rigorously define the ϵ-optimality gap that measures the progress of the algorithms;

2) We develop the first lower complexity bound for class $\mathcal{A}$ to solve class $(\mathcal{P}, \mathcal{N})$: To achieve ϵ-optimality, it is necessary for any $a \in \mathcal{A}$ to perform $\mathcal{O}(1/\sqrt{\xi(\mathcal{G})} \times \bar{L}/\epsilon)$ rounds of communication among all the nodes, where $\xi(\mathcal{G})$ represents certain spectral gap of the graph Laplacian matrix, and $\bar{L}$ is the averaged Lipschitz constants of the gradients of local functions. On the other hand, it is necessary for any such algorithm to perform $\mathcal{O}(\bar{L}/\epsilon)$ rounds of computation among all the nodes.

2.2 Preliminaries

2.2.1 The class $\mathcal{P}$, $\mathcal{N}$, $\mathcal{A}$

We present the classes of problems, networks and algorithms to be studied, as well as some useful results. We parameterize these classes using a few key parameters so that we can specify their subclasses when needed.

Problem Class. A problem is in class $\mathcal{P}_L^M$ if it satisfies the following conditions.

A1. The objective is an average of M functions; see (2.1).

A2. Each component function $f_i(x)$'s has Lipschitz gradient:

$$\|\nabla f_i(x_i) - \nabla f_i(z_i)\| \leq L_i \|x_i - z_i\|, \ \forall \ x_i, z_i \in \mathbb{R}^S, \ \forall \ i, \tag{2.4}$$

where $L_i \geq 0$ is the *smallest* positive number such that the above inequality holds true. Define $\bar{L} := \frac{1}{M}\sum_{i=1}^M L_i$, $L_{\max} := \max_i L_i$, and $L_{\min}$ similarly.

Define the matrix of Lipschitz constants as:

$$L := \text{diag}([L_1, \cdots, L_M]) \otimes I_S \in \mathbb{R}^{MS \times MS}. \tag{2.5}$$

A3. The function $f(x)$ is lower bounded over $x \in \mathbb{R}^{MS}$, i.e.,

$$\underline{f} := \inf_x f(x) > -\infty. \tag{2.6}$$

These assumptions are rather mild. For example an f_i satisfies [A2-A3] is not required to be second-order differentiable. Below we provide a few non-convex functions that satisfy Assumption [A2-A3], and each of those can be the component function f_i's. Note that the first four functions are of particular interest in learning neural networks, as they are commonly used as activation functions.

(1) The sigmoid function is given by $\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$. We have $\text{sigmoid}(x) \geq 0$, $\text{sigmoid}''(x) \in (-1, 1)$, therefore [A2-A3] are true with $L \leq 1$.

(2) The arctan function satisfies $\arctan(x) \in (-\frac{\pi}{2}, \frac{\pi}{2})$, $\arctan''(x) = \frac{-2x}{(x^2+1)^2} \in [-1, 1]$. So [A2-A3] hold with $L \leq 1$.

(3) The tanh function satisfies $\tanh(x) \geq -1,\ \tanh''(x) \in [-1, 1]$, so [A2-A3] hold with $L \leq 1$.

(4) The logit function is related to the tanh function as follows

$$2\text{logit}(x) = \frac{2e^x}{e^x + 1} = 1 + \tanh(x/2),$$

then Assumptions [A2-A3] are again satisfied.

(5) The $\log(1 + x^2)$ function has applications in structured matrix factorization [39]. Clearly it is lower bounded. Its second-order derivative is also bounded.

(6) Other functions like $\sin(x)$, $\text{sinc}(x)$, $\cos(x)$ are easy to verify. Consider $f(x) := -x_1x_2 + (x_1 - 1)_+^2 + (-x_1 - 1)_+^2$ where $(z)_+^2 := \max\{0, z\}^2$. This function is interesting because it is not second-order differentiable; nonetheless we can verify that [A2-A3] are satisfied with $L = \sqrt{2} + 1$.

Network Class. Let $\mathcal{N}$ denote a class of networks represented by an *undirected* and *unweighted* graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, with $|\mathcal{V}| = M$ vertices and $|\mathcal{E}| = E$ edges, and edge weights all being 1. In this paper the term 'network' and 'graph' will be used interchangeably. Also, we use $\mathcal{N}_D^M$ to denote a class of network similarly as above, but with M nodes and a diameter of D, defined below [where $\text{dist}(\cdot)$ indicates the distance between two nodes]:

$$D := \max_{u,v \in \mathcal{V}} \text{dist}(u, v). \tag{2.7}$$

Following the convention in [40], we define a number of graph related quantities below. First, define the *degree* of node i as d_i, and define the averaged degree as:

$$\bar{d} := \frac{1}{M} \sum_{i=1}^{M} d_i. \tag{2.8}$$

Define the incidence matrix (IM) $A \in \mathbb{R}^{E \times M}$ as follows: if $e \in \mathcal{E}$ and it connects vertex i and j with $i > j$, then $A_{ev} = 1/\sqrt{d_v}$ if $v = i$, $A_{ev} = -1/\sqrt{d_v}$ if $v = j$ and $A_{ev} = 0$ otherwise; see the definition in [40, Theorem 8.3]. Using these definitions, the *graph*

Laplacian matrix and the *degree matrix* are defined as follows (see [40, Section 1.2]):

$$\mathcal{L} := A^T A \in \mathbb{R}^{M\times M}, \quad \text{and} \quad P := \text{diag}[d_1, \cdots, d_M] \in \mathbb{R}^{M\times M}. \tag{2.9}$$

In particular, the elements of the Laplacian are given as:

$$[\mathcal{L}]_{ij} = \begin{cases} 1 & \text{if } i = j \\ -\frac{1}{\sqrt{d_i d_j}} & \text{if } i \sim j, i \neq j \\ 0 & \text{otherwise.} \end{cases}$$

We note that the graph Laplacian defined here is sometimes known as the *normalized* graph Laplacian in the literature, but throughout this paper we follow the convention used in the classical work [40] and simply refer it as the *graph Laplacian*. For convenience, we also define a scaled version of the IM:

$$F := AP^{1/2} \in \mathbb{R}^{E\times M}. \tag{2.10}$$

It is known that IM and scaled IM satisfy the following (where $\mathbb{1} \in \mathbb{R}^M$ is an all one vector):

$$F\mathbb{1} = AP^{1/2}\mathbb{1} = 0. \tag{2.11}$$

Define the second smallest eigenvalue of $\mathcal{L}$, as $\underline{\lambda}_{\min}(\mathcal{L})$:

$$\underline{\lambda}_{\min}(\mathcal{L}) = \inf_{x:\sum_{i=1}^M x_i d_i = 0, x\neq 0} \frac{x^T \mathcal{L} x}{\sum_{i=1}^M x_i^2 d_i}. \tag{2.12}$$

Then the *spectral gap* of the graph $\mathcal{G}$ can be defined below:

$$\xi(\mathcal{G}) = \frac{\underline{\lambda}_{\min}(\mathcal{L})}{\lambda_{\max}(\mathcal{L})} \leq 1. \tag{2.13}$$

Algorithm Class. Define the *neighbor set* for node $i \in \mathcal{E}$ as

$$\mathcal{N}_i := \{i \mid i \sim j, j \neq i\}. \tag{2.14}$$

We say that a *distributed, first-order* algorithm is in class $\mathcal{A}$ if it satisfies the following conditions.

B1. At iteration 0, each node can obtain some network related constants, such as M, D, eigenvalues of the graph Laplacian $\mathcal{L}$, etc.

B2. At iteration $t+1$, *each node* $i \in [M]$ first conducts a communication step by broadcasting the local x_i^t to all its neighbors, through a function $Q_i^t(\cdot) : \mathbb{R}^S \to \mathbb{R}^S$. Then each node will generate the new iterate, by combining the received message with its past gradients using a function $W_i^t(\cdot)$:

$$v_i^t = \underbrace{Q_i^t(x_i^t)}_{\text{communication step}} \;,\; x_i^{t+1} \in \underbrace{W_i^t\left(\{\{v_j^k\}_{j\in\mathcal{N}_i}, \nabla f_i(x_i^k), x_i^k\}_{k=1}^t\right)}_{\text{computation step}}. \tag{2.15}$$

In this work, we will focus on the case where the $Q_i^t(\cdot)$'s and $W_i^t(\cdot)$'s are linear operators.

Clearly $\mathcal{A}$ belongs to the class of *first-order* methods because only local gradient information is used. It is also a class of *distributed* algorithms because at each iteration the nodes only communicate with their immediate neighbors.

Additionally, in practical distributed algorithms such as DSG, ADMM or EXTRA, nodes are dictated to use a *fixed strategy* to linearly combine all its neighbors' information. To model such a requirement, below we consider a slightly restricted algorithm class $\mathcal{A}'$, where we require each node to use the same coefficients to combine its neighbors (note that allowing the nodes to use a fixed but *arbitrary* linear combination is also possible, but the resulting analysis will be more involved).

In particular, we say that a *distributed, first-order* algorithm is in $\mathcal{A}'$ if it satisfies B1 and the following:

B2'. At iteration $t+1$, *each node* $i \in [M]$ performs:

$$v_i^t = Q_i^t(x_i^t),\; x_i^{t+1} \in W_i^t\left(\{\sum_{j\in\mathcal{N}_i} v_j^t, \nabla f_i(x_i^k), x_i^k\}_{k=1}^t\right). \tag{2.16}$$

We remark that, in both algorithm classes, *one round* of communication occurs at

each iteration, where *each* node broadcasts its local variable x_i^t once. Therefore, the total iteration number is the same as the total communication rounds. However, the total times that the entire gradient $\{\nabla f_i(x_i)\}_{i=1}^M$ is evaluated could be *smaller* than the total iteration number/communication rounds. This is because when we compute x_i^{t+1}, the operation $W_i^t(\cdot)$ can set the coefficient in front of $\nabla f_i(x_i^r)$ to be zero, effectively *skipping* the local gradient computation.

2.2.2 Solution Quality Measure

Next we provide definitions for the quality of the solution. Note that since we consider using first-order methods to solve non-convex problems, it is expected that in the end some first-order stationary solution with small $\|\nabla f\|$ will be computed.

Our first definition is related to a *global variable* $y^t \in \mathbb{R}^S$. We say that y^t is a *global* ϵ-solution if the following holds:

$$y^t \in \text{span}\left\{x_i^t\right\}_{i=1}^M, \ \min_{t\in[T]} \|\nabla g(y^t)\|^2 \leq \epsilon. \tag{2.17}$$

This definition is conceptually simple and it is identical to the centralized criteria in Section 2.1.2. However it has the following issues. First, no global variable y^t will be formed in the entire network, so criteria (2.17) is difficult to evaluate. Second, there is no characterization of how close the local variables x_i^t's are. To see the second point, consider the following toy example.

Example 1: Consider a network with $M = 2$ and $f_1(y) = -y^2$ and $f_2(y) = y^2$. Suppose that the local variables take the following values: $x_1^T = -10$ and $x_2^T = 10$. Then if we pick $y^T = (x_1^T + x_2^T)/2 = 0$, we have

$$\nabla g(y^T) = \frac{1}{2}(\nabla f_1(y^T) + \nabla f_2(y^T)) = 0.$$

This suggests that at iteration T, there exists one linear combination that makes measure (2.17) precisely zero. However one can hardly say that the current solution $(x_1^T, x_2^T) = (-10, 10)$ is a good solution for problem (2.2). □

To address the above issue, we provide a second definition which is directly related to *local variables* $\{x_i \in \mathbb{R}^S\}_{i=1}^M$. At a given iteration T, we say that $\{x_i^T\}$ is a *local*

ϵ-solution if the following holds:

$$h_T^* := \min_{t\in[T]} \left\| \sum_{i=1}^{M} \frac{\nabla f_i(x_i^t)}{M} \right\|^2 + \frac{1}{M\underline{\lambda}_{\min}(P^{1/2}\mathcal{L}P^{1/2})} \sum_{(i,j):i\sim j} \sqrt{L_i L_j}\|x_i^t - x_j^t\|^2 \leq \epsilon. \quad (2.18)$$

Clearly this definition takes into consideration the consensus error as well as the size of the local gradients. When applied to Example 1, this measure will be large. Note that the constant $\frac{1}{M\underline{\lambda}_{\min}(P^{1/2}\mathcal{L}P^{1/2})}$ is needed to balance the two different measures. Also note that the "$\min_{t\in[T]}$" operation is needed to track the best solution obtained before iteration T, because the quantity inside this operation may not be monotonically decreasing.

In our work we will focus on providing answers to the following specific version of question (**Q**):

For *any* given $\epsilon > 0$, what is the minimum iteration T (as a function of ϵ) needed for any algorithm in class $\mathcal{A}$ (or class $\mathcal{A}$') to solve instances in classes $(\mathcal{P}, \mathcal{N})$, so to achieve $h_T^* \leq \epsilon$?

2.2.3 Some Useful Facts and Definitions

Below we provide a few facts about the above classes.

On Lipschitz constants. Assume that each f_i has Lipschitz continuous gradient with constant L_i in (2.4). Then we have :

$$\|\nabla \bar{f}(y_1) - \nabla \bar{f}(y_2)\| \leq \sum_{i=1}^{M} \frac{1}{M} L_i \|y_1 - y_2\| := \bar{L}\|y_1 - y_2\|, \ \forall\ y_1,\ y_2 \in \mathbb{R}^S, \quad (2.19)$$

where $\bar{L}$ is the average of the local Lipschitz gradients. We also have the following

$$\|\nabla f(x) - \nabla f(z)\|^2 = \frac{1}{M^2} \sum_{i=1}^{M} \|\nabla f_i(x_i) - \nabla f_i(z_i)\|^2, \ \forall\ x_i,\ z_i \in \mathbb{R}^S$$

which implies

$$\|\nabla f(x) - \nabla f(z)\| \leq \frac{1}{M}\|L(x - z)\|, \quad \forall\ x, z \in \mathbb{R}^{MS}, \quad (2.20)$$

where the matrix L is defined in (2.5).

On Quantities for Graph $\mathcal{G}$. This section presents a number of properties for a given graph $\mathcal{G}$. Define the following matrices:

$$\Sigma := \text{diag}[\sigma_1, \cdots, \sigma_E] \succ 0, \ \Upsilon := \text{diag}([\beta_1, \cdots, \beta_M]) \succ 0. \tag{2.21}$$

Define $B \in \mathbb{R}^{E\times M} = |F|$ where the absolute value is taken component-wise. Then we have the following:

$$\begin{aligned} &\frac{1}{2}\left(F^T F + B^T B\right) = P = \text{diag}[d_1, \cdots, d_M] \in \mathbb{R}^{M\times M} \\ &\frac{1}{2}\left(F^T \Sigma^2 F + B^T \Sigma^2 B\right) = \text{diag}\left(\left\{\sum_{j:i\sim j} \sigma_{ij}^2\right\}_{j\in\mathcal{N}}\right) := \Delta, \end{aligned} \tag{2.22}$$

where P is the *degree matrix* defined in (2.9).

For two diagonal matrices Υ^2 and Σ^2 of appropriate sizes, the *generalized Laplacian* (GL) matrix is defined as:

$$\mathcal{L}_G = \Upsilon^{-1} F^T \Sigma^2 F \Upsilon^{-1}, \tag{2.23}$$

and its elements are given by:

$$[\mathcal{L}_G]_{ij} = \begin{cases} \frac{\sum_{q:i\sim q}\sigma_{iq}^2}{\beta_i^2} & \text{if } i = j \\ -\frac{\sigma_{ij}^2}{\beta_i \times \beta_j} & \text{if } (ij) \in \mathcal{E}, i \neq j \\ 0 & \text{otherwise} \end{cases}.$$

Define a diagonal matrix $K \in \mathbb{R}^{E\times E}$ as below:

$$[K]_{e,q} = \begin{cases} \sqrt{L_i L_j} & \text{if } e = q, \text{ and } e = (i,j) \\ 0 & \text{otherwise} \end{cases}. \tag{2.24}$$

Then when specializing $\Upsilon = P^{1/2} L^{1/2}$ and $\Sigma^2 = K$, the GL matrix becomes:

$$\widetilde{\mathcal{L}} := L^{-1/2} P^{-1/2} F^T K F P^{-1/2} L^{-1/2}. \tag{2.25}$$

Note that if any diagonal element in the matrix L is zero, then L^{-1} denotes the Moore - Penrose matrix pseudoinverse. Similarly, when specializing $\Upsilon = L^{1/2}$ and $\Sigma^2 = K$, then the GL matrix becomes:

$$\widehat{\mathcal{L}} := L^{-1/2} F^T K F L^{-1/2}. \tag{2.26}$$

These matrices will be used later in our derivations.

Below we list some useful results about the Laplacian matrix [40–42]. First, all eigenvalues of $\mathcal{L}$ lie in the interval $[0,\ 2]$. Also because $\underline{\lambda}_{\min}(\mathcal{L}) = \underline{\lambda}_{\min}(P^{-1/2} F^T F P^{-1/2})$, we have

$$\underline{\lambda}_{\min}(\mathcal{L}) \leq \underline{\lambda}_{\min}(F^T F). \tag{2.27}$$

Also we have that [40, Lemma 1.9]

$$\underline{\lambda}_{\min}(\mathcal{L}) \geq \frac{1}{D \sum_i d_i}. \tag{2.28}$$

The eigenvalues of $\mathcal{L}$ for a number of special graphs are given below:

1) Complete Graph: The eigenvalues are 0 and $M/(M-1)$ (with multiplicity $M-1$), so $\xi(\mathcal{G}) = 1$;

2) Star Graph: The eigenvalues are 0 and 1 (with multiplicity $M-2$), and 2, so $\xi(\mathcal{G}) = 1/2$;

3) Path Graph: The eigenvalues are $1 - \cos(\pi m/(M-1))$ for $m = 0,\ 1, \cdots, M-1$, and $\xi(\mathcal{G}) \geq 1/M^2$.

4) Cycle Graph: The eigenvalues are $1 - \cos(2\pi m/M)$ for $m = 0,\ 1, \cdots, M-1$, and $\xi(\mathcal{G}) \geq 1/M^2$.

5) Grid Graph: The grid graph is obtained by placing the nodes on a $\sqrt{M} \times \sqrt{M}$ grid, and connecting nodes to their nearest neighbors. We have $\xi(\mathcal{G}) \geq 1/M$.

6) Random Geometric Graph: Place the nodes uniformly in $[0,1]^2$ and connect any two nodes separated by a distance less than a radius $R \in (0,1)$. Then if the connectivity radius R satisfies [42]

$$R = \Omega\left(\sqrt{\log^{1+\epsilon}(M)/M}\right), \quad \text{for any } \epsilon > 0, \tag{2.29}$$

then with high probability

$$\xi(\mathcal{G}) = \mathcal{O}\left(\frac{\log(M)}{M}\right). \tag{2.30}$$

2.3 Lower Complexity Bounds

In this section we develop the lower complexity bounds for algorithms in class $\mathcal{A}$ to solve problems $\mathcal{P}_L^M$ over network $\mathcal{N}$. We will mainly focus on the case where f_i's have uniform Lipschitz constants, that is, we assume that

$$L_i = U, \quad \forall\ i \in [M],$$

and we denote the resulting problem class as $\mathcal{P}_U^M$. At the end of this section, generalization to the non-uniform case will be briefly discussed.

Our proof combines ideas from the classical proof in Nesterov [24], as well as two recent constructions [36] (for centralized non-convex problems) and [32] (for strongly convex distributed problems). Our construction differs from the previous works in a number of ways, in particular, the constructed functions are only first-order differentiable, but not second-order differentiable. Further, we use the local-ϵ solution (2.18) to measure the quality of the solution, which makes the analysis more involved compared with the existing *global* error measures in [24, 32, 36].

To begin with, we construct the following two non-convex functions

$$h(x) := \frac{1}{M}\sum_{i=1}^{M} h_i(x_i), \quad f(x) := \frac{1}{M}\sum_{i=1}^{M} f_i(x_i), \tag{2.31}$$

as well as the corresponding versions that evaluate on a "centralized" variable y

$$\bar{h}(y) := \frac{1}{M}\sum_{i=1}^{M} h_i(y), \quad \bar{f}(y) := \frac{1}{M}\sum_{i=1}^{M} f_i(y). \tag{2.32}$$

Here we have $x_i \in \mathbb{R}^T$, for all i, $y \in \mathbb{R}^T$, and $x := (x_1, \cdots x_M) \in \mathbb{R}^{TM\times 1}$. Later we make our construction so that functions h and $\bar{h}$ are easy to analyze, while f and $\bar{f}$ will be in the desired function class in $\mathcal{P}_U^M$. Without loss of generality, in the construction we

Figure 2.1: The path graph used in our construction.

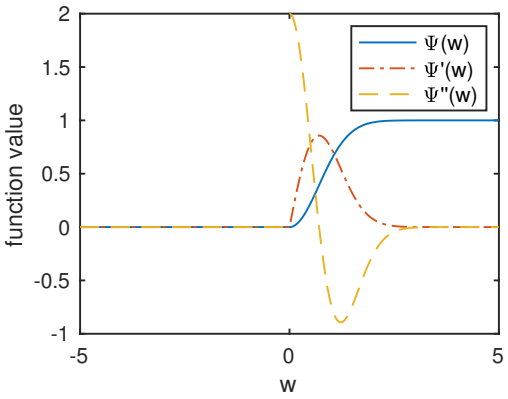


Figure 2.2: The functional value, and derivatives of Ψ.

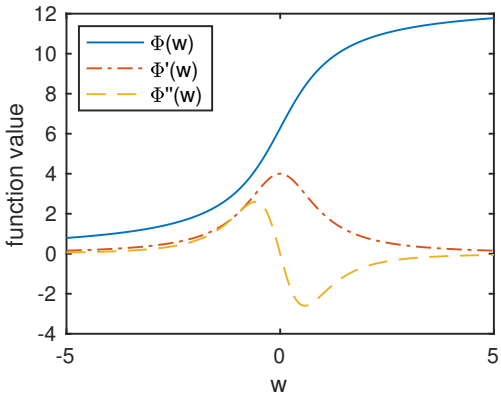


Figure 2.3: The functional value, and derivatives of Φ.

will assume ∇f_i will be Lipschitz with constant $U \in (0, 1)$, for all $i \in [M]$.

2.3.1 Path Graph ($D = M - 1$)

First we consider the extreme case in which the nodes form a path graph with M nodes and each node i has its own local function h_i, shown in Figure 2.1. For notational simplicity assume that M is a multiple of 3, that is $M = 3C$ for some integer $C > 0$. Also assume that T is an odd number without loss of generality.

Let us define the component functions h_i's in (2.31) as follows.

$$h_i(x_i) = \begin{cases} \Theta(x_i, 1) + 3 \sum_{j=1}^{\lfloor T/2 \rfloor} \Theta(x_i, 2j), & i \in \left[1, \frac{M}{3}\right] \\ \Theta(x_i, 1), & i \in \left[\frac{M}{3} + 1, \frac{2M}{3}\right] \\ \Theta(x_i, 1) + 3 \sum_{j=1}^{\lfloor T/2 \rfloor} \Theta(x_i, 2j+1), & i \in \left[\frac{2M}{3} + 1, M\right] \end{cases} \tag{2.33}$$

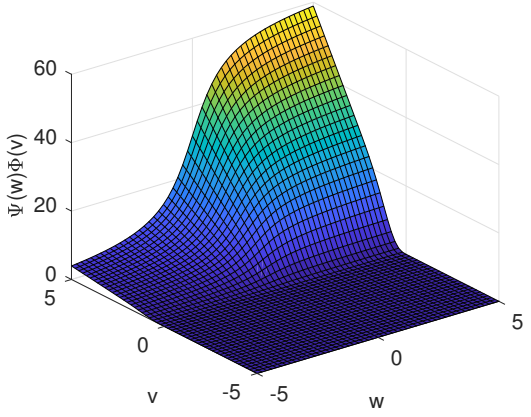


Figure 2.4: The functional value for $\Theta(w, v) = \Psi(w)\Phi(v)$.

where we have defined the following functions

$$\Theta(x_i, j) := \Psi(-x_i[j-1])\Phi(-x_i[j]) - \Psi(x_i[j-1])\Phi(x_i[j]),\ \forall\ j \geq 2 \tag{2.34a}$$

$$\Theta(x_i, 1) := -\Psi(1)\Phi(x_i[1]). \tag{2.34b}$$

The component functions $\Psi, \Phi : \mathbb{R} \to \mathbb{R}$ are given as below

$$\Psi(w) := \begin{cases} 0 & w \leq 0 \\ 1 - e^{-w^2} & w > 0, \end{cases} \quad \text{and} \quad \Phi(w) := 4 \arctan w + 2\pi.$$

Suppose $x_1 = x_2 = \cdots = x_M = y$, then the average function becomes:

$$\begin{aligned} \bar{h}(y) &:= \frac{1}{M} \sum_{j=1}^{M} h_i(y) = \Theta(y, 1) + \sum_{i=2}^{T} \Theta(y, i) \\ &= -\Psi(1)\Phi\left(y[1]\right) + \sum_{i=2}^{T} \left[\Psi\left(-y[i-1]\right)\Phi\left(-y[i]\right) - \Psi\left(y[i-1]\right)\Phi\left(y[i]\right)\right]. \end{aligned}$$

Further for a given error constant $\epsilon > 0$ and a given averaged Lipschitz constant $U \in (0, 1)$, let us define

$$f_i(x_i) := \frac{150\pi\epsilon}{U} h_i\left(\frac{x_i U}{75\pi\sqrt{2\epsilon}}\right). \tag{2.35}$$

Therefore we also have, if $x_1 = x_2 = \cdots = x_M = y$, then

$$\bar{f}(y) := \frac{1}{M}\sum_{i=1}^{M} f_i(y) = \frac{150\pi\epsilon}{U}\bar{h}\left(\frac{yU}{75\pi\sqrt{2\epsilon}}\right). \tag{2.36}$$

First we present some properties of the component functions h_i's.

Lemma 2.3.1. *The functions Ψ and Φ satisfy the following.*

1. *For all $w \le 0$, $\Psi(w) = 0$, $\Psi'(w) = 0$.*

2. *The following bounds hold for the functions and their first and second-order derivatives:*

$$0 \le \Psi(w) < 1, \quad 0 \le \Psi'(w) \le \sqrt{\frac{2}{e}}, \quad -\frac{4}{e^{\frac{3}{2}}} \le \Psi''(w) \le 2, \quad \forall w > 0$$

$$0 < \Phi(w) < 4\pi, \quad 0 < \Phi'(w) \le 4, \quad -\frac{3\sqrt{3}}{2} \le \Phi''(w) \le \frac{3\sqrt{3}}{2}, \quad \forall w \in \mathbb{R}$$

3. *The following key property holds:*

$$\Psi(w)\Phi'(v) > 1, \quad \forall\ w \ge 1,\ |v| < 1. \tag{2.37}$$

4. *The function h is lower bounded as follows:*

$$h_i(0) - \inf_{x_i} h_i(x_i) \le 10\pi T,\ h(0) - \inf_{x} h(x) \le 10\pi T.$$

5. *The first-order derivative of $\bar{h}$ (resp. h_j) is Lipschitz continuous with constant $\ell = 75\pi$ (resp. $\ell_j = 75\pi$, $\forall\ i$).*

Proof. Property 1) is obviously true.

To prove Property 2), note that following holds for $w > 0$:

$$\Psi(w) = 1 - e^{-w^2}, \quad \Psi'(w) = 2e^{-w^2}w, \quad \Psi''(w) = 2e^{-w^2} - 4e^{-w^2}w^2,\ \forall\ w > 0. \tag{2.38}$$

Obviously, $\Psi(w)$ is an increasing function over $w > 0$, therefore the lower and upper bounds are $\Psi(0) = 0, \Psi(\infty) = 1$; $\Psi'(w)$ is increasing on $[0, \frac{1}{\sqrt{2}}]$ and decreasing on

$[\frac{1}{\sqrt{2}}, \infty]$, where $\Psi''(\frac{1}{\sqrt{2}}) = 0$, therefore the lower and upper bounds are $\Psi'(0) = \Psi'(\infty) = 0, \Psi'(\frac{1}{\sqrt{2}}) = \sqrt{\frac{2}{e}}$; $\Psi''(w)$ is decreasing on $(0, \sqrt{\frac{3}{2}}]$ and increasing on $[\sqrt{\frac{3}{2}}, \infty)$ [this can be verified by checking the signs of $\Psi'''(w) = 4e^{-w^2}w(2w^2-3)$ in these intervals]. Therefore the lower and upper bounds are $\Psi''(\sqrt{\frac{3}{2}}) = -\frac{4}{e^{\frac{3}{2}}}, \Psi''(0^+) = 2$, i.e.,

$$0 \leq \Psi(w) < 1, \quad 0 \leq \Psi'(w) \leq \sqrt{\frac{2}{e}}, \quad -\frac{4}{e^{\frac{3}{2}}} \leq \Psi''(w) \leq 2, \quad \forall w > 0.$$

Further, for all $w \in \mathbb{R}$, the following holds:

$$\Phi(w) = 4 \arctan w + 2\pi, \quad \Phi'(w) = \frac{4}{w^2+1}, \quad \Phi''(w) = -\frac{8w}{(w^2+1)^2}. \tag{2.39}$$

Similarly, as above, we can obtain the following bounds:

$$0 < \Phi(w) < 4\pi, \quad 0 < \Phi'(w) \leq 4, \quad -\frac{3\sqrt{3}}{2} \leq \Phi''(w) \leq \frac{3\sqrt{3}}{2}, \quad \forall w \in \mathbb{R}.$$

We refer the readers to Fig. 2.2 – Fig. 2.3 for illustrations of these functions.

To show Property 3), note that for all $w \geq 1$ and $|v| < 1$,

$$\Psi(w)\Phi'(v) > \Psi(1)\Phi'(1) = 2(1 - e^{-1}) > 1$$

where the first inequality is true because $\Psi(w)$ is strictly increasing and $\Phi'(v)$ is strictly decreasing for all $w > 0$ and $v > 0$, and that $\Phi'(v) = \Phi'(|v|)$.

Next we show Property 4). Note that $0 \leq \Psi(w) < 1$ and $0 < \Phi(w) < 4\pi$. Therefore we have $h(0) = -\Psi(1)\Phi(0) < 0$ and using the construction in (2.33)

$$\inf_{x_i} h_i(x_i) \geq -\Psi(1)\Phi(x_i[1]) - 3\sum_{j=1}^{\lfloor T/2 \rfloor} \sup_{w,v} \Psi(w)\Phi(v) \tag{2.40}$$

$$\geq -4\pi - 6\pi T \geq -10\pi T, \tag{2.41}$$

where the first inequality follows $\Psi(w)\Phi(v) > 0$ and second follows $\Psi(w)\Phi(v) < 4\pi$, we reach the conclusion.

Finally we show Property 5), using the fact that a function is Lipschitz if it is piecewise smooth with bounded derivative. From construction (2.33), the first-order

partial derivative of $h_q(y)$ can be expressed below.

Case I) If i is even, we have

$$\frac{\partial h_q}{\partial y[i]} = \begin{cases} 3\left(-\Psi\left(-y[i-1]\right)\Phi'\left(-y[i]\right) - \Psi\left(y[i-1]\right)\Phi'\left(y[i]\right)\right), & q \in [1, \frac{M}{3}] \\ 0, & q \in [\frac{M}{3}+1, \frac{2M}{3}] \\ 3\left(-\Psi'\left(-y[i]\right)\Phi\left(-y[i+1]\right) - \Psi'\left(y[i]\right)\Phi\left(y[i+1]\right)\right), & q \in [\frac{2M}{3}+1, M] \end{cases} . \tag{2.42}$$

Case II) If i is odd but not 1, we have

$$\frac{\partial h_q}{\partial y[i]} = \begin{cases} 3\left(-\Psi'\left(-y[i]\right)\Phi\left(-y[i+1]\right) - \Psi'\left(y[i]\right)\Phi\left(y[i+1]\right)\right), & q \in [1, \frac{M}{3}] \\ 0, & q \in [\frac{M}{3}+1, \frac{2M}{3}] \\ 3\left(-\Psi\left(-y[i-1]\right)\Phi'\left(-y[i]\right) - \Psi\left(y[i-1]\right)\Phi'\left(y[i]\right)\right), & q \in [\frac{2M}{3}+1, M] \end{cases} . \tag{2.43}$$

Case III) If $i = 1$, we have

$$\frac{\partial h_q}{\partial y[1]} = \begin{cases} -\Psi(1)\Phi'(y[1]) + 3\left(-\Psi'\left(-y[1]\right)\Phi\left(-y[2]\right) - \Psi'\left(y[1]\right)\Phi\left(y[2]\right)\right), & q \in [1, \frac{M}{3}] \\ -\Psi(1)\Phi'(y[1]), & q \in [\frac{M}{3}+1, M] \end{cases} . \tag{2.44}$$

Obviously, $\frac{\partial h_q}{\partial y[i]}$ is a piecewise smooth function for any i, q, and it either equals zero or is separated at the non-differentiable point $y[i] = 0$ because of the function Ψ.

Further, fix a point $y \in \mathbb{R}^T$ and a unit vector $v \in \mathbb{R}^T$ where $\sum_{i=1}^{T} v[i]^2 = 1$. Define

$$g_q(\theta; y, v) := h_q(y + \theta v)$$

to be the directional projection of h_q on to the direction v at point y. We will show that there exists $\ell > 0$ such that $|g_q{}''(0; y, v)| \leq \ell$ for all $y \neq 0$ (where the second-order derivative is taken with respect to θ).

First we can compute $g_q{}''(0; y, v)$ as follows:

$$g_q^{''}(0; y, v) = \sum_{i_1, i_2 = 1}^{T} \frac{\partial^2}{\partial y[i_1]\partial y[i_2]} h_q(y)\, v[i_1]v[i_2] = \sum_{\delta \in \{0,1,-1\}} \sum_{i=1}^{T} \frac{\partial^2}{\partial y[i]\partial y[i+\delta]} h_q(y)\, v[i]v[i+\delta],$$

where we take $v[0] := 0$ and $v[T+1] := 0$.

The second-order partial derivative of $h_q(y)$ $(\forall y \neq 0)$ is given as follows when i is

even:

$$\frac{\partial^2 h_q}{\partial y[i]\partial y[i]} = \begin{cases} 3\left(\Psi\left(-y[i-1]\right)\Phi''\left(-y[i]\right) - \Psi\left(y[i-1]\right)\Phi''\left(y[i]\right)\right), & q \in [1, \frac{M}{3}] \\ 0, & q \in [\frac{M}{3}+1, \frac{2M}{3}] \\ 3\left(\Psi''\left(-y[i]\right)\Phi\left(-y[i+1]\right) - \Psi''\left(y[i]\right)\Phi\left(y[i+1]\right)\right), & q \in [\frac{2M}{3}+1, M] \end{cases} \tag{2.45}$$

$$\frac{\partial^2 h_q}{\partial y[i]\partial y[i+1]} = \begin{cases} 0, & q \in [1, \frac{2M}{3}] \\ 3\left(\Psi'\left(-y[i]\right)\Phi'\left(-y[i+1]\right) - \Psi'\left(y[i]\right)\Phi'\left(y[i+1]\right)\right), & q \in [\frac{2M}{3}+1, M] \end{cases} \tag{2.46}$$

$$\frac{\partial^2 h_q}{\partial y[i]\partial y[i-1]} = \begin{cases} 3\left(\Psi'\left(-y[i-1]\right)\Phi'\left(-y[i]\right) - \Psi'\left(y[i-1]\right)\Phi'\left(y[i]\right)\right), & q \in [1, \frac{M}{3}] \\ 0, & q \in [\frac{M}{3}+1, M] \end{cases} . \tag{2.47}$$

By applying Lemma 2.3.1 – i) [i.e., $\Psi(w) = \Psi'(w) = \Psi''(w) = 0$ for $\forall\ w \leq 0$], we immediately obtain that at least one of the terms $\Psi\left(-y[i-1]\right)\Phi''\left(-y[i]\right)$ or $-\Psi\left(y[i-1]\right)\Phi''\left(y[i]\right)$ is zero. It follows that

$$\Psi\left(-y[i-1]\right)\Phi''\left(-y[i]\right) - \Psi\left(y[i-1]\right)\Phi''\left(y[i]\right) \leq \sup_w |\Psi(w)| \sup_v |\Phi''(v)|.$$

Similarly,

$$\Psi''\left(-y[i]\right)\Phi\left(-y[i+1]\right) - \Psi''\left(y[i]\right)\Phi\left(y[i+1]\right) \leq \sup_w |\Psi''(w)| \sup_v |\Phi(v)|$$

$$\Psi'\left(-y[i]\right)\Phi'\left(-y[i+1]\right) - \Psi'\left(y[i]\right)\Phi'\left(y[i+1]\right) \leq \sup_w |\Psi'(w)| \sup_v |\Phi'(v)|.$$

Therefore, take the maximum over equations (2.45) to (2.47) and plug in the above inequalities, we obtain

$$\begin{aligned} \left|\frac{\partial^2 h_q}{\partial y[i_1]\partial y[i_2]}\right| &\leq 3\max\{\sup_w |\Psi''(w)| \sup_v |\Phi(v)|, \sup_w |\Psi(w)| \sup_v |\Phi''(v)|, \sup_w |\Psi'(w)| \sup_v |\Phi'(v)|\} \\ &= 3\max\left\{8\pi, \frac{3\sqrt{3}}{2}, 4\sqrt{\frac{2}{e}}\right\} < 25\pi, \quad \forall\ i_1 \text{ being even, } \forall\ i_2 \end{aligned}$$

where the equality comes from Lemma 2.3.1 – ii).

We can also verify that the above bound for i being odd but not 1 is exactly the same.

When $i = 1$ we have following:

$$\frac{\partial^2 h_q}{\partial y[1]\partial y[1]} = \begin{cases} -\Psi(1)\Phi''(y[1]) + 3\left(-\Psi''\left(-y[1]\right)\Phi\left(-y[2]\right) - \Psi''\left(y[1]\right)\Phi\left(y[2]\right)\right), & q \in [1, \frac{M}{3}] \\ -\Psi(1)\Phi''(y[1]), & q \in [\frac{M}{3}+1, M] \end{cases}$$

$$\frac{\partial^2 h_q}{\partial y[1]\partial y[2]} = \begin{cases} 3\left(-\Psi'\left(-y[1]\right)\Phi'\left(-y[2]\right) - \Psi'\left(y[1]\right)\Phi'\left(y[2]\right)\right), & q \in [1, \frac{M}{3}] \\ 0, & q \in [\frac{M}{3}+1, M] \end{cases}$$

Again by applying Lemma 2.3.1 – i) and ii),

$$\begin{aligned} \left|\frac{\partial^2 h_q}{\partial y[1]\partial y[i_2]}\right| &\leq \max\{\sup_w |\Psi(1)\Phi''(w)| + 3\sup_w |\Psi''(w)| \sup_v |\Phi(v)|, 3\sup_w |\Psi'(w)| \sup_v |\Phi'(v)|\} \\ &= \max\left\{\frac{3\sqrt{3}}{2}(1-e^{-1}) + 24\pi, 12\sqrt{\frac{2}{e}}\right\} < 25\pi, \ \forall\ i_2. \end{aligned}$$

Summarizing the above results, we obtain:

$$\begin{aligned} |g_q''\left(0; y, v\right)| &= |\sum_{\delta\in\{0,1,-1\}} \sum_{i=1}^{T} \frac{\partial^2}{\partial y[i]\partial y[i+\delta]} h_q\left(y\right) v[i]v[i+\delta]| \\ &\leq 25\pi \sum_{\delta\in\{0,1,-1\}} |\sum_{i=1}^{T} v[i]v[i+\delta]| \\ &= 25\pi\left(|\sum_{i=1}^{T} v[i]^2| + 2|\sum_{i=1}^{T} v[i]v[i+1]|\right) \\ &\leq 75\pi \sum_{i=1}^{T} |v[i]^2| = 75\pi. \end{aligned}$$

Overall, the first-order derivatives of h_q are Lipschitz continuous for any q with constant $\ell = 75\pi$.

To show the same result for the function $\bar{h}$, we can apply (2.19). This completes the proof. **Q.E.D.**

The following lemma is a simple extension of the previous result.

Lemma 2.3.2. *We have the following properties for the functions f and $\bar{f}$ defined in* (2.36) *and* (2.35).

1. *We have* $\forall\ x \in \mathbb{R}^{TM\times 1}$

$$f(0) - \inf_x f(x) + \frac{1}{MU}\|d_0\|^2 \leq \frac{1650\pi^2\epsilon}{U}T,$$

where we have defined

$$d_0 := [\nabla f_1(0), \cdots, \nabla f_M(0)]. \tag{2.48}$$

2. *We have*

$$\left\|\nabla \bar{f}(y)\right\| = \sqrt{2\epsilon}\left\|\nabla \bar{h}\left(\frac{yU}{75\pi\sqrt{2\epsilon}}\right)\right\|,\ \forall\ y \in \mathbb{R}^{T\times 1}. \tag{2.49}$$

3. *The first-order derivatives of $\bar{f}$ and that for each $f_j, j \in [M]$ are Lipschitz continuous, with the same constant $U > 0$.*

Proof. To show that property 1) is true, note that from the definition of $f_i(x_i)$ we have

$$\nabla f_i(x_i) = \sqrt{2\epsilon} \times \nabla h_i\left(\frac{x_i U}{75\pi\sqrt{2\epsilon}}\right).$$

Therefore the following holds:

$$\begin{aligned}\frac{1}{M}\|d_0\|^2 &= \frac{2\epsilon}{M}\sum_{i=1}^{M}\|\nabla h_i(0)\|^2 \\ &= \frac{2\epsilon}{M}\sum_{i=1}^{M}|\Psi(1)\Phi'(0)|^2 = 32\epsilon(1-\exp(-1))^2.\end{aligned} \tag{2.50}$$

Therefore we have the following:

$$f(0) - \inf_x f(x) + \frac{\|d_0\|^2}{MU} = \frac{150\pi\epsilon}{U}\left(h(0) - \inf_x h(x) + \frac{16(1-\exp(-1))^2}{75\pi}\right).$$

Then by applying Lemma 2.3.1 we have that for any $T \geq 1$, the following holds

$$f(0) - \inf_x f(x) + \frac{\|d_0\|^2}{MU} \leq \frac{150\pi\epsilon}{U} \times (10\pi T + 0.03) \leq \frac{150\pi\epsilon}{U} \times 11\pi T.$$

Property 2) is true due to the definition of $\bar{f}$.

Property 3) is true because the following

$$\|\nabla \bar{f}(z) - \nabla \bar{f}(y)\| = \sqrt{2\epsilon} \left\| \nabla \bar{h}\left(\frac{zU}{75\pi\sqrt{2\epsilon}}\right) - \nabla \bar{h}\left(\frac{yU}{75\pi\sqrt{2\epsilon}}\right) \right\| \leq U\|z - y\|$$

where the last inequality comes from Lemma 2.3.1 – (5). This completes the proof. **Q.E.D.**

Next let us analyze the size of $\nabla \bar{h}$. We have the following result.

Lemma 2.3.3. *If there exists $k \in [T]$ such that $|y[k]| < 1$, then*

$$\left\|\nabla \bar{h}(y)\right\| = \left\| \frac{1}{M} \sum_{i=1}^{M} \nabla h_i(y) \right\| \geq \left| \frac{1}{M} \sum_{i=1}^{M} \frac{\partial}{\partial y[k]} h_i(y) \right| > 1.$$

Proof. The first inequality holds for all $k \in [T]$, since $\frac{1}{M}\sum_{i=1}^{M} \frac{\partial}{\partial y[k]} h_i(y)$ is one element of $\frac{1}{M}\sum_{i=1}^{M} \nabla h_i(y)$.

We divide the proof for second inequality into two cases.

Case 1. Suppose $|y[j-1]| < 1$ for all $2 \leq j \leq k$. Therefore, we have $|y[1]| < 1$. Using (2.44), we have the following inequalities:

$$\frac{\partial}{\partial y[1]} h_i(y) \overset{\text{(i)}}{\leq} -\Psi(1)\Phi'(y[1]) \overset{\text{(ii)}}{<} -1, \forall i \tag{2.51}$$

where (i) is true because $\Psi'(w), \Phi(w)$ are all non-negative from Lemma 2.3.1 -(2); (ii) is true due to Lemma 2.3.1 – (3). Therefore, we have the following

$$\left\|\nabla \bar{h}(y)\right\| = \left\| \frac{1}{M} \sum_{i=1}^{M} \nabla h_i(y) \right\| \geq \left| \frac{1}{M} \sum_{i=1}^{M} \frac{\partial}{\partial y[1]} h_i(y) \right| > 1.$$

Case 2) Suppose there exists $2 \leq j \leq k$ such that $|y[j-1]| \geq 1$.

We choose j so that $|y[j-1]| \geq 1$ and $|y[j]| < 1$. Therefore, depending on the choices

of (i,j) we have three cases

$$\frac{\partial h_i(y)}{\partial y[j]} = \begin{cases} -3\left(\Psi\left(-y[i-1]\right)\Phi'\left(-y[j]\right) + \Psi\left(y[i-1]\right)\Phi'\left(y[j]\right)\right), & i \in [1, \frac{M}{3}] \\ 0, & i \in [\frac{M}{3}+1, \frac{2M}{3}] \\ -3\left(\Psi'\left(-y[j]\right)\Phi\left(-y[i+1]\right) + \Psi'\left(y[j]\right)\Phi\left(y[i+1]\right)\right), & i \in [\frac{2M}{3}+1, M] \end{cases}.$$

If $i \in [1, \frac{M}{3}]$, because $|y[j-1]| \geq 1$ and $|y[j]| < 1$, using Lemma 2.3.1 – (3), and the fact that the negative part is zero for Ψ, and Φ' is even function, the expression further equals to

$$-3 \cdot \Psi(|y[j-1]|)\Phi'\left(|y[j]|\right)] \overset{(2.37)}{<} -3, \tag{2.52}$$

If $i \in [\frac{2M}{3}+1, M]$ the expression is obviously non-positive because both Ψ' and Φ are nonnegative. Overall, we have

$$\left|\frac{1}{M}\sum_{i=1}^{M}\frac{\partial h_i(y)}{\partial y[j]}\right| > \left|\frac{1}{M}\sum_{i=1}^{M/3} 3\right| = 1.$$

This completes the proof. **Q.E.D.**

Lemma 2.3.4. *Define $\bar{x} := \frac{1}{M}\sum_{i=1}^{M} x_i$, and assume that $U \in (0,1)$. Then we have*

$$\left\|\frac{1}{M}\sum_{i=1}^{M}\nabla f_i(x_i)\right\|^2 + \frac{U}{M\underline{\lambda}_{\min}(P^{1/2}\mathcal{L}P^{1/2})}\sum_{(i,j):i\sim j}\|x_i - x_j\|^2 \geq \frac{1}{2}\left\|\nabla \bar{f}(\bar{x})\right\|^2.$$

Proof. First let us derive a useful property. Define $d := [d_1; d_2; \cdots; d_M]$ where d_i is the degree for node i; further define

$$\bar{x} := \frac{1}{M}\sum_{i=1}^{M} x_i, \quad \tilde{x_i} := x_i - \bar{x}, \quad \tilde{x} := [\tilde{x}_1; \tilde{x}_2; \cdots; \tilde{x}_M].$$

It is easy to observe that :

$$\tilde{x}^T\mathbb{1} = 0, \quad \text{and} \quad \tilde{x} \notin \text{Null}(F^TF).$$

Then the following holds:

$$x^T F^T F x = \sum_{(i,j):i\sim j} \|x_i - x_j\|^2 = \sum_{(i,j):i\sim j} \|\tilde{x}_i - \tilde{x}_j\|^2 = \tilde{x}^T F^T F \tilde{x} \geq \underline{\lambda}_{\min}(F^T F)\|\tilde{x}\|^2. \tag{2.53}$$

Therefore the following holds:

$$\sum_{i=1}^M \|\bar{x} - x_i\|^2 \leq \frac{1}{\underline{\lambda}_{\min}(F^T F)} \sum_{(i,j):i\sim j} \|x_i - x_j\|^2 = \frac{1}{\underline{\lambda}_{\min}(P^{1/2}\mathcal{L}P^{1/2})} \sum_{(i,j):i\sim j} \|x_i - x_j\|^2. \tag{2.54}$$

Based on the above property, we have the following series of inequalities

$$\begin{aligned}
\left\|\nabla \bar{f}(\bar{x})\right\|^2 &\leq 2\left\|\frac{1}{M}\sum_{i=1}^M \left(\nabla f_i(\bar{x}) - \nabla f_i(x_i)\right)\right\|^2 + 2\left\|\frac{1}{M}\sum_{i=1}^M \nabla f_i(x_i)\right\|^2 \\
&\overset{\text{(i)}}{\leq} \frac{2}{M}\sum_{i=1}^M \left\|\nabla f_i(\frac{1}{M}\sum_{j=1}^M x_j) - \nabla f_i(x_i)\right\|^2 + 2\left\|\frac{1}{M}\sum_{i=1}^M \nabla f_i(x_i)\right\|^2 \\
&\overset{\text{(ii)}}{\leq} \frac{2}{M}\sum_{i=1}^M U^2 \left\|\frac{1}{M}\sum_{j=1}^M x_j - x_i\right\|^2 + 2\left\|\frac{1}{M}\sum_{i=1}^M \nabla f_i(x_i)\right\|^2 \\
&\overset{\text{(iii)}}{\leq} \frac{2U}{M\underline{\lambda}_{\min}(P^{1/2}\mathcal{L}P^{1/2})} \sum_{(i,j):i\sim j} \|x_j - x_i\|^2 + 2\left\|\frac{1}{M}\sum_{i=1}^M \nabla f_i(x_i)\right\|^2,
\end{aligned}$$

where in (i) and (iii) we have used the convexity of the function $\|\cdot\|^2$; in (ii) we used Lemma 2.3.2 – (3); in (iii) we have also used the assumption that $U \in (0,1)$ and (2.54). Overall we have

$$\left\|\frac{1}{M}\sum_{i=1}^M \nabla f_i(x_i)\right\|^2 + \frac{U}{M\underline{\lambda}_{\min}(P^{1/2}\mathcal{L}P^{1/2})} \sum_{(i,j):i\sim j} \|x_i - x_j\|^2 \geq \frac{1}{2}\left\|\nabla f(\bar{x})\right\|^2.$$

This completes the proof. **Q.E.D.**

Lemma 2.3.5. *Consider using an algorithm in class $\mathcal{A}$ or in class $\mathcal{A}'$ to solve the*

following problem:

$$\min_{x\in\mathbb{R}^{TM\times 1}} \; h(x) = \frac{1}{M}\sum_{i=1}^{M} h_i(x_i), \tag{2.55}$$

over a path graph. Assume the initial solution: $x_i = 0,\ \forall\ i \in [M]$. *Let* $\bar{x} = \frac{1}{M}\sum_{i=1}^{M} x_i$ *denote the average of the local variables. Then the algorithm needs at least* $(\frac{M}{3}+1)T$ *iterations to have* $x_i[T] \neq 0,\ \forall\ i$ *and* $\bar{x}[T] \neq 0$.

Proof. For a given $k \geq 2$, suppose that $x_i[k], x_i[k+1], ..., x_i[T] = 0,\ \forall i$, that is, $\text{support}\{x_i\} \subseteq \{1, 2, 3, ..., k-1\}$ for all i. Then $\Psi'\left(x_i[k]\right) = \Psi'\left(-x_i[k]\right) = 0$ for all i, and h_i has the following partial derivative when k is even:

$$\frac{\partial h_i(x_i)}{\partial x_i[k]} = \begin{cases} -3\left(\Psi\left(-x_i[k-1]\right)\Phi'\left(-x_i[k]\right)\right) + 3\left(\Psi\left(x_i[k-1]\right)\Phi'\left(x_i[k]\right)\right), & i \in [1, \frac{M}{3}] \\ 0, & i \in [\frac{M}{3}+1, M] \end{cases} \tag{2.56}$$

and the following partial derivative when k is odd and $k \geq 3$:

$$\frac{\partial h_i(x_i)}{\partial x_i[k]} = \begin{cases} 0, & i \in [1, \frac{2M}{3}] \\ -3\left(\Psi\left(-x_i[k-1]\right)\Phi'\left(-x_i[k]\right)\right) + 3\left(\Psi\left(x_i[k-1]\right)\Phi'\left(x_i[k]\right)\right), & i \in [\frac{2M}{3}+1, M] \end{cases} . \tag{2.57}$$

Recall that for any algorithm in class $\mathcal{A}$ or $\mathcal{A}'$, each agent is only able to compute linear combination of historical gradient and neighboring iterates [cf. (2.15) and (2.16)]. Therefore, for a given node i, as long as the kth element of the gradient as well as that of its neighbors have never been updated once, $x_i[k]$ remains to be zero. Combining this observation with the above two expressions for $\frac{\partial h_i(x_i)}{\partial x_i[k]}$, we can conclude that when $\text{support}\{x_i\} \subseteq \{1, 2, 3, ..., k-1\}$ for all i, then in the next iteration $x_i[k]$ will be possibly non-zero on the node $i \in [1, \frac{M}{3}]$ for even k and $i \in [\frac{2M}{3}+1, M]$ for odd k, and all other nodes still have $x_j[k] = 0,\ \forall\ j \neq i$.

Now suppose that the initial solution is $x_i[k] = 0$ for all (i, k). Then at the first iteration only $\frac{\partial h_i(x_i)}{\partial x_i[1]}$ is non-zero for all i, due to the fact that $\frac{\partial h_i(x_i)}{\partial x_i[1]} = \Psi(1)\Phi'(0) = 4(1-e^{-1})$ for all i from (2.44). If follows that even if every node is able to compute its local gradient, and can communicate with their neighbors, it is only possible to have $x_i[1] \neq 0, \forall i$. At the second iteration, we can use (2.56) to conclude that it is only

possible to have $\frac{\partial h_r(x_r)}{\partial x_r[k]} \neq 0$ for some $r \in [1,\ M/3]$, therefore when using an algorithm in class $\mathcal{A}$, we can conclude that $x_i[2] = 0$ for all $i \notin [1,\ M/3]$.

Then following our construction (2.33), we know the nodes in the set $[1, \frac{M}{3}]$ and the set $[\frac{2M}{3} + 1, M]$ have minimum distance $M/3$. It follows that using an algorithm in $\mathcal{A}$ or $\mathcal{A}'$, it takes at least $M/3$ iterations for the non-zero $x_r[2]$ and the corresponding gradient vector to propagate to at least one node in set $[2M/3 + 1, M]$. Once we have $x_j[2] \neq 0$ for some $j \in [2M/3 + 1,\ M]$, then according to (2.57), it is possible to have $\frac{\partial h_j(x_j)}{\partial x_j[3]} \neq 0$, and once this gradient becomes non-zero, the corresponding variable $x_j[3], j \in [2M/3 + 1,\ M]$ can become nonzero.

Following the above procedure, it is clear that we need at least $\frac{MT}{3}$ iterates and T computations to make $x_i[T]$ possibly non-zero. **Q.E.D.**

Theorem 2.3.1. *Let $U \in (0, 1)$ and ϵ be positive. Let $x^0[i] = 0$ for all $i \in [M]$. Then for any distributed first-order algorithm in class $\mathcal{A}$ or $\mathcal{A}'$, there exists a problem in class $\mathcal{P}_U^M$ and a network in class $\mathcal{N}$, such that it requires at least the following number of iterations*

$$t \geq \frac{1}{3\sqrt{\xi(\mathcal{G})}} \left\lfloor \frac{\left(f(0) - \inf_x f(x) + \frac{\|d_0\|^2}{MU}\right) U}{1650\pi^2} \epsilon^{-1} \right\rfloor \tag{2.58}$$

to achieve the following error

$$h_t^* = \left\| \frac{1}{M} \sum_{i=1}^{M} \nabla f_i(x_i^t) \right\|^2 + \frac{U}{M\underline{\lambda}_{\min}(P^{1/2}\mathcal{L}P^{1/2})} \sum_{(i,j):i\sim j} \|x_i^t - x_j^t\|^2 < \epsilon. \tag{2.59}$$

Proof. By Lemma 2.3.5 we have $\bar{x}[T] = 0$ for all $t < \frac{M+3}{3}T$. Then by applying Lemma 2.3.2 – (2) and Lemma 2.3.3, we can conclude that the following holds

$$\left\|\nabla \bar{f}(\bar{x}[T])\right\| = \sqrt{2\epsilon} \left\| \nabla \bar{h} \left(\frac{\bar{x}[T]U}{75\pi\sqrt{2\epsilon}} \right) \right\| > \sqrt{2\epsilon}, \tag{2.60}$$

where the second inequality follows that there exists $k \in [T]$ such that $|\frac{\bar{x}[k]U}{75\pi\sqrt{2\epsilon}}| = 0 < 1$, then we can directly apply Lemma 2.3.3. Then by applying Lemma 2.3.4 gives $h^*_{(M+3)T/3} > \epsilon$.

The third part of Lemma 2.3.2 ensures that f_i's are U-Lipschitz continuous gradient,

and the first part shows

$$f(0) - \inf_x f(x) + \frac{\|d_0\|^2}{MU} \leq \frac{1650\pi^2\epsilon}{U}T,$$

Therefore we obtain

$$T \geq \left\lfloor \frac{\left(f(0) - \inf_x f(x) + \frac{\|d_0\|^2}{MU}\right)U}{1650\pi^2}\epsilon^{-1} \right\rfloor. \tag{2.61}$$

Summarizing the above argument, we have

$$t \geq \frac{M+3}{3}T \geq \frac{M+3}{3}\left\lfloor \frac{\left(f(0) - \inf_x f(x) + \frac{\|d_0\|^2}{MU}\right)U}{1650\pi^2}\epsilon^{-1} \right\rfloor.$$

By noting that for path graph $\xi(\mathcal{G}) \geq 1/M^2$, this completes the proof. **Q.E.D.**

2.3.2 Generalization

The previous section analyzes the lower complexity bounds for problem $\mathcal{P}_U^M$ over a path network. The obtained results can be extended in a number of direction.

Uniform L_i, Fixed D and M

In this subsection, we would like to generalize Theorem 2.3.1 to a slightly wider class of networks (beyond the path graph used in our construction). Towards this end, consider a *path-star graph* shown in Fig. 2.5. The graph contains a path graph with $D-1$ nodes, and the remaining nodes are divided into $D-1$ groups, each with either $\lfloor M/(D-1)-1 \rfloor$ or $\lfloor M/(D-1)-1 \rfloor + 1$ nodes, and each group is connected to the nodes in the path graph by using a star topology. We have the following corollary to Theorem 2.3.1.

Corollary 2.3.1. *Let $U \in (0,1)$ and ϵ be positive, and fix any D and M such that $D \leq M-1$. For any algorithm in class $\mathcal{A}$ or $\mathcal{A}'$, there exists a problem in class $\mathcal{P}_U^M$ and a network in class $\mathcal{N}_D^M$, so that to achieve accuracy $h_t^* < \epsilon$, it requires at least the*

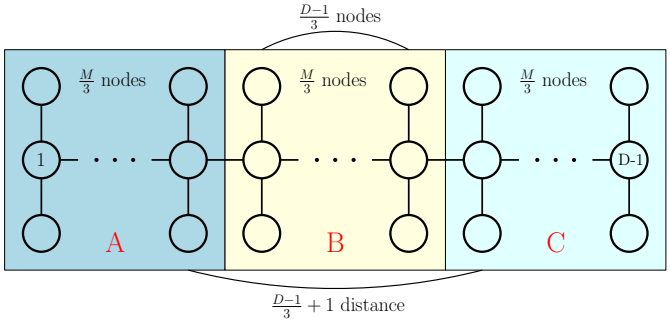


Figure 2.5: The path-star graph used in our construction.

following number iterations

$$t \geq \frac{D}{3}\left\lfloor \frac{\left(f(0) - \inf_x f(x) + \frac{\|d_0\|^2}{MU}\right)U}{1650\pi^2}\epsilon^{-1} \right\rfloor.$$

Alternatively, the above bound can be expressed as the following

$$t \geq \frac{\sqrt{(D-1)/(2M)}}{3\sqrt{\xi(\mathcal{G})}}\left\lfloor \frac{\left(f(0) - \inf_x f(x) + \frac{\|d_0\|^2}{MU}\right)U}{1650\pi^2}\epsilon^{-1} \right\rfloor.$$

Proof. Fix any D and M such that $D \leq M-1$, we can construct a path-star graph as described in Fig.2.5, whose diameter is D.

To show the lower bounds for such a graph, we split all M nodes into three sets $\mathcal{A}, \mathcal{B}, \mathcal{C}$ based on the main path, each with $\frac{M}{3}$ nodes (assume M is a multiple of 3), where $\mathcal{A}$ and $\mathcal{C}$ has minimum $\frac{D+2}{3}$ distance (assume $D-1$ is a multiple of 3). Then we construct the component functions h_i's as follows.

$$h_i(x_i) = \begin{cases} \Theta(x_i, 1) + 3\sum_{j=1}^{\lfloor T/2 \rfloor} \Theta(x_i, 2j), & i \in \mathcal{A} \\ \Theta(x_i, 1), & i \in \mathcal{B} \\ \Theta(x_i, 1) + 3\sum_{j=1}^{\lfloor T/2 \rfloor} \Theta(x_i, 2j+1), & i \in \mathcal{C} \end{cases} \tag{2.62}$$

Since the graph has diameter D in the above construction, and the distance between

any two elements in $\mathcal{A}$ and $\mathcal{C}$ is at least $\frac{D+2}{3}$ (assume $D-1$ is a multiple of 3), by a similar step in Lemma 2.3.5 we can conclude that we need at least $(\frac{D+2}{3}+1)T$ iterations to achieve $x_i[T] \neq 0$. By applying (2.61), we can obtain the desired result.

To show the second result, note that from (2.28) we have

$$\sum_i d_i D \geq \frac{1}{\underline{\lambda}_{\min}(\mathcal{L})} \tag{2.63}$$

For the path-star graph under consideration,

$$\sum_i d_i \leq 2(D-1) - 2 + 2\left(M - (D-1)\right) \leq 2M,$$

so the following holds:

$$D^2 \geq \frac{D/2M}{\underline{\lambda}_{\min}(\mathcal{L})} \geq \frac{(D-1)/(2M)}{\underline{\lambda}_{\min}(\mathcal{L})}.$$

The desired result is then immediate. **Q.E.D.**

Non-uniform L_i, Fixed $\mathcal{N}$

Finally, for the problem class with non-uniform Lipschitz constants, we can extend the previous result to any network in class $\mathcal{N}$ (by properly assigning different values of L_i's to different nodes). In this case the lower bound will be dependent on the spectrum property of $\widehat{\mathcal{L}}$ as defined in (2.26) (expressed below for easy reference)

$$\widehat{\mathcal{L}} := L^{-1/2} F^T K F L^{-1/2}. \tag{2.64}$$

Corollary 2.3.2. *Let ϵ be positive. For any given network in $\mathcal{N}_D^M$, and for any algorithm in $\mathcal{A}$, there exists a problem in $\mathcal{P}_L^M$ such that to achieve accuracy $h_t^* < \epsilon$, it requires at least the following iterations*

$$t \geq \frac{1}{3\sqrt{\xi(\widehat{\mathcal{L}})}} \left\lfloor \frac{\left(f(0) - \inf_x f(x) + \frac{\|d_0\|^2_{L^{-1}}}{M}\right)\bar{L}}{1650\pi^2} \epsilon^{-1} \right\rfloor. \tag{2.65}$$

To prove this result, we select the values of the coefficient set $\{L_i\}_{i=1}^M$, so that the

"effective" network topology becomes a path. In particular, for any given network in $\mathcal{N}$, we can construct local functions as follows: First, along the longest path of size D, we distributed the functions into three sets $\mathcal{A}, \mathcal{B}, \mathcal{C}$, where $\mathcal{A}$ and $\mathcal{C}$ denotes the first and last $\frac{D}{3}$ nodes on the path respectively, and $\mathcal{B}$ denotes the rest nodes on the path. Second, for the rest of the functions not on the path, denoted as set $\mathcal{D}$, set their local functions to zero (or equivalently, set the corresponding L_i's to zero). Then, the local function belongs to each set can be expressed as:

$$h_i(x_i) = \begin{cases} \frac{M}{D}\Theta(x_i, 1) + \frac{3M}{D}\sum_{j=1}^{\lfloor T/2 \rfloor} \Theta(x_i, 2j), & i \in \mathcal{A} \\ \frac{M}{D}\Theta(x_i, 1), & i \in \mathcal{B} \\ \frac{M}{D}\Theta(x_i, 1) + \frac{3M}{D}\sum_{j=1}^{\lfloor T/2 \rfloor} \Theta(x_i, 2j+1), & i \in \mathcal{C} \\ 0, & i \in \mathcal{D} \end{cases} \tag{2.66}$$

This way the network reduces to a path graph. Note that the Lipschitz constant for the gradient of $h(y) = \frac{1}{M}\sum_{i=1}^{M} h_i(y)$ is still 1, and we can use the similar constructions and proof steps leading to Theorem 2.3.1 to prove the claim.

Chapter 3

Optimal Algorithms for Distributed Non-convex Learning

We consider a class of popular distributed non-convex optimization problems, in which agents connected by a network $\mathcal{G}$ collectively optimize a sum of smooth (possibly non-convex) local objective functions. We address the following question: if the agents can only access the gradients of local functions, what are the *fastest* rates that any distributed algorithms can achieve (cf. Chapter 2), and how to achieve those rates (cf. Chapter 3).

In this Chapter, we propose (near) optimal methods whose rates match the developed lower rate bound (up to a ploylog factor) (cf. Chapter 2). The key in the algorithm design is to properly embed the classical polynomial filtering techniques into modern first-order algorithms. To the best of our knowledge, this is the first time that lower rate bounds and optimal methods have been developed for distributed non-convex optimization problems.

3.1 Introduction

3.1.1 Problem and motivation

In this work, we consider the same distributed optimization problem as in Chapter 2,

$$\min_{y\in\mathbb{R}^S} \ \bar{f}(y) := \frac{1}{M}\sum_{i=1}^{M} f_i(y), \tag{3.1}$$

where $f_i(y) : \mathbb{R}^S \to \mathbb{R}$ is a smooth and possibly non-convex function accessible to agent i. And its global consensus reformulation can be written as

$$\min_{x\in\mathbb{R}^{SM}} \ f(x) := \frac{1}{M}\sum_{i=1}^{M} f_i(x_i), \quad \text{s.t. } x_i = x_j, \forall\ (i,j) \in \mathcal{E}. \tag{3.2}$$

The main benefit of the above formulation is that the objective function is now separable, and the linear constraint encodes the network connectivity pattern.

3.1.2 Distributed non-convex optimization

Distributed non-convex optimization has gained considerable attention recently. For example, it finds applications in training neural networks [20], clustering [43], and dictionary learning [14], just to name a few.

The problem (3.1) and (3.2) have been studied extensively in the literature when f_i's are all convex; see for example [44–46]. Primal based methods such as distributed subgradient (DSG) method [44], the EXTRA method [46], as well as primal-dual based methods such as distributed augmented Lagrangian method [47], Alternating Direction Method of Multipliers (ADMM) [48, 49] have been proposed.

On the contrary, only recently there have been works addressing the more challenging problems without assuming convexity of f_i; see [8–23]. The convergence behavior of the distributed consensus problem (3.1) has been studied in [8,9,14]. Reference [10] develops a non-convex ADMM based methods for solving the distributed consensus problem (3.1). However the network considered therein is a star network in which the local nodes are all connected to a central controller. References [12,13] propose a primal-dual based method for unconstrained problem over a connected network, and derives a global convergence

Table 3.1: The main results of the paper when specializing to a few popular graphs.

Network Instances	Problem Classes		Rate Achieving Algorithm
	Uniform Lipschitz U	Non-uniform Lipschitz $\{L_i\}$	
Complete/Star	$\mathcal{O}(U/\epsilon)$	$\mathcal{O}(1/\epsilon \times \sum_i L_i/M)$	D-GPDA (proposed)
Random Geometric	$\widetilde{\mathcal{O}}(U\sqrt{M}/(\sqrt{\log M}\epsilon))$	$\widetilde{\mathcal{O}}(\sqrt{M}/(\sqrt{\log(M)}\epsilon) \times \sum_i L_i/M)$	xFILTER (proposed)
Path/Circle	$\widetilde{\mathcal{O}}(UM/\epsilon)$	$\widetilde{\mathcal{O}}(M/\epsilon \times \sum_i L_i/M)$	xFILTER (proposed)
Grid	$\widetilde{\mathcal{O}}(U\sqrt{M}/\epsilon)$	$\widetilde{\mathcal{O}}(\sqrt{M}/\epsilon \times \sum_i L_i/M)$	xFILTER (proposed)
Centralized	$\mathcal{O}(U/\epsilon)$	$\mathcal{O}(1/\epsilon \times \sum_i L_i/M)$	Gradient Descent

The entries show the best rate bounds achieved by the proposed algorithms (either D-GPDA or xFILTER) for a number of specific graphs and problem class; L_i is the Lipschitz constant for ∇f_i [see (2.4)]; for the uniform case $U = L_1, \cdots, L_M$. For the uniform Lipschitz the lower rate bounds derived for the particular graph matches the upper rate bounds (we only show the latter in the table). The last row shows the rate achieved by the centralized gradient descent algorithm. The notation $\widetilde{\mathcal{O}}$ denotes big $\mathcal{O}$ with some polynomial in logarithms, i.e, use $\widetilde{\mathcal{O}}$ to denote $\mathcal{O}(\log(M))$ where M is the problem dimension.

rate for this setting. In [11, 16, 17], the authors utilize certain gradient tracking idea to solve a constrained nonsmooth distributed problem over possibly time-varying networks. The work [18] summarizes a number of recent progress in extending the DSG-based methods for non-convex problems. References [15,19,20] develop methods for distributed stochastic zeroth and/or first-order non-convex optimization. It is worth noting that the distributed algorithms proposed in all these works converge to first-order stationary solutions, which contain local maximum, local minimum and saddle points.

Recently, the authors of [22,50–52] have developed first-order distributed algorithms that are capable of computing second-order stationary solutions (which under suitable conditions become local optimal solutions). Other second-order distributed algorithms such as [53, 54] are design for convex problems, and they utilize high-order Hessian information about local problems.

3.1.3 Contribution of this work

Our main contributions in this Chapter are given below:

1) We design two algorithms belonging to $\mathcal{A}$, one based on primal-dual optimization scheme, the other based on a novel *approximate filtering -then- predict and tracking* (xFILTER) strategy, both of which achieve ϵ-optimality condition with provable global rates [in the order of $\mathcal{O}(1/\epsilon)$];

2) We show that the xFILTER is an optimal method in $\mathcal{A}$ for problem class $(\mathcal{P}, \mathcal{N})$ as well as a number of its refinements, in that they precisely achieve the lower complexity

bounds that we derived (up to a ploylog factor).

In Table 3.1, we specialize some key results developed in the paper to a few popular graphs, and compare them with the achievable rates of centralized GD.

3.2 The Proposed Algorithms

In this section, we introduce two different types of algorithms for solving problem (3.2). The algorithm is *near-optimal*, and can achieve the lower bounds derived in Section 2.3 except for a multiplicative polylog factor in M. To simplify the notation, we utilize the definitions introduced in Section 2.2, and rewrite problem (3.2) in the following compact form

$$\min_{x\in\mathbb{R}^{SM}} \; f(x) := \frac{1}{M}\sum_{i=1}^{M} f_i(x_i), \quad \text{s.t. } (F\otimes I_S)x = 0. \tag{3.3}$$

It can be verified that, by using the definition of F, the constraint in this problem is equivalent to the ones given in (3.2). For notational simplicity, in the following we will assume that $S = 1$ (scalar variables). All the results presented in subsequent sections extend easily to case with $S > 1$.

3.2.1 The D-GPDA Algorithm

We first present a Distributed Gradient Primal-Dual Algorithm (D-GPDA), which relaxes the linear constraint (3.3), and gradually enforces it as the algorithm proceeds. To describe the algorithm, let us introduce the augmented Lagrangian (AL) function as

$$\mathsf{AL}(x,\lambda) = f(x) + \langle \lambda, Fx\rangle + \frac{1}{2}\|\Sigma F x\|^2, \tag{3.4}$$

where $\lambda \in \mathbb{R}^E$ is the dual variable; $\Sigma = \mathrm{diag}([\sigma_1, \cdots, \sigma_E]) \in \mathbb{R}^{E\times E}$ is a diagonal positive definite matrix. In the following, we will use the shorthanded notation $\mathsf{AL}^r := \mathsf{AL}(x^r, \lambda^r)$ where r is the iteration counter.

Define a *penalty matrix* as

$$\Upsilon = \mathrm{diag}\{[\beta_1, \cdots, \beta_M]\} \succ 0. \tag{3.5}$$

Then the D-GPDA is described in the following table.

(S1). Assign each node $i \in \mathcal{N}$ with a parameter $\beta_i > 0$; Assign each edge $(ij) \in \mathcal{E}$ with a parameter $\sigma_{ij} > 0$;
(S2). At iteration $r = -1$, initialize $\lambda^{-1} = 0$ and $x^{-1} = 0$;
(S3). At iteration $r = 0$, set λ^0 and x^0 using the following:

$$\nabla f(x^{-1}) + (2\Delta + \Upsilon^2)x^0 = 0,\ \lambda^0 = \Sigma^2 F x^0; \tag{3.6}$$

Equivalently x^0 can be written as:

$$x_i^0 = \left(2\sum_{j:j\sim i}\sigma_{ij}^2 + \beta_i^2\right)^{-1} \nabla f_i(0)/M,\ \forall\ i \in [M]; \tag{3.7}$$

(S4). At each iteration $r+1$, $r \geq 0$, update variables by:

$$\begin{aligned} x^{r+1} = \arg\min_x\ & \langle \nabla f(x^r) + F^T\lambda^r, x - x^r\rangle \qquad (3.8a)\\ & + \frac{1}{2}\|\Sigma F x\|^2 + \frac{1}{2}\|\Sigma B(x - x^r)\|^2 + \frac{1}{2}\|\Upsilon(x - x^r)\|^2 \\ \lambda^{r+1} = \lambda^r + \Sigma^2 F x^{r+1}. & \qquad (3.8b) \end{aligned}$$

Algorithm 1: The D-GPDA Algorithm

We note that each iteration of the D-GPDA performs a gradient descent type step on the AL function, followed by taking a step of dual gradient ascent (with a *stepsize matrix* $\Sigma^2 \succ 0$). The term $\frac{1}{2}\|\Sigma B(x - x^r)\|^2$ used in (3.8a) is a *network proximal term* that regularizes the x update using network structure, and its presence is critical to ensure separability and distributed implementation (see Remark 3.2.3 below).

The D-GPDA is closely related to many classical primal-dual methods, such as the Uzawa method [55] (which has been recently utilized to solve linearly constrained *convex* problems [56]), and the proximal method of multipliers (prox-MM) [57, 58]. The latter method has been first developed by Rockafellar in [57], in which a proximal term has been added to the AL in order to make it strongly convex in each iteration. However, the theoretical results derived for Prox-MM in [57,58] are only valid for convex problems. It is also important to note that when the matrices Σ and Υ are specialized as multiples of identity matrices, that is, when $\Sigma = \sigma I_M$ and $\Upsilon = \kappa I_M$ for some $\sigma, \kappa > 0$,

then the D-GPDA reduces to the Prox-GPDA algorithm briefly discussed in our earlier work [13, Section 5], for solving a general linearly constrained problem.

3.2.2 The xFILTER Algorithm

Despite the fact that D-GPDA is conceptually simple, we will show shortly that it is only optimal for special network classes with small diameter [or large gap function $\xi(\mathcal{G})$], such as the complete/star networks (see Table 3.1 and our detailed analysis in Section 3.4). Intuitively, the issue is that having the *network proximal term* imposes very heavy regularization, enforcing the new iterates to be close to the old ones. This causes slow information propagation over the network.

In this section, we present a *near-optimal* algorithm that can achieve the lower bounds derived in Section 2.3 for a number of different graphs (up to some polylog factor in the problem dimension). To motivate our algorithm design, observe that the communication lower bound $\mathcal{O}(1/\sqrt{\xi(\mathcal{G})} \times \bar{L}/\epsilon)$ in Section 2.3 can be decomposed into the product two parts, $\mathcal{O}(1/\sqrt{\xi(\mathcal{G})})$ and $\mathcal{O}(\bar{L}/\epsilon)$, corresponding roughly to the communication efficiency and the computational complexity, respectively. Such a product form motivates us to *separate* the computation and communication tasks, and design a *double loop* algorithm to achieve the desired lower bound.

Our proposed algorithm is based on a novel *approximate filtering -then- predict and tracking* (xFILTER) strategy, which properly combines the modern first-order optimization methods and the classical polynomial filtering techniques. It is a "double-loop" algorithm, where in the outer loop local gradients are computed to extract information from local functions, while in the inner loop some filtering techniques are used to facilitate efficient information propagation. Please see **Algorithm 1** for the detailed description, from the system perspective. It is important to note that the algorithm contains an outer loop **(S3)**–**(S4)** and an inner loop **(S2)**, indexed by r and q, respectively. Further, the local gradient evaluation only appears in the outer loop step **(S3)**.

To understand the algorithm, we note that one important task of each agent is to update its local variable so that it is close to the average $\frac{1}{M}\sum_{i=1}^{M} x_i$. Let us use d_i to denote a local variable that *approximates* the above average. At the beginning of the algorithm, d_i is just a rough estimate of the average, so we have $d_i = \frac{1}{M}\sum_j x_j + e_i$,

where e_i is the *deviation* from the true average, and it can be viewed as some kind of "estimation noise". To gradually remove such a noise, in step **S1)** we resort to the so-called graph based joint bilateral filtering used for image denoising [59, 60], which can be formulated as the following regularized least squares problem:

$$x_*^{r+1} := \arg\min_{x\in\mathbb{R}^M} \frac{1}{2}\|x - d^r\|^2_{\Upsilon^2} + \frac{1}{2}x^\top F^\top \Sigma^2 F x, \tag{3.9}$$

where d^r is the noisy signal, F is a penalty high pass filter related to the graph structure (in our case, F is the adjacency matrix), and Σ^2 is a regularization parameter. Its solution, denoted as x_*^{r+1} as given below, will be close to the "unfiltered" signal d^r, while having reduced high frequency components, or high fluctuations across the components:

$$Rx_*^{r+1} = d^r, \quad \text{with} \quad R := \Upsilon^{-2}F^\top\Sigma^2 F + I_M. \tag{3.10}$$

It is important to note that if x_*^{r+1} indeed achieves consensus, then by (2.11) we have $F^\top\Sigma^2 F x_*^{r+1} = 0$, implying $x_*^{r+1} = d^r$, which says d^r should "track" x_*^{r+1}.

Unfortunately, the system (3.10) cannot be precisely solved in a distributed manner, because inverting R destroys its pattern about the network structure embedded in the product $F^\top\Sigma^2 F$. More specifically, $F^\top\Sigma^2 F$ is the weighted graph Laplacian matrix whose (i,j)th entry is nonzero if and only if node i, j are connected, but $(\Upsilon^{-2}F^\top\Sigma^2 F + I_M)^{-1}$ is a dense matrix without such a property. Therefore in **S2)**, we use a degree-Q Chebyshev polynomial to approximate x_*^{r+1}. The output, denoted as x^{r+1}, stays in a Krylov space $span\{d^r, Rd^r, \cdots, R^Q d^r\}$. Specifically, at each iteration, the only step that requires communication is the operation Ru, which is given by

$$\begin{aligned}(Ru_{q-1})[i] &= (\Upsilon^{-2}F^\top\Sigma^2 F u_{q-1})[i] + d_{q-1}[i] \\ &= \frac{1}{\beta_i^2}\sum_{j:j\sim i}\sigma_{ij}^2(u_{q-1}[j] - d^r[i]) + u_{q-1}[i],\ \forall\ i,\end{aligned} \tag{3.11}$$

so this step can be done distributedly, via one round of local message exchange.

After completing $Q > 0$ such Chebyshev iterations (3.13) (C-iteration for short), the obtained solution x^{r+1} will be an approximate solution to the system 3.10, with a

residual error vector ϵ^{r+1} as given below

$$Rx^{r+1} = d^r + R\epsilon^{r+1}, \text{ with } \quad \epsilon^{r+1} := x^{r+1} - x_*^{r+1}. \tag{3.12}$$

Up to this point, the filtering technique we have discussed aims at removing the "non-consensus" parts from a vector $d = [d_1, \cdots, d_N]^T$. However, recall that the goal of distributed optimization is not only to achieve consensus, but also to optimize the objective function $\sum_i f_i(x_i)$. Therefore, a *prediction* step **(S3)** is performed to incorporate the most up-to-date local gradient $\nabla f_i(x_i)$. Then a *tracking* step **(S4)** is performed to update d. Ideally, one would like the new d_i^{r+1} to have the following three properties: 1) It is close to the previous d_i^r; 2) it takes into consideration the new local gradient information offered by the "predicted" $\tilde{x}_i^{r+1}$; 3) it is a "low frequency" signal, meaning d_i^{r+1} and d_j^{r+1} are relatively close, for all $i \neq j$. Taking a closer look at the "tracking" step, we can see that all three components are included: It adds to the previous d^r the differences of the last two predictions, and it removes some non-consensus components among the local variables. The detailed algorithm is given in the Algorithm 2.

To end this subsection, we emphasize that, the use of the polynomial Chebyshev filtering requires Q vector communications steps every time that **(S2)** is performed. However, such a filtering step is critical to make the proposed algorithm achieve performance lower bounds predicted in Section 2.3. Intuitively, it helps to accelerate information propagation across the network. Indeed, as will be shown shortly, the number Q in **(S2)** is directly related to properties of the underlying graph. It is also somewhat surprising that the inner problem (3.9) is not required to be solved with *increased* accuracy. On the contrary, only a *fixed* number of filtering steps are needed.

3.2.3 Discussion

In this subsection, we establish some key connections between the two algorithms discussed so far, and provide some additional remarks.

First, we provide an important interpretation of the xFILTER strategy, which will help us subsequently provide an unified analysis framework for both D-GPDA and xFILTER. First, similarly as in the D-GPDA algorithm, let us introduce an auxiliary

(S1) [Initialization]. Assign each node $i \in \mathcal{N}$ with $\beta_i > 0$; Assign each edge $(ij) \in \mathcal{E}$ with $\sigma_{ij} > 0$; Initialize $x^{-1} = 0$, $d^{-1} = -\Upsilon^{-2}\nabla f(x^{-1})$ and $\tilde{x}^{-1} = x^{-1} - \Upsilon^{-2}\nabla f(x^{-1})$. Compute R by (3.10);
(S2) [Filtering]. At iteration $r+1$, $r \geq -1$: For a fixed constant $Q > 0$, run the following C-iterations (with parameters $\{\alpha_q, \tau\}$)

$$\begin{aligned} &u_0 = x^r,\ u_1 = (I - \tau R)u_0 + \tau d^r, \\ &u_q = \alpha_q(I - \tau R)u_{q-1} + (1-\alpha_q)u_{q-2} + \tau\alpha_q d^r,\ q = 2, \cdots, Q; \end{aligned} \tag{3.13}$$

Set $x^{r+1} = u_Q$;
(S3) [Prediction]. Compute $\tilde{x}^{r+1}$ by:

$$\tilde{x}^{r+1} = x^{r+1} - \Upsilon^{-2}\nabla f(x^{r+1}); \tag{3.14}$$

(S4) [Tracking]. Compute d^{r+1} by:

$$d^{r+1} = d^r + (\tilde{x}^{r+1} - \tilde{x}^r) - \Upsilon^{-2}F^\top\Sigma^2 F x^{r+1}. \tag{3.15}$$

Set $r = r+1$, go to **(S2)**.

Algorithm 2: The xFILTER Algorithm

variable $\lambda^r \in \mathbb{R}^E$, which is updated as follows:

$$\lambda^{r+1} = \lambda^r + \Sigma^2 F x^{r+1}. \tag{3.16}$$

Suppose $\lambda^{-1} = 0$, then according to (3.15) and (3.14) we have the following relationship

$$\begin{aligned} d^0 &:= -\Upsilon^{-2}\nabla f(x^{-1}) + (x^0 - \Upsilon^{-2}\nabla f(x^0) - (x^{-1} - \Upsilon^{-2}\nabla f(x^{-1}))) - \Upsilon^{-2}F^T\lambda^0 \\ &= x^0 - \Upsilon^{-2}\nabla f(x^0) - \Upsilon^{-2}F^T\lambda^0. \end{aligned}$$

By using the induction argument, we can show that for all $r \geq 0$, the following holds

$$d^r := x^r - \Upsilon^{-2}\nabla f(x^r) - \Upsilon^{-2}F^T\lambda^r. \tag{3.17}$$

Combining (3.10) and (3.17), we obtain the following useful alternative expressions of

(3.10) and (3.12):

$$\Upsilon^{-2}\Big(\nabla f(x^r)+F^\top(\lambda^r+\Sigma^2 F x_*^{r+1})\Big)+(x_*^{r+1}-x^r)=0 \tag{3.18a}$$

$$\Upsilon^{-2}\Big(\nabla f(x^r)+F^\top(\lambda^r+\Sigma^2 F x^{r+1})\Big)+(x^{r+1}-x^r)=R\epsilon^{r+1}. \tag{3.18b}$$

Using (3.18a), it is clear that x_*^{r+1} can be equivalently written as the optimal solution of the following problem:

$$x_*^{r+1}=\operatorname*{argmin}_x \langle \nabla f(x^r)+F^T\lambda^r, x-x^r\rangle+\frac{1}{2}\|\Sigma F x\|^2+\frac{1}{2}\|\Upsilon(x-x^r)\|^2. \tag{3.19}$$

The relations (3.16) and (3.19) together show that D-GPDA and xFILTER are closely related. However, we note that when comparing (3.19) with (3.8a), one key difference is that the *network proximal* term $\frac{1}{2}\|\Sigma B(x-x^r)\|^2$ used in D-GPDA is no longer used in xFILTER.

We have the following additional remarks on the proposed algorithms. *Remark 3.2.1.* **(Parameters)** It is important to note that in both Alg. 1 and 2, in the update of the primal and dual variables, some *"matrix parameters"* are used instead of scalar ones. In particular, the matrix Υ^2 is used as the primal *"proximal parameter"*, while Σ^2 is used as the *"dual stepsize"*. Using these matrices ensures that we can appropriately design parameters for each node/link, which is one key ingredient in ensuring the optimal rate.

Remark 3.2.2. **(Initialization)** The initialization steps in **(S2)** and **(S3)** of Alg. 1 can be done in a distributed manner. Each node i only requires to know the neighbors' σ_{ij}^2's in order to update x_i^0. Once x^0 is updated, λ^0 can be updated by using:

$$\lambda_{ij}^0=\sigma_{ij}^2(x_i^0-x_j^0),\quad \forall\ (i,j)\in E.$$

Remark 3.2.3. **(Distributed Implementation and Algorithm Classes)** To see how the D-GPDA can be executed distributedly, we write down the optimality condition of (3.8a). For notational simplicity, define:

$$H:=B^T\Sigma^2 B+\Upsilon^2. \tag{3.20}$$

Then we have

$$\nabla f(x^r) + F^T\lambda^r + F^T\Sigma^2 F x^{r+1} + H(x^{r+1} - x^r) = 0. \tag{3.21}$$

Rearranging, and using property (2.22), we have

$$\nabla f(x^r) + F^T\lambda^r + (2\Delta + \Upsilon^2)x^{r+1} - Hx^r = 0.$$

Subtracting the same equation from the rth iteration, and use the fact that $F^T(\lambda^r - \lambda^{r-1}) = F^T\Sigma^2 F x^r$, we have

$$x^{r+1} = x^r - \left(2\Delta + \Upsilon^2\right)^{-1}\left(\nabla f(x^r) - \nabla f(x^{r-1}) + (F^T\Sigma^2 F - H)x^r + Hx^{r-1}\right). \tag{3.22}$$

According to the above update rule, each node i can distributedly implement (3.22) by performing the following

$$\begin{aligned} x_i^{r+1} = x_i^r - \frac{1}{2\sum_{j:j\sim i}\sigma_{ij}^2 + \beta_i^2}\Bigg(&\frac{1}{M}(\nabla f_i(x_i^r) - \nabla f_i(x_i^{r-1})) \\ &- 2\sum_{j:j\sim i}\sigma_{ij}^2 x_j^r - \beta_i^2 x_i^r + \beta_i^2 x_i^{r-1} + \sum_{j:j\sim i}\sigma_{ij}^2(x_j^{r-1} + x_i^{r-1})\Bigg). \end{aligned} \tag{3.23}$$

It is also easy to see that the Chebyshev iteration in xFILTER can be implemented distributedly, since the R matrix defined in (3.10) preserves the network structure. To see how we can compute the d^r vector distributedly, we first note that $d^{-1} = -\Upsilon^{-2}\nabla f(0)$. Then suppose we know d^{r-1}, by combining (3.15) and (3.14) we have

$$d^r = d^{r-1} + (x^r - x^{r-1}) - \Upsilon^{-2}(\nabla f(x^r) - \nabla f(x^{r-1})) - \Upsilon^{-2}F^T\Sigma^2 F x^r.$$

Therefore each d_i^r can be updated as

$$d_i^r = d_i^{r-1} + (x_i^r - x_i^{r-1}) - \frac{1}{M\beta_i^2}(\nabla f_i(x_i^r) - \nabla f_i(x_i^{r-1})) + \sum_{j:j\sim i}\frac{\sigma_{ij}^2}{\beta_i^2}(x_i^r - x_j^r). \tag{3.24}$$

Combining the above expression with the expression in (3.11) for computing Rd^r, it is clear that all the computation only involves in local communication and local gradient computation.

These observations also suggest that for a general choice of parameter matrix $\Sigma^2 \succ 0$, both D-GPDA and xFILTER are in class $\mathcal{A}$. Further, if Σ^2 is a multiple of identity matrix (i.e., there exists $\sigma^2 > 0$ such that $\Sigma^2 = \sigma^2 I_E$), then the computations in (3.23) and (3.24) only involve the sum of neighboring iterates, therefore both algorithms belong to class $\mathcal{A}'$ as well.

3.3 The Convergence Rate Analysis

In this section we provide the analysis steps of the convergence rate of the D-GPDA and xFILTER. All the proofs of the results can be found in the appendix. Note that we use the primal-dual representation discussed in Section 3.2.3 for xFILTER, so that it can be analyzed together with the D-GPDA.

Step 1. We first analyze the dynamics of the dual variable.

Lemma 3.3.1. *Suppose that $f(x)$ is in class $\mathcal{P}_L^M$. Then for all $r \geq 0$, the iterates of D-GPDA satisfy*

$$\|\lambda^{r+1} - \lambda^r\|_{\Sigma^{-2}}^2 \leq 2\kappa \left(\frac{\|\Upsilon^{-1} L(x^r - x^{r-1})\|^2}{M^2} + \|w^{r+1}\|_H^2 \right). \tag{3.25}$$

Further, for all $r \geq 0$, the iterates of xFILTER satisfy

$$\|\lambda^{r+1} - \lambda^r\|_{\Sigma^{-2}}^2 \leq \widetilde{\kappa} \left(\frac{3}{M^2} \|\Upsilon^{-1} L(x^r - x^{r-1})\|^2 + 3\|w^{r+1}\|_{\Upsilon^2}^2 + 3\|\Upsilon R(\epsilon^{r+1} - \epsilon^r)\|^2 \right). \tag{3.26}$$

In the above we have defined the following

$$\kappa := \frac{1}{\underline{\lambda}_{\min}(\Sigma F H^{-1} F^T \Sigma)},\ \widetilde{\kappa} := \frac{1}{\underline{\lambda}_{\min}(\Sigma F \Upsilon^{-2} F^T \Sigma)} = \frac{1}{\underline{\lambda}_{\min}(\mathcal{L}_G)} \tag{3.27a}$$

$$w^{r+1} := (x^{r+1} - x^r) - (x^r - x^{r-1}). \tag{3.27b}$$

Step 2. In this step we analyze the dynamics of the AL.

Lemma 3.3.2. *Suppose that $f(x)$ is in class $\mathcal{P}_L^M$. Then for all $r \geq 0$, the iterates of D-GPDA satisfy*

$$\begin{aligned}\mathsf{AL}^{r+1} - \mathsf{AL}^r \leq & -\frac{1}{2}\|x^{r+1} - x^r\|^2_{\Delta+2\Upsilon^2-L/M} \\ & + \kappa\left(\frac{2}{M^2}\|\Upsilon^{-1}L(x^r - x^{r-1})\|^2 + 2\|w^{r+1}\|_H^2\right).\end{aligned} \tag{3.28}$$

Further, for all $r \geq 0$, the iterates of xFILTER satisfy

$$\begin{aligned}\mathsf{AL}^{r+1} - \mathsf{AL}^r \leq & -\frac{1}{2}\|x^{r+1} - x^r\|^2_{\Upsilon^2 R - \frac{L}{M}} + \langle \Upsilon^2 R\epsilon^{r+1}, x^{r+1} - x^r\rangle \\ & + \widetilde{\kappa}\left(\frac{3}{M^2}\|\Upsilon^{-1}L(x^r - x^{r-1})\|^2 + 3\|w^{r+1}\|^2_{\Upsilon^2} + 3\|\Upsilon R(\epsilon^{r+1} - \epsilon^r)\|^2\right).\end{aligned} \tag{3.29}$$

Before moving forward, we provide bounds for the important parameters κ and $\tilde{\kappa}$. From (3.27a) we can express κ as

$$\begin{aligned}\kappa =& \frac{1}{\underline{\lambda}_{\min}(\Sigma F\Upsilon^{-1}\left(\Upsilon^{-1}B^T\Sigma^2B\Upsilon^{-1} + I\right)^{-1}\Upsilon^{-1}F^T\Sigma)} \\ =& \frac{1}{\underline{\lambda}_{\min}(\left(\Upsilon^{-1}B^T\Sigma^2B\Upsilon^{-1} + I\right)^{-1}\mathcal{L}_G)} \\ \overset{(2.23)}{=}& \frac{1}{\underline{\lambda}_{\min}\left((-\mathcal{L}_G + 2\Upsilon^{-1}\Delta\Upsilon^{-1} + I)^{-1}\mathcal{L}_G\right)}.\end{aligned} \tag{3.30}$$

Similar derivation applies for $\tilde{\kappa}$. In summary we have

$$\kappa \leq \frac{\lambda_{\max}(2\Upsilon^{-1}\Delta\Upsilon^{-1} + I)}{\underline{\lambda}_{\min}(\mathcal{L}_G)}, \quad \widetilde{\kappa} = \frac{1}{\underline{\lambda}_{\min}(\mathcal{L}_G)}. \tag{3.31}$$

Step 3. In this step, we analyze the error sequences $\{\epsilon^{r+1}\}$ generated by the xFILTER. First we have the following well-known result on the behavior of the Chebyshev iteration; see, e.g., [61, Chapter 6] and [62, Theorem 1, Chapter 7].

Lemma 3.3.3. *Consider using the Chebyshev iteration* (3.13) *to solve $Rx = d^r$. Define $x_*^{r+1} = R^{-1}d^r$, with*

$$R := \Upsilon^{-2}(F^T\Sigma^2F + \Upsilon^2). \tag{3.32}$$

Define the following constants:

$$\xi(R) := \frac{\lambda_{\min}(R)}{\lambda_{\max}(R)} \leq 1,\ \xi(\Upsilon^2) := \frac{\lambda_{\min}(\Upsilon^2)}{\lambda_{\max}(\Upsilon^2)} \leq 1,\ \theta(R) := \lambda_{\min}(R) + \lambda_{\max}(R). \quad (3.33)$$

Choose the following parameters:

$$\tau = \frac{2}{\theta(R)},\ \alpha_1 = 2,\ \alpha_{t+1} = \frac{4}{4 - \rho_0^2 \alpha_t},\ \rho_0 = \frac{1 - \xi(R)}{1 + \xi(R)}.$$

Then for any $\eta \in (0, 1)$, in order to achieve the following accuracy

$$\|u_Q - x_*^{r+1}\|_{\Upsilon^2}^2 \leq \eta \|u_0 - x_*^{r+1}\|_{\Upsilon^2}^2, \quad (3.34)$$

it requires the following number of iterations

$$Q \geq -\frac{1}{4} \ln(\eta/4) \sqrt{1/\xi(R)}.$$

Recall that in Algorithm 2 the initial and final solutions for the Chebyshev iteration are assigned to x^r and x^{r+1}, respectively. Define $\widetilde{\epsilon}^r := u_0 - x_*^{r+1}$, we have

$$Rx^r = Ru_0 = R(u_0 - x_*^{r+1}) + Rx_*^{r+1} := R\widetilde{\epsilon}^r + d^r, \forall\ r \geq -1.$$

Plugging in the definition of d^r in (3.17), we obtain

$$R\widetilde{\epsilon}^r = Rx^r + \Upsilon^{-2}(\nabla f(x^r) + F^T \lambda^r - \Upsilon^2 x^r). \quad (3.35)$$

Using the definition of ϵ^{r+1} in (3.18b), and the fact that R is invertible, we obtain the following key relationship

$$\epsilon^{r+1} - \widetilde{\epsilon}^r = x^{r+1} - x^r,\ \forall\ r \geq -1. \quad (3.36)$$

Recall that $\epsilon^{r+1} := x^{r+1} - x_*^{r+1}$, and $x^{r+1} = u_Q$, $x^r = u_0$, then (3.34) implies

$$\|\epsilon^{r+1}\|_{\Upsilon^2}^2 \leq \eta \|\widetilde{\epsilon}^r\|_{\Upsilon^2}^2. \quad (3.37)$$

By combining Lemma 3.3.3, (3.36) and (3.37), the following result provides some essential relationships between the error sequences $\{\epsilon^{r+1}\}$ incurred by running finite number of C-iterations, with the outer-loop iterations $\{x^{r+1}\}$.

Lemma 3.3.4. *Choose the inner iteration of xFILTER as*

$$Q = -\frac{1}{4}\ln\left(\frac{\theta^2}{16 + 128M\max\{\lambda_{\max}(\Upsilon^2 R), 1\}}\right)\sqrt{1/\xi(R)}. \tag{3.38}$$

where $\theta := \xi(\Upsilon^2 R)\xi(\Upsilon^2) \times \min\{1, \lambda_{\min}(\Upsilon^2)\}$. *Then we have the following inequalities*

$$\|\Upsilon^2 R\epsilon^{r+1}\|^2 \leq \frac{1}{16M}\|x^{r+1} - x^r\|^2_{\Upsilon^2 R}, \tag{3.39a}$$

$$\|\epsilon^{r+1}\|^2_{\Upsilon^2 R} \leq \frac{1}{16M}\|x^{r+1} - x^r\|^2_{\Upsilon^2 R}, \tag{3.39b}$$

$$\|\Upsilon R\epsilon^{r+1}\|^2 \leq \frac{1}{16M}\|x^{r+1} - x^r\|^2_{\Upsilon^2 R}, \tag{3.39c}$$

$$\langle \Upsilon^2 R\epsilon^{r+1}, x^{r+1} - x^r\rangle \leq \frac{3}{16}\|x^{r+1} - x^r\|^2_{\Upsilon^2 R}, \tag{3.39d}$$

$$\langle \Upsilon^2 R\epsilon^{r}, x^{r+1} - x^r\rangle \leq \frac{1}{8}\|x^{r} - x^{r-1}\|^2_{\Upsilon^2 R} + \frac{1}{16}\|x^{r+1} - x^r\|^2_{\Upsilon^2 R}. \tag{3.39e}$$

3.3.1 Proof of Lemma 3.3.4

Proof. Let us choose

$$\eta = \theta^2/(4 + 32M\max\{\lambda_{\max}(\Upsilon^2 R), 1\}). \tag{3.40}$$

Then from Lemma 3.3.3, it is clear that if Q satisfies (3.38), then

$$\|\epsilon^{r+1}\|^2_{\Upsilon^2} \leq \eta\|\tilde{\epsilon}^r\|^2_{\Upsilon^2}. \tag{3.41}$$

Note that $\Upsilon^2 R = F^\top \Sigma^2 F + \Upsilon^2 \succ 0$, then it follows that

$$\begin{aligned}
\|\Upsilon^2 R\epsilon^{r+1}\|^2 &\leq \frac{\lambda_{\max}(R\Upsilon^2\Upsilon^2 R)}{\lambda_{\min}(\Upsilon^2)}\|\epsilon^{r+1}\|^2_{\Upsilon^2} \\
&\overset{(3.37)}{\leq} \frac{\eta\lambda_{\max}(R\Upsilon^2\Upsilon^2 R)}{\lambda_{\min}(\Upsilon^2)}\|\tilde{\epsilon}^r\|^2_{\Upsilon^2} \leq \frac{\eta\lambda_{\max}(R\Upsilon^2\Upsilon^2 R)\lambda_{\max}(\Upsilon^2)}{\lambda_{\min}(\Upsilon^2)}\|\tilde{\epsilon}^r\|^2 \\
&\leq \frac{\eta\lambda_{\max}(R\Upsilon^2\Upsilon^2 R)\lambda_{\max}(\Upsilon^2)}{\lambda_{\min}(R\Upsilon^2\Upsilon^2 R)\lambda_{\min}(\Upsilon^2)}\|\Upsilon^2 R\tilde{\epsilon}^r\|^2 \leq \eta\theta^{-2}\|\Upsilon^2 R\tilde{\epsilon}^r\|^2.
\end{aligned}$$

Using the above relation, we can then obtain the following

$$\begin{aligned}\|\Upsilon^2 R\epsilon^{r+1}\|^2 &\leq 2\eta\theta^{-2}(\|\Upsilon^2 R\epsilon^{r+1}\|^2 + \|\Upsilon^2 R(\epsilon^{r+1} - \tilde{\epsilon}^r)\|^2) \\ &\overset{(3.36)}{\leq} 2\eta\theta^{-2}(\|\Upsilon^2 R\epsilon^{r+1}\|^2 + \|\Upsilon^2 R(x^{r+1} - x^r)\|^2).\end{aligned}$$

Therefore, we obtain

$$\|\Upsilon^2 R\epsilon^{r+1}\|^2 \leq 2\eta\theta^{-2}/(1 - 2\eta\theta^{-2})\|\Upsilon^2 R(x^{r+1} - x^r)\|^2.$$

Plugging the definition of η in (3.40), we have

$$\begin{aligned}\|\Upsilon^2 R\epsilon^{r+1}\|^2 &\leq \lambda_{\max}(\Upsilon^2 R)2\eta\theta^{-2}/(1 - 2\eta\theta^{-2})\|x^{r+1} - x^r\|^2_{\Upsilon^2 R} \\ &\overset{(3.40)}{\leq} 1/(16M)\|x^{r+1} - x^r\|^2_{\Upsilon^2 R}, \quad \forall\ r \geq -1.\end{aligned}$$

To obtain the second inequality, notice that

$$\|\epsilon^{r+1}\|^2_{\Upsilon^2 R} \leq \theta^{-1}\eta\|\tilde{\epsilon}^r\|^2_{\Upsilon^2 R} \leq \theta^{-2}\eta\|\tilde{\epsilon}^r\|^2_{\Upsilon^2 R} \tag{3.42}$$

where the last inequality is due to the fact that $\theta \leq 1$. Then repeating the above derivation we can obtain the desired result. The third inequality in (3.39) can be derived in a similar way, and the last two in (3.39) can be obtained by using Cauchy-Swartz inequality. **Q.E.D.**

Clearly, using the Chebyshev iteration is one critical step that ensures fast reduction of the error $\{\epsilon^{r+1}\}$. In particular, to achieve constant reduction of error, the total number of required Chebyshev iteration is proportional to $\sqrt{1/\xi(R)}$, rather than $1/\xi(R)$ in conventional iterative scheme such as the Richardson's iteration [61]. Such a choice enables the final bound to be dependent on $\sqrt{1/\xi(\mathcal{G})}$, rather than $1/\xi(\mathcal{G})$.

Step 4. Let us construct the following potential functions (parameterized by constants

$c, \widetilde{c} > 0$)

$$P_c(x^{r+1}, x^r, \lambda^{r+1}) := \mathsf{AL}^{r+1} + \frac{2\kappa}{M^2}\|\Upsilon^{-1}L(x^{r+1} - x^r)\|^2$$
$$+ \frac{c}{2}\left(\|\Sigma F x^{r+1}\|^2 + \|x^{r+1} - x^r\|^2_{H+L/M}\right). \tag{3.43a}$$
$$\widetilde{P}_{\widetilde{c}}(x^{r+1}, x^r, \lambda^{r+1}) := \mathsf{AL}^{r+1} + \frac{3\widetilde{\kappa}}{M^2}\|\Upsilon^{-1}L(x^{r+1} - x^r)\|^2 \tag{3.43b}$$
$$+ \frac{3\widetilde{\kappa}}{8}\|x^{r+1} - x^r\|^2_{\Upsilon^2 R} + \frac{\widetilde{c}}{2}\left(\|\Sigma F x^{r+1}\|^2 + \|x^{r+1} - x^r\|^2_{\Upsilon^2 + \frac{\Upsilon^2 R}{4} + \frac{L}{M}}\right).$$

For notational simplicity we will denote them as P^{r+1} and $\widetilde{P}^{r+1}$, respectively. In the following we show that when the algorithm parameters are chosen properly, the potential functions will decrease along the iterations.

Lemma 3.3.5. *Suppose that $f(x)$ is in class $\mathcal{P}_L^M$, and that the parameters of D-GPDA are chosen as below*

$$c = \max\{6\kappa, 1\},\ \Upsilon^2 \succeq \frac{L\Upsilon^{-2}L}{M^2}, \tag{3.44a}$$
$$\frac{1}{2}\left(\Delta + \Upsilon^2\right) - \frac{L}{M} - \frac{4\kappa}{M^2}L\Upsilon^{-2}L - \frac{2cL}{M} \succeq 0. \tag{3.44b}$$

Then for all $r \geq 0$, we have

$$P^r - P^{r+1} \geq \frac{1}{4}\|x^{r+1} - x^r\|^2_{\Delta + \Upsilon^2} + \kappa\|w^{r+1}\|^2_H. \tag{3.45}$$

Lemma 3.3.6. *Suppose that $f(x)$ is in class $\mathcal{P}_L^M$, Q is chosen according to* (3.38)*, and the rest of the parameters of xFILTER are chosen as below*

$$\tilde{c} = 8\tilde{\kappa} = \frac{8}{\underline{\lambda}_{\min}(\Sigma F \Upsilon^{-2} F^T \Sigma)},\ \Upsilon^2 \succeq \frac{L\Upsilon^{-2}L}{M^2}, \tag{3.46a}$$
$$(1/4 - 3\tilde{\kappa} - \tilde{c})\Upsilon^2 R - (1 + 2\tilde{c})L/M - \frac{6\tilde{\kappa}}{M^2}L\Upsilon^{-2}L \succeq 0. \tag{3.46b}$$

Then for all $r \geq 0$, we have

$$\widetilde{P}^r - \widetilde{P}^{r+1} \geq \frac{1}{8}\|x^{r+1} - x^r\|^2_{\Upsilon^2 R} + \tilde{\kappa}\|w^{r+1}\|^2_{\Upsilon^2}. \tag{3.47}$$

Step 5. Next we show the lower and upper boundedness of the potential function.

Lemma 3.3.7. *Suppose that $f(x)$ is in class $\mathcal{P}_L^M$ and the parameters are chosen according to* (3.44)*. Then the iterates generated by D-GPDA satisfy*

$$P^{r+1} \geq \underline{f} > -\infty, \quad \forall\ r > 0, \tag{3.48a}$$

$$P^0 \leq f(x^0) + \frac{2}{M} d_0^T L^{-1} d_0, \tag{3.48b}$$

where d_0 is defined in (2.48)*.*

Similarly, for xFILTER the function $\widetilde{P}^{r+1}$ has the same expression as in (3.48a)*, and*

$$\widetilde{P}^0 \leq f(x^0) + \frac{5}{M} d_0^T L^{-1} d_0. \tag{3.49}$$

Step 6. We are ready to derive the final bounds for the convergence rate of the proposed algorithms.

Theorem 3.3.1. *Suppose that $f(x)$ is in class $\mathcal{P}_L^M$ and the parameters are chosen according to* (3.44)*. Let T denote an iteration index in which D-GPDA satisfies*

$$e(T) := \min_{r \in [T]} \left\| 1/M \sum_{i=1}^{M} \nabla f_i(x_i^r) \right\|^2 + \|\Sigma F x^r\|^2 \leq \epsilon. \tag{3.50}$$

Then we have the following bound for the error:

$$\epsilon \leq C_1 \times \frac{C_2}{T}, \text{ with } C_1 := 8\left(f(x^0) - \underline{f} + \frac{2}{M} d_0^T L^{-1} d_0 \right)$$
$$C_2 := 4 \sum_{(i,j):i\sim j} \sigma_{ij}^2 + \sum_{i=1}^{M} \beta_i^2 + 4. \tag{3.51}$$

Similarly, for xFILTER when the parameters are chosen according to (3.46) *and* (3.38)*, the same equation*

$$\epsilon \leq \widetilde{C}_1 \times \frac{\widetilde{C}_2}{T_r} \tag{3.52}$$

holds true (with T_r denoting the total number of outer iterations), with the following

constants

$$\widetilde{C}_1 := f(x^0) - \underline{f} + \frac{5}{M} d_0^T L^{-1} d_0 \tag{3.53a}$$

$$\widetilde{C}_2 := 128 \left(\sum_{i=1}^{M} \beta_i^2 + 3 + \frac{1}{32\widetilde{\kappa}} \right). \tag{3.53b}$$

We note that one key difference between the two rates is that, the constant C_2 for D-GPDA depends explicitly on σ_e's, while its counterpart for xFILTER depends on $1/\widetilde{\kappa}$ instead. Further, for xFILTER, the constant $\widetilde{\kappa}$ in (3.31) only depends on $\underline{\lambda}_{\min}(\mathcal{L}_G)$, while for D-GPDA κ is further dependent on $\lambda_{\max}(\Upsilon^{-1}\Delta\Upsilon^{-1})$. These properties will be leveraged later when choosing algorithm parameters to ensure that optimal rates for different problems and networks are obtained.

3.4 Rate Bounds and Tightness

In this section we provide explicit choices of various parameters, and discuss the tightness of the resulting bounds for D-GPDA and xFILTER.

3.4.1 Parameter Selection and Rate Bounds for D-GPDA

Let us pick the following parameters for D-GPDA

$$\sigma_{ij}^2 = \frac{\beta^2 \sqrt{L_i L_j}}{\sqrt{d_i d_j}},\ \Upsilon^2 = \beta^2 L,\ \beta^2 = \frac{80 \max\{\lambda_{\max}(W), 1\}}{\min\{\underline{\lambda}_{\min}(\mathcal{L}_G), 1\} M}. \tag{3.54}$$

It follows that the following relations hold

$$\Delta = \beta^2 W, \quad \beta_i^2 = \beta^2 L_i,\ \forall\ i, \quad \kappa \overset{(3.31)}{\leq} \frac{1 + 2\lambda_{\max}(W)}{\min\{\underline{\lambda}_{\min}(\widetilde{L}), 1\}}. \tag{3.55}$$

In the above definitions, we have defined $W \in \mathbb{R}^M$ as a diagonal matrix with

$$[W]_{ii} = \frac{\sqrt{L_i}}{\sqrt{d_i}} \sum_{q:q\sim i} \frac{\sqrt{L_q}}{\sqrt{d_q}},$$

and that

$$[\mathcal{L}_G]_{ij} = \begin{cases} \sum_{q:q\sim i} \frac{1}{\sqrt{d_q d_i}} & \text{if } i = j \\ -\frac{1}{\sqrt{d_i d_j}} & \text{if } (ij) \in \mathcal{E}, i \neq j \\ 0 & \text{otherwise.} \end{cases} \tag{3.56}$$

Note that when $d_i = d_j$, $\forall\ i, j$, we have $\mathcal{L}_G = \mathcal{L}$. We have the following result.

Theorem 3.4.1. *Consider using D-GPDA to solve problems in class* $(\mathcal{P}_L^M, \mathcal{N}_D^M)$*, using parameters in* (3.54)*. Then the condition* (3.44b) *will be satisfied. Further, to achieve* $e(T) \leq \epsilon$*, it requires at most the following number of iterations*

$$T \leq \frac{8}{\epsilon}\left(f(x^0) - \underline{f} + \frac{2}{M}\|d_0\|_{L^{-1}}^2\right) \times C_2 \tag{3.57}$$

where C_2 *is given by [with* W *and* $\widetilde{\mathcal{L}}$ *defined in* (3.56)*]*

$$C_2 \leq \frac{320\max\{\lambda_{\max}(W), 1\}}{\min\{\underline{\lambda}_{\min}(\mathcal{L}_G), 1\}} \sum_{(i,j):i\sim j}\left(\frac{\sqrt{L_i L_j}}{\sqrt{d_i d_j}M} + \frac{\bar{L}}{4}\right) + 4. \tag{3.58}$$

Proof. For D-GPDA, use the parameters in (3.54), we have

$$c \leq \frac{6 + 12\lambda_{\max}(W)}{\min\{\underline{\lambda}_{\min}(\mathcal{L}_G), 1\}}, \quad \Upsilon^2 = \frac{80\max\{\lambda_{\max}(W), 1\}}{\min\{\underline{\lambda}_{\min}(\mathcal{L}_G), 1\}M}L.$$

Therefore to ensure condition (3.44b), it suffices to ensure the following

$$\frac{40\max\{\lambda_{\max}(W), 1\}}{\min\{\underline{\lambda}_{\min}(\mathcal{L}_G), 1\}M}L - \frac{(4 + 8\lambda_{\max}(W))}{M80\max\{\lambda_{\max}(W), 1\}}L - \frac{1}{M}L - \frac{6 + 12\lambda_{\max}(W)}{\min\{\underline{\lambda}_{\min}(\mathcal{L}_G), 1\}}\frac{2}{M}L \succ 0. \tag{3.59}$$

It is easy to check that this inequality will be satisfied using the above choice of parameters. Using these choices, we can obtain the desired expression for C_2. **Q.E.D.**

3.4.2 Parameter Selection and Rate Bounds for xFILTER

First, recall that we have defined the matrix $\widetilde{L}$ and $\widehat{\mathcal{L}}$ as follows [see the definition in (2.25)]

$$\widetilde{\mathcal{L}} = L^{-1/2}P^{-1/2}F^T KFP^{-1/2}L^{-1/2},$$
$$\widehat{\mathcal{L}} = L^{-1/2}F^T KFL^{-1/2}.$$

Below we will provide two different choices of parameters.

Choice I. We will focus on a class of graphs such that there exists an *absolute* constant $k > 0$ such that the following holds (i.e., the degrees of the nodes are not quite different from their averages):

$$kP \succeq \bar{d}I_M. \tag{3.60}$$

The above condition says that the degrees of the nodes are not quite different from their averages. For example the following graphs satisfy (3.60): Complete graph ($k = 1$), star graph ($k = 2$), grid graph ($k = 2$), cubic graph ($k = 1$), path graph ($k = 2$), and any regular graph ($k = 1$).

For the class of graphs satisfy (3.60), let us pick the parameters for xFILTER as follows

$$\Sigma^2 = \frac{48 \times 96k}{\sum_i d_i \underline{\lambda}_{\min}(\widetilde{\mathcal{L}})}K, \quad \Upsilon^2 = \frac{96k}{\sum_i d_i}P^{1/2}LP^{1/2}. \tag{3.61}$$

Using the above choice, we have

$$\beta_i^2 = \frac{96L_i d_i k}{\sum_i d_i} \tag{3.62}$$

and that the matrix Υ satisfies the following

$$\Upsilon^2 = \frac{96k}{\sum_i d_i}P^{1/2}LP^{1/2} \succeq \frac{96}{M}L. \tag{3.63}$$

Plugging these choices to the generalized Laplacian $\mathcal{L}_G$ in (2.23) we obtain

$$\begin{aligned}\mathcal{L}_G &= \Upsilon^{-1}F^T\Sigma^2F\Upsilon^{-1} \\ &= \frac{48}{\underline{\lambda}_{\min}(\widetilde{\mathcal{L}})}L^{-1/2}P^{-1/2}F^TKFP^{-1/2}L^{-1/2} = \frac{48}{\underline{\lambda}_{\min}(\widetilde{\mathcal{L}})}\widetilde{\mathcal{L}}.\end{aligned} \tag{3.64}$$

Therefore by (3.31) we have

$$\tilde{\kappa} \ = \frac{\underline{\lambda}_{\min}(\widetilde{\mathcal{L}})}{48\underline{\lambda}_{\min}(\widetilde{\mathcal{L}})} = \frac{1}{48}. \tag{3.65}$$

Also in this case we have

$$R = \Upsilon^{-2}F^T\Sigma^2F + I = \frac{48}{\underline{\lambda}_{\min}(\widetilde{\mathcal{L}})}P^{-1/2}L^{-1}P^{-1/2}F^TKF + I.$$

By noting that the matrix $P^{-1/2}L^{-1}P^{-1/2}F^TKF$ and $\widetilde{\mathcal{L}}$ has the same set of eigenvalues, we obtain

$$\lambda_{\max}(R) \leq \left(\frac{48\lambda_{\max}(\widetilde{\mathcal{L}})}{\underline{\lambda}_{\min}(\widetilde{\mathcal{L}})} + 1\right) \leq \frac{50}{\xi(\widetilde{\mathcal{L}})}, \ \lambda_{\min}(R) = 1, \tag{3.66a}$$

$$\xi(R) \geq 1/\left(\frac{48\lambda_{\max}(\widetilde{\mathcal{L}})}{\underline{\lambda}_{\min}(\widetilde{\mathcal{L}})} + 1\right) \geq \frac{\xi(\widetilde{\mathcal{L}})}{50}. \tag{3.66b}$$

Choice II. For general graphs not necessarily satisfying (3.60), let us pick the parameters for xFILTER as follows

$$\Sigma^2 = \frac{48 \times 96}{M\underline{\lambda}_{\min}(\widehat{\mathcal{L}})}K, \quad \Upsilon^2 = \frac{96}{M}L. \tag{3.67}$$

Using the above choice, we have

$$\beta_i^2 = \frac{96L_i}{M}. \tag{3.68}$$

We have that

$$\mathcal{L}_G = \Upsilon^{-1}F^T\Sigma^2F\Upsilon^{-1} = \frac{48}{\underline{\lambda}_{\min}(\widehat{\mathcal{L}})}L^{-1/2}F^TKFL^{-1/2} = \frac{48}{\underline{\lambda}_{\min}(\widehat{\mathcal{L}})}\widehat{\mathcal{L}}. \tag{3.69}$$

Therefore by (3.31) we have

$$\tilde{\kappa} \ = \frac{\underline{\lambda}_{\min}(\widehat{\mathcal{L}})}{48\underline{\lambda}_{\min}(\widehat{\mathcal{L}})} = \frac{1}{48}. \tag{3.70}$$

Also in this case we have

$$R = \Upsilon^{-2}F^T\Sigma^2 F + I = \frac{48}{\underline{\lambda}_{\min}(\widehat{\mathcal{L}})}L^{-1}F^T K F + I.$$

By noting that the matrix $L^{-1}F^T K F$ and $\widehat{\mathcal{L}}$ has the same set of eigenvalues, we obtain

$$\lambda_{\max}(R) \leq \left(\frac{48\lambda_{\max}(\widehat{\mathcal{L}})}{\underline{\lambda}_{\min}(\widehat{\mathcal{L}})} + 1\right) \leq \frac{50}{\xi(\widehat{\mathcal{L}})}, \ \lambda_{\min}(R) = 1, \tag{3.71a}$$

$$\xi(R) \geq 1/\left(\frac{48\lambda_{\max}(\widehat{\mathcal{L}})}{\underline{\lambda}_{\min}(\widehat{\mathcal{L}})} + 1\right) \geq \frac{\xi(\widehat{\mathcal{L}})}{50}. \tag{3.71b}$$

Remark 3.4.1. **(Choices of Parameters)** The main difference between the above two choices of parameters is whether Υ^2 is scaled with the degree matrix or not. The resulting bounds are also dependent on the spectral gap for $\widetilde{L}$ and $\widehat{\mathcal{L}}$, one inversely scaled with the degree matrix, and the other does not. Note that the spectral gap of $\widetilde{L}$ and $\widehat{\mathcal{L}}$ may not be the same. For example for a star graph with $L_i = L_j$, $\xi(\widehat{\mathcal{L}}) = \mathcal{O}(1/M)$ but $\xi(\widetilde{\mathcal{L}}) = \mathcal{O}(1)$. Therefore one has to be careful in choosing these parameters so that $\xi(R)$ is made as large as possible.

Additionally, since we are mainly interested in choosing the optimal parameters so that the resulting rate bounds will be optimal in their dependency on problem parameters, the absolute constants in the above parameter choices have not been optimized.

The following result is a direct consequence of the second part of Theorem 3.3.1.

Theorem 3.4.2. *Consider using xFILTER to solve problems in class* $(\mathcal{P}_L^M, \mathcal{N}_D^M)$*, then the following holds.*

Case I. *Further restricting* $\mathcal{N}_D^M$ *to a subclass satisfying* (3.60)*. If parameters in* (3.61) *is used, then the condition* (3.46b) *will be satisfied. Further, to achieve* $e(T) \leq \epsilon$*, it requires at most the following number of iterations (where* T *denotes the total iterations of the xFILTER algorithm)*

$$
\begin{aligned}
T &\leq \frac{1}{\epsilon}\left(f(x^0) - \underline{f} + \frac{5}{M}\|d_0\|^2_{L^{-1}}\right) \times \widetilde{C}_2 \\
&\quad \times \frac{1}{4}\ln\left(\frac{16 + 128M\max\{\lambda_{\max}(\Upsilon^2 R), 1\}}{\theta^2}\right)\sqrt{1/\xi(R)} \\
&\leq \frac{1}{\epsilon}\left(f(x^0) - \underline{f} + \frac{5}{M}\|d_0\|^2_{L^{-1}}\right) \times \widetilde{C}_2 \\
&\quad \times \frac{1}{4}\ln\left(\frac{50^2(ML_{\max}/L_{\min})^4 \times (16 + 128M\max\{50 \times 96kL_{\max}, 1\})}{\xi^3(\widetilde{\mathcal{L}}) \times \min\{1, 96^2k^2L^2_{\min}/M^2\}}\right)\sqrt{50/\xi(\widetilde{\mathcal{L}})}
\end{aligned}
\tag{3.72}
$$

where $\widetilde{C}_2$ *is given by*

$$
\widetilde{C}_2 \leq 128\left(\frac{96k}{\sum_{i=1}^M d_i}\sum_{i=1}^M d_iL_i + 19\right). \tag{3.73}
$$

Case II. *Suppose parameters in* (3.67) *are used. Then the condition* (3.46b) *will be satisfied. Further, to achieve* $e(T) \leq \epsilon$*, it requires at most the following number of iterations*

$$
\begin{aligned}
T &\leq \frac{1}{\epsilon}\left(f(x^0) - \underline{f} + \frac{5}{M}\|d_0\|^2_{L^{-1}}\right) \times \widetilde{C}_2 \\
&\quad \times \frac{1}{4}\ln\left(\frac{50^2(L_{\max}/L_{\min})^4 \times (16 + 128M\max\{50 \times 96L_{\max}/M, 1\})}{\xi^3(\widetilde{\mathcal{L}}) \times \min\{1, 96^2L^2_{\min}/M^2\}}\right)\sqrt{50/\xi(\widetilde{\mathcal{L}})}
\end{aligned}
\tag{3.74}
$$

where $\widetilde{C}_2$ *is given by*

$$
\widetilde{C}_2 \leq 128\left(\frac{96}{M}\sum_{i=1}^M L_i + 19\right). \tag{3.75}
$$

We note that compared with the results in Theorem 3.3.1, the additional multiplicative term in (3.72) accounts for the Chebyshev iterations that are needed for every iteration t. It is interesting to observe that comparing with the previous result, the constant $\widetilde{C}_2$ in (3.73) is independent on any graph parameters. Such a desirable property

turns out to be crucial for obtaining tight rate bounds.

3.4.3 Tightness of the Upper Rate Bounds

In this section, we present some tightness results of the upper rate bounds for our proposed D-GPDA and xFILTER. In particular, we compare the expressions derived in Theorem 3.4.1 – 3.4.2, and the lower bounds derived in Section 2.3, over different kinds of graphs and for different problems. We will mainly focus on the case with uniform Lipschitz constants, i.e., $L_i = U,\ \forall\ i$. WE will briefly discuss the case of non-uniform Lipschitz constants at the end of this section.

First, we consider the problem class $\mathcal{P}_U^M$ with the following properties:

$$L_1 = L_2 = \cdots L_M = \frac{1}{M}\sum_{i=1}^{M} L_i := U, \quad L = UI_M. \tag{3.76}$$

It follows that in this case $\widetilde{\mathcal{L}} = \mathcal{L}$, and $\widehat{\mathcal{L}} = P^{1/2}\mathcal{L}P^{1/2}$. Let us first make some useful observations. *Remark 3.4.2.* Let us specialize the parameter choices for D-GPDA algorithm in (3.54) and derive the bounds for C_2 in (3.58) for two special graphs.
Complete graph. For complete graphs we have $d_i = d_j = M-1, \forall\ i,j$, which implies that $\mathcal{L}_G = \mathcal{L}$, so $\underline{\lambda}_{\min}(\mathcal{L}_G) = M/(M-1)$. Because $L_i = L_i = U,\ \forall\ i,j$, we have $W = I_M$. Therefore using the expression (3.58) we obtain the following:

$$C_2^{\text{comp}} \leq 400U + 4. \tag{3.77}$$

Cycle graph. For cycle graph we have $d_i = d_j = 2, \forall\ i,j$, which implies that $\mathcal{L}_G = \mathcal{L}$, and $\underline{\lambda}_{\min}(\mathcal{L}_G) \geq 1/M^2$. Because $L_i = L_i = U,\ \forall\ i,j$, we have $W = I_M$. Therefore using the expression (3.58) we obtain the following:

$$C_2^{\text{cycle}} \leq 240UM^2 + 4. \tag{3.78}$$

It is clear that for cycle graph whose diameter is in $\mathcal{O}(M)$, the rate bounds is very large.

Remark 3.4.3. Let us specialize the parameter choices for xFILTER algorithm in (3.61) and derive the bounds for $\widetilde{C}_2 \times 1/\sqrt{\xi(\widetilde{\mathcal{L}})}$ in (3.73) for the following special graphs. Note that because uniform L_i's are assumed, we have $\widetilde{\mathcal{L}} = \mathcal{L}$.

Complete graph. Complete graphs satisfy (3.60) with $k = 1$. It also satisfies $\underline{\lambda}_{\min}(\widetilde{\mathcal{L}}) = M/(M-1) \geq 1$. Therefore using the expression (3.73) we obtain the following:

$$\widetilde{C}_2^{\text{comp}} \times \frac{1}{\sqrt{\xi(\widetilde{\mathcal{L}})}} \leq 12500U + 2560. \tag{3.79}$$

Grid graph. Grid graphs satisfy (3.60) with $k = 2$. It also satisfies $\underline{\lambda}_{\min}(\widetilde{\mathcal{L}}) \geq 1/M$. Therefore using the expression (3.73) we obtain the following:

$$\widetilde{C}_2^{\text{grid}} \times \frac{1}{\sqrt{\xi(\widetilde{\mathcal{L}})}} \leq (12500U + 2560) \times \sqrt{M}. \tag{3.80}$$

Star graph. Star graphs satisfy (3.60) with $k = 2$. It also has $\xi(\widetilde{\mathcal{L}}) = 1/2$. Therefore using the expression (3.73) we obtain the following:

$$\widetilde{C}_2^{\text{star}} \times \frac{1}{\sqrt{\xi(\widetilde{\mathcal{L}})}} \leq (12500U + 2560) \times \sqrt{2}. \tag{3.81}$$

Geometric graph. For geometric graphs which place the nodes uniformly in $[0,1]^2$ and connect any two nodes separated by a distance less than a radius $R \in (0,1)$. Then if the connectivity radius R satisfies [42]

$$R = \Omega\left(\sqrt{\log^{1+\epsilon}(M)/M}\right), \quad \text{for any } \epsilon > 0, \tag{3.82}$$

then with high probability

$$\xi(\widetilde{\mathcal{L}}) = \mathcal{O}\left(\frac{\log(M)}{M}\right). \tag{3.83}$$

Further, from the proof of [63, Lemma 10], for any ϵ and $c > 0$, if

$$R = \Omega\left(\sqrt{\log^{1+\epsilon}(M)/(M\pi)}\right) \tag{3.84}$$

then with probability at least $1 - 2/M^{c-1}$, the following holds

$$\log^{1+\epsilon} M - \sqrt{2}c \log M \leq d_i \leq \log^{1+\epsilon} M + \sqrt{2}c \log M, \ \forall \ i. \tag{3.85}$$

This means that (3.60) is satisfied (with $k = 1$) with high probability (also see discussion at the end of [42, Section V]). Therefore using the expression (3.58) we obtain the following:

$$\widetilde{C}_2^{\text{geometric}} \times \frac{1}{\sqrt{\xi(\widetilde{\mathcal{L}})}} \leq (12500U + 2560) \times \mathcal{O}\left(\frac{\sqrt{M}}{\sqrt{\log(M)}}\right). \tag{3.86}$$

Cycle/Path graph. Cycle/path graphs satisfy (3.60) with $k = 2$. We also have $\underline{\lambda}_{\min}(\widetilde{\mathcal{L}}) \geq 1/M^2$ (see the discussion in Sec. 2.2.3). Therefore using the expression (3.73) we obtain the following:

$$\widetilde{C}_2^{\text{cycle}} \times \frac{1}{\sqrt{\xi(\widetilde{\mathcal{L}})}} \leq (12500U + 2560) \times M. \tag{3.87}$$

From the above comparison, it is clear that the rate bounds for xFILTER is about $\mathcal{O}(M)$ times better than the D-GPDA for the path/cycle graph.

We also note that for the xFILTER algorithm, the fact that $L_i = U, \ \forall \ i$ implies that the matrix Σ^2 given in (3.61) is a multiple of identity matrix. Therefore by Remark 3.2.3, we can conclude that in this case xFILTER belongs to both $\mathcal{A}$ and $\mathcal{A}'$.

Now we are ready to present our tightness analysis on D-GPDA and xFILTER.

Theorem 3.4.3. *We have the following tightness results.*
(1) *Let $D = 1$ and consider the class $(P_U^M, \mathcal{N}_D^M)$. Then D-GPDA is an optimal algorithm, and its convergence rate in* (3.57) *is tight (up to a universal constant).*
(2) *Let $D = M - 1$ and consider the class $(P_U^M, \mathcal{N}_D^M)$. Then xFILTER is an optimal algorithm, and its convergence rate in* (3.72) *is tight (up to a polylog factor).*
(3) *More generally, consider the problem class P_U^M, and a subclass of $\mathcal{N}_D^M$ satisfying* (3.60)*. Then the convergence rate in* (3.72) *is tight (up to a polylog factor).*

Proof. We divide the proof into different cases.

Case 1). The network class is a complete graph with M nodes. Using the parameters in (3.54), C_2 is given by (3.77), and we have that $\Sigma^2 = \frac{80U}{(M-1)M} I_E$. Note that the following holds

$$\|Fx\|^2 = \sum_{(i,j):i\sim j} \|x_i - x_j\|^2.$$

If (3.50) holds, then Theorem 3.3.1 and Theorem 3.4.1 imply

$$T \leq 8\big(f(0) - \underline{f} + \frac{2}{MU}\|d_0\|^2\big) \times \frac{400U+4}{\epsilon}.$$

For complete graph it is easy to check that $\xi(\mathcal{G}) \geq 1$. Using the definition in (2.18), we also have

$$\begin{aligned} h_T^* &= \min_{r\in[T]} \big\| \frac{1}{M}\sum_{i=1}^M \nabla f_i(x_i^r)\big\|^2 + \frac{U}{M^2}\|Ax^r\|^2 \\ &\leq \min_{r\in[T]} \big\| \frac{1}{M}\sum_{i=1}^M \nabla f_i(x_i^r)\big\|^2 + \frac{1}{80}\|\Sigma A x^r\|^2 \leq e(T) \leq \epsilon. \end{aligned}$$

By comparing the lower bound derived in Lemma 2.3.2, we conclude that the above rate bound is tight (up to some universal constants).

Case 2). The network class is a path graph with $M = D + 1$. From Section 2.2.3 we have

$$\xi(\mathcal{G}) \geq \frac{1}{M^2}. \tag{3.88}$$

Further we note that condition (3.60) satisfies with $k = 2$. We have

$$\widetilde{\mathcal{L}} = P^{-1/2} F^T F P^{-1/2} = \mathcal{L}. \tag{3.89}$$

Therefore we conclude that

$$\xi(\widetilde{\mathcal{L}}) \geq \xi(\mathcal{G}) \geq \frac{1}{M^2}. \tag{3.90}$$

Applying the above estimate to (3.61), we can choose

$$\Sigma^2 = \frac{4608U}{4(M-2)\underline{\lambda}_{\min}(\widetilde{\mathcal{L}})} I_E, \quad \Upsilon^2 = \frac{96U}{4(M-2)} P. \tag{3.91}$$

Using these choices, again we will have

$$\xi(R) \overset{(3.66)}{\geq} \frac{\xi(\widetilde{\mathcal{L}})}{50} \overset{(3.90)}{\geq} \frac{1}{50M^2}. \tag{3.92}$$

Using these constants, and note $D \leq M$, we have

$$\begin{aligned} h^*_{T_r} &= \min_{r\in[T_r]} \Big\| \frac{1}{M} \sum_{i=1}^M \nabla f_i(x_i^r) \Big\|^2 + \frac{U}{M\underline{\lambda}_{\min}(P^{1/2}\mathcal{L}P^{1/2})} \sum_{(i,j):i\sim j} \|x_i - x_j\|^2 \\ &\leq \min_{r\in[T_r]} \Big\| \frac{1}{M} \sum_{i=1}^M \nabla f_i(x_i^r) \Big\|^2 + \frac{U}{\underline{\lambda}_{\min}(\mathcal{L})M} \|Fx^r\|^2 \\ &\overset{(3.91)}{\leq} \min_{r\in[T_r]} \Big\| \frac{1}{M} \sum_{i=1}^M \nabla f_i(x_i^r) \Big\|^2 + \frac{1}{2304} \|\Sigma F x^r\|^2 \leq e(T_r), \end{aligned}$$

where in the first inequality we have used $P \succeq I_M$. Similarly as in the previous case, suppose $e(T_r) \leq \epsilon$, then according to Theorem 3.4.2 we have

$$\epsilon \leq \left(f(0) - \inf_x f(x) + \|d_0\|^2 \frac{5}{MU} I_M \right) \times \frac{128(96U+19)}{T_r}.$$

Recall that for xFILTER, T_r represents the number of times the dual update (3.16) is performed. Between two dual updates Q primal iterations are performed, where the precise number is given in (3.38). According to (3.92) we have

$$\sqrt{1/\xi(R)} \leq 13M. \tag{3.93}$$

Overall, the total number of iterations required is given by

$$\begin{aligned} T \leq \frac{1}{\epsilon} &\left(f(x^0) - \underline{f} + \frac{5}{MU} \|d_0\|^2 \right) \times 128(96U+19) \\ &\times \frac{1}{4} \ln \left(\frac{50^2 M^{10} \times (16 + 128M \max\{50 \times 192U, 1\})}{\min\{1, 96^2 \times 4U^2/M^2\}} \right) \times 13M. \end{aligned} \tag{3.94}$$

This implies that the lower bound obtained in Theorem 2.3.1 is tight up to some universal constant and a ploylog factor in M, and the bound-achieving algorithm in class $\mathcal{A}$ is the xFILTER.

Case 3). The proof follows similar steps are in the previous case. When $L_i = L_j$, $\forall\, i \neq j$, and when (3.60) is satisfied, it is easy to verify that the following holds

$$h^*_{T_r} \leq e(T_r), \text{ and } \widetilde{\mathcal{L}} = \mathcal{L}. \tag{3.95}$$

To bound the total number of iteration required to achieve $h^*_{T_r} \leq \epsilon$, note that when (3.60) is satisfied, we can apply the bound (3.72) in Theorem 3.4.2 and obtain

$$\begin{aligned} T \leq \frac{1}{\epsilon}\left(f(x^0) - \underline{f} + \frac{5}{MU}\|d_0\|^2\right) \times 128\,(96kU + 19) \\ \times \frac{1}{4}\ln\left(\frac{50^2M^4 \times (16 + 128M\max\{50 \times 96kU, 1\})}{\xi^3(\mathcal{G}) \times \min\{1, 96^2k^2U^2/M^2\}}\right)\sqrt{50/\xi(\mathcal{G})}. \end{aligned} \tag{3.96}$$

Comparing with the lower bound obtained in Theorem 2.3.1, it is clear that apart from the multiplicative $\ln(\cdot)$ term, the remaining bound is in the same order as the lower bound given in (2.58). **Q.E.D.**

Remark 3.4.4. **(Optimal Number of Gradient Evaluations)** It is important to note that the "outer" iteration of the xFILTER required to achieve ϵ-local solution scales with $\mathcal{O}(U/\epsilon)$, which is independent of the network size. Because local gradient evaluation is only performed in the outer iterations, the above fact suggests that the total number of gradient evaluation required is also in this order, which is optimal because it is the same as what is needed for the centralized gradient descent.

Remark 3.4.5. **(Performance Gap Between D-GPDA and xFILTER)** If we apply D-GPDA to the path or cycle graph, then according to Remark 3.4.3, the corresponding C_2, as well as the final upper bound, will be in $\mathcal{O}(M^2U)$, which is $\mathcal{O}(M)$ worse than the lower bound. Intuitively, this phenomenon happens because of the following: in order to decompose the entire problem into the individual nodes, the x-update (3.8a) has to create a proximal term that matches the quadratic penalty $\|\Sigma Ax\|^2$. But such an additional term forces the variables to stay close to their previous iteration. In contrast, xFILTER circumvents the above difficulty by leaving the quadratic penalty intact, but instead using a few fast and decomposable iterations to approximately solve

the resulting problem.

Remark 3.4.6. **(An Alternative Bound)** For problems and graphs in $(\mathcal{P}_U^M, \mathcal{N}_D^M)$ without additional conditions, it can be verified that the second choice of the parameters (3.67) gives the following convergence rates [cf. (3.74)]

$$T = \widetilde{\mathcal{O}}\left(\left(f(0) - \inf_x f(x) + \|d_0\|^2 \frac{5}{MU} I_M \right) \times \frac{U}{\epsilon} \times \frac{1}{\sqrt{\xi(P^{1/2}\mathcal{L}P^{1/2})}} \right), \tag{3.97}$$

where the notation $\widetilde{\mathcal{O}}$ denotes $\mathcal{O}$ with a multiplicative ploylog factor. The above rate is proportional to the square root of the eigengap for the matrix $P^{1/2}\mathcal{L}P^{1/2}$, which is the *unnormalized* Laplacian matrix for graph $\mathcal{G}$.

Remark 3.4.7. **(Non-uniform Lipschitz Constants)** We comment that for the general case $L_i \neq L_j$, $\forall\ i, j$, we can use similar steps to verify that the bound (3.74) derived in Theorem 3.4.2 is optimal, in the sense that they achieve the lower bound (2.65) predicted in Corollary 2.3.2.

3.5 Numerical Results

This section presents numerical examples to show the effectiveness of the proposed algorithms. Two kinds of problems are considered, distributed binary classification and distributed neural networks training. We use the former one to demonstrate the behavior and scalability of our algorithm and use the latter one to show the practical performance.

3.5.1 Simulation Setup

In our simulations, all algorithms are implemented in MATLAB R2017a for binary classification problem and implemented in Python 3.6 for training neural networks, running on a computer node with two 12-core Intel Haswell processors and 128 GB of memory (unless otherwise specified). Both synthetic and real data are used for performance comparison. For synthetic data, the feature vector is randomly generated with standard normal distribution with zero mean and unit variance. The label vector is randomly generated with uniformly distributed pseudorandom integers taking the values

$\{-1, 1\}$. For real data, we use the breast cancer dataset [1] for binary classification and MNIST[2] for training neural network. The breast cancer dataset contains a total of 569 samples each with 30 real positive features. The MNIST dataset contains a total of 60,000 handwritten digits, each with a 28×28 gray scale image and a label from ten categories.

3.5.2 Distributed Binary Classification

We consider a non-convex distributed binary classification problem [64]. The global consensus problem (3.2) can be expressed as follows:

$$\min_{x \in \mathbb{R}^{SM}} \ f(x) := \frac{1}{M} \sum_{i=1}^{M} f_i(x_i), \quad \text{s.t. } x_i = x_j, \forall \ (i,j) \in \mathcal{E}.$$

And each component function f_i is expressed by

$$f_i(x_i) = \frac{1}{B} \sum_{j=1}^{B} \log\left(1 + \exp(-y_{ij} x_i^T v_{ij})\right) + \sum_{s=1}^{S} \frac{\lambda \alpha x_{i,s}^2}{1 + \alpha x_{i,s}^2}.$$

Here $v_{ij} \in \mathbb{R}^S$ denotes the feature vector with dimension S, $y_{ij} \in \{1, -1\}$ denotes the label for the jth date point in ith agent, and there are total B data points for each agent. Unless otherwise noted, the graph $\mathcal{E}$ used in our simulation is generated using the random geometric graph and the graph parameter Ra is set to 0.5. The regularization parameter is set to $\lambda = 0.001, \alpha = 1$.

To compare the convergence performance of the proposed algorithms, we randomly generated MB data points with dimension K and distribute them into M nodes, i.e. each node contains B data points with K features. Then we compare the proposed xFILTER and D-GPDA with the distributed subgradient (DSG) method [44], the Push-sum algorithm [65], and the NEXT algorithm [11]. The parameters for NEXT are chosen as $\tau = 1, \alpha[0] = 0.1$ and $\mu = 0.01$ as suggested by [11], while the parameters for xFILTER are chosen based on (3.61).

[1] https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)
[2] http://yann.lecun.com/exdb/mnist/

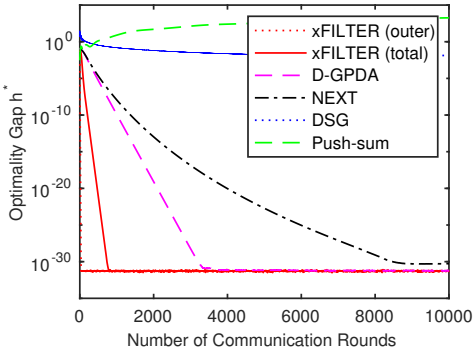


Figure 3.1: Results on synthetic data: $M = 5, B = 200, K = 10$

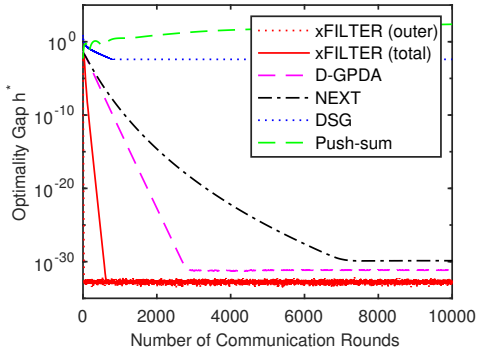


Figure 3.2: Results on synthetic data: $M = 10, B = 200, K = 10$

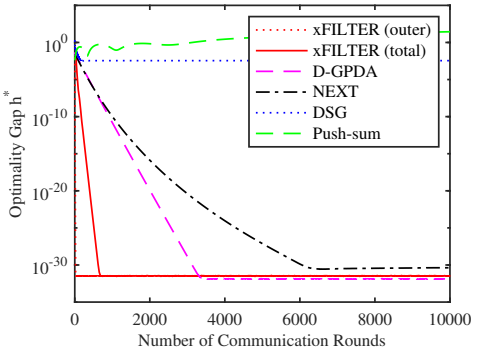


Figure 3.3: Results on synthetic data: $M = 20, B = 200, K = 10$

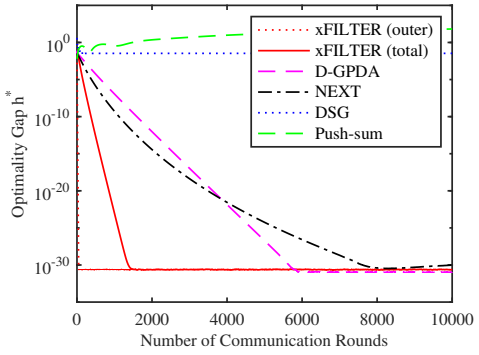


Figure 3.4: Results on synthetic data: $M = 20, B = 50, K = 10$

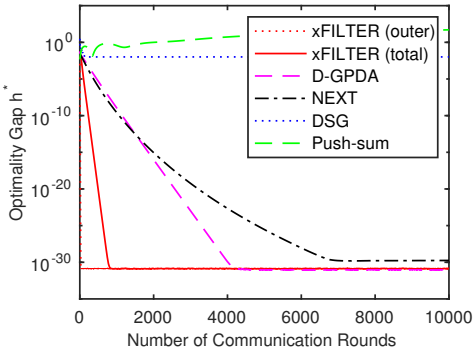


Figure 3.5: Results on synthetic data: $M = 20, B = 100, K = 10$

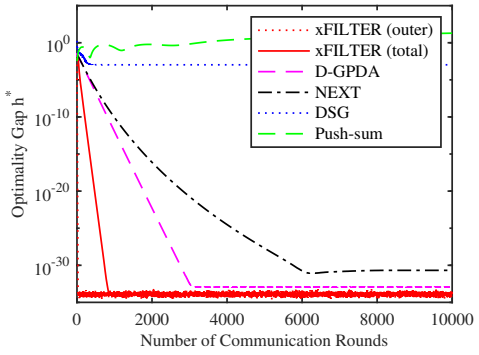


Figure 3.6: Results on synthetic data: $M = 20, B = 400, K = 10$

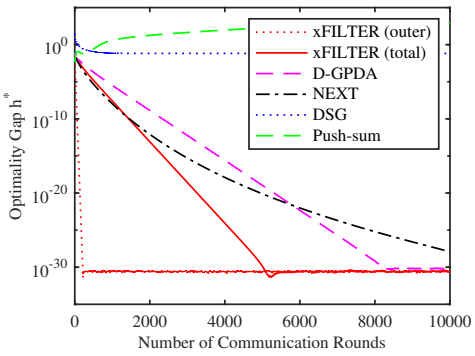


Figure 3.7: Results on synthetic data: $M = 10, B = 20, K = 5$

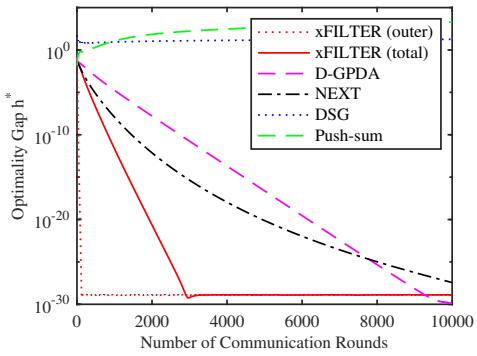


Figure 3.8: Results on synthetic data: $M = 10, B = 20, K = 10$

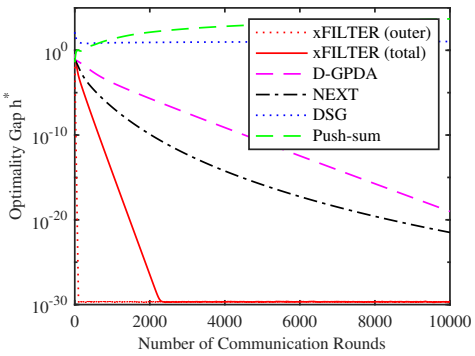


Figure 3.9: Results on synthetic data: $M = 10, B = 20, K = 20$

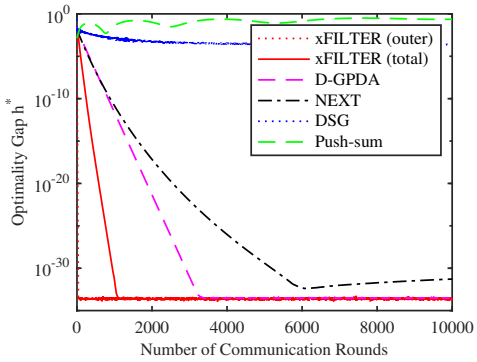


Figure 3.10: Results on synthetic data: $M = 50, B = 2000, K = 10$

Simulation results on synthetic data for different M, B, K averaged over 30 realizations are investigated and shown in Fig. 3.1 to Fig. 3.10, where the x-axis denotes the total rounds of communications required, and the y-axis denotes the quality measure (2.18) proposed in Section 2.2. Note that the curves xFILTER (outer) included in these figures show the number of communication rounds required for xFILTER to perform the "outer" iterations (which is equivalent to r in Algorithm 2, since in each outer iteration only one round of communication is required in Step **S3**). The performance evaluated on real data is also characterized in Fig. 3.11, in which we choose $M = 10$, $B = 56$, and $K = 30$. These results show that the proposed algorithms perform well in all parameter settings compared with existing methods.

We further note that these figures also show (rough) comparison about computation efficiency of different algorithms. Specifically, for D-GPDA, DSG and Push Sum (resp.

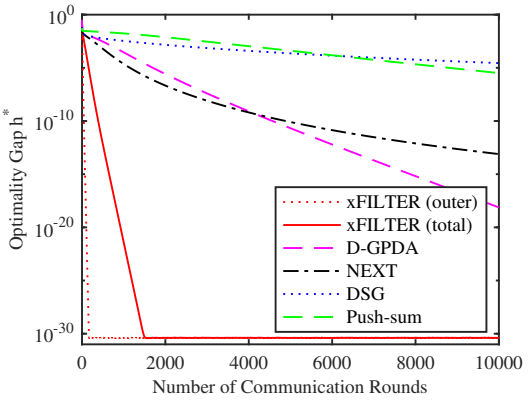
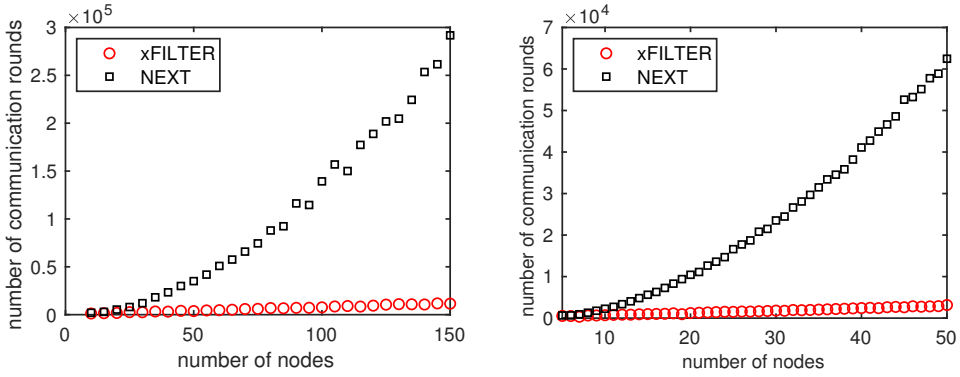


Figure 3.11: Results on real data: $M = 10, B = 56, K = 30$

NEXT), the total rounds of communication is the same as (resp. twice as) the total number of gradient evaluations per node. In contrast, the total rounds of communication in the *outer loop* of xFILTER is the same as the local gradient evaluations. Therefore, the comparison between xFILTER (outer) and other algorithms in Fig. 3.3 to Fig. 3.10 shows the relative computational efficiency of these algorithms. Clearly, xFILTER has a significant advantage over the rest of the algorithms.

Further, we compare the scalability performance of the proposed algorithms with increased network dimension M, and the results are shown in Fig. 3.12, Table 3.2 and Table 3.3. In particular, in Fig. 3.12 we compare the total communication rounds required for NEXT and the xFILTER for reaching $h_T^* \leq 10^{-10}$ and $h_T^* \leq 10^{-15}$, over path graphs with increasing number of nodes. Overall, we see that the xFILTER performs reasonably fast.

We do want to point out that although for the unconstrained problems that we have tested, our proposed algorithms compare relatively favorably with NEXT, NEXT can in fact handle a larger class of problems because it is designed for nonsmooth and constrained nonconvex problems. Further, for all the algorithms we have used, we did not tune the parameters: For xFILTER and D-GPDA, we use the theoretical upper bound suggested in Theorem 3.3.1, and for NEXT we use the parameters suggested in the paper [11]. For all our tested problems and algorithms, it is possible to fine-tune the stepsizes to make them faster, but since this paper is mostly on the theoretical



(a) $B = 10, K = 10, \epsilon = 10^{-10}$ (b) $B = 200, K = 10, \epsilon = 10^{-15}$

Figure 3.12: Comparison of NEXT and xFILTER over path graphs with increasing number of nodes ($M \in [10, 150]$ in (a) and $M \in [5, 50]$ in (b)). Each point in the figure represents the total number of communication needed to reach $h_T^* \leq \epsilon$.

Table 3.2: Optimality gap after 200 rounds of communications ($B = 200, K = 10$)

number of nodes M	D-GPDA	xFILTER
10	3.96×10^{-4}	2.50×10^{-11}
20	5.45×10^{-4}	1.92×10^{-9}
30	1.20×10^{-4}	4.71×10^{-11}
40	2.95×10^{-4}	4.07×10^{-10}
50	3.88×10^{-4}	8.47×10^{-11}

properties of rate optimal algorithms, we choose not to go down that path.

3.5.3 Distributed Neural Network Training

In our second experiment, we present some numerical results under a more realistic setting. We consider training a neural network model for fitting the MNIST data set. The dataset is first randomly partitioned into 10 subsets, and then gets distributed over 10 machines. A fully connected neural network with one hidden layer is used in the experiment. The number of neurons for the hidden layer and the output layer are set as 128 and 10, respectively. The initial weights for the neural network are drawn from a truncated normal distribution centered at zero with variance scaled with the number of input units. The algorithms are written in Python, and the communication protocol is implemented using the Message Passing Interface (MPI). The empirical performance of

Table 3.3: Optimality gap after 1000 rounds of communications ($B = 200, K = 10$)

number of nodes M	D-GPDA	xFILTER
10	8.24×10^{-13}	1.93×10^{-33}
20	9.41×10^{-12}	1.43×10^{-32}
30	2.09×10^{-13}	2.26×10^{-32}
40	1.52×10^{-11}	4.19×10^{-33}
50	2.30×10^{-10}	6.48×10^{-33}

the xFILTER is evaluated and compared with the DSG algorithm [66]. Fig. 3.13 shows that, compared with DSG, the proposed algorithm achieves better communication and computation efficiency, and has improved classification accuracy.

Note that despite the fact that some global parameters (such as the Lipschitz constants) are unknown, the rules provided in (3.61) or (3.67) still can help us roughly estimate a set of good parameters. For example, we choose the following parameters

$$\Sigma^2 = \frac{\sigma}{\sum_i d_i \underline{\lambda}_{\min}(\widetilde{\mathcal{L}})}, \quad \Upsilon^2 = \frac{\beta P}{\sum_i d_i}, \tag{3.98}$$

and tune the parameter β and σ by searching from the set $\{0.1, 0.2, 0.5, 1, 2, 5, \cdots, 100, 200, 500\}$. Based on the best practical performance over 10 runs, we choose $\beta = 100$ and $\sigma = 20$ for xFILTER and $\alpha = 0.1$ for DSG.

3.6 Proofs of Lemmas and Theorems

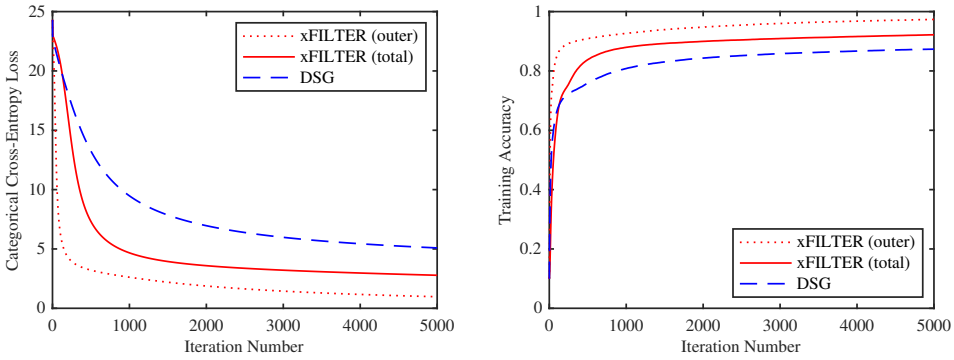
3.6.1 Proof of Lemma 3.3.1

Proof. First we show that for all $r \geq -1$ the following holds for D-GPDA

$$\nabla f(x^r) + F^T \lambda^r + F^T \Sigma^2 F x^{r+1} + H(x^{r+1} - x^r) = 0. \tag{3.99}$$

Note that for the initialization (3.6) we have

$$\nabla f(x^{-1}) + (2\Delta + \Upsilon^2) x^0 = \nabla f(x^{-1}) + (F^T \Sigma^2 F + H) x^0 = 0.$$



(a) Training Loss (b) Training Accuracy

Figure 3.13: Comparison of DSG and xFILTER over path graphs on distributed training neural networks; Plot (a) shows the dynamic of the categorical cross-entropy loss, and plot (b) shows the training classification accuracy. The parameters are chosen based on their best practical performance through grid search. The curves xFILTER (outer) and xFILTER (total) again represent the number of outer iteration, and the total number of iterations required for xFILTER.

Setting $x^{-1} = 0, \lambda^{-1} = 0$ and using (3.6), we obtain

$$\nabla f(0) + F^T \lambda^{-1} + F^T \Sigma^2 F x^0 + H(x^0 - x^{-1}) = 0. \tag{3.100}$$

Further, the optimality condition of the x update (3.8a) suggests that (3.99) holds for all $r \geq 0$, therefore (3.99) is proved.

Second, by using (3.99) and the y update (3.8b), we obtain

$$F^T \lambda^{r+1} = -\nabla f(x^r) - H(x^{r+1} - x^r), \ \forall \ r \geq -1. \tag{3.101}$$

Then subtracting the previous iteration leads to

$$F^T(\lambda^{r+1} - \lambda^r) = -(\nabla f(x^r) - \nabla f(x^{r-1})) - H w^{r+1}, \ \forall \ r \geq 0.$$

Note that the matrix $H \succ 0$, $\Sigma^2 \succ 0$, then we have

$$H^{-1/2}(\Sigma F)^T \Sigma^{-1}(\lambda^{r+1} - \lambda^r) = -H^{-1/2}(\nabla f(x^r) - \nabla f(x^{r-1})) - H^{1/2} w^{r+1}. \tag{3.102}$$

Then using the fact that

$$\Sigma^{-1}(\lambda^{r+1}-\lambda^r)=\Sigma F x^{r+1}\in \mathrm{col}(\Sigma F),$$

we can square both sides and obtain the following

$$\begin{aligned}
&\underline{\lambda}_{\min}(\Sigma F H^{-1}F^T\Sigma)\|\Sigma^{-1}(\lambda^{r+1}-\lambda^r)\|^2\\
&\le 2\|H^{-1/2}(\nabla f(x^r)-\nabla f(x^{r-1}))\|^2+2(w^{r+1})^T H w^{r+1}\\
&\le 2\|\Upsilon^{-1}(\nabla f(x^r)-\nabla f(x^{r-1}))\|^2+2(w^{r+1})^T H w^{r+1}\\
&\overset{(2.20)}{\le} \frac{2}{M^2}\|\Upsilon^{-1}L(x^r-x^{r-1})\|^2+2\|w^{r+1}\|_H^2,\ \forall\ r\ge 0.
\end{aligned}\tag{3.103}$$

This concludes the proof of the first part.

To show the second part, note that according to (3.18b), x^{r+1} generated by xFILTER is given by (for all $r\ge -1$)

$$\nabla f(x^r)+F^T(\lambda^r+\Sigma^2 F x^{r+1})+\Upsilon^2(x^{r+1}-x^r)=\Upsilon^2 R\epsilon^{r+1}.\tag{3.104}$$

Then use the same analysis steps as in the first part, we arrive at the desired result. **Q.E.D.**

3.6.2 Proof of Lemma 3.3.2

Proof. Using the Lipschitz gradient assumption (2.20), we have

$$\begin{aligned}
&\mathsf{AL}(x^{r+1},\lambda^r)-\mathsf{AL}(x^r,\lambda^r)\le \langle \nabla f(x^r)+f^T\lambda^r+F^T\Sigma^2 F x^r, x^{r+1}-x^r\rangle\\
&\quad+\frac{1}{2M}\|x^{r+1}-x^r\|_L^2+\frac{1}{2}\|\Sigma F(x^{r+1}-x^r)\|^2\\
&=\langle \nabla f(x^r)+F^T\lambda^r+A^T\Sigma^2 F x^{r+1}, x^{r+1}-x^r\rangle\\
&\quad+\langle H(x^{r+1}-x^r), x^{r+1}-x^r\rangle+\frac{1}{2M}\|x^{r+1}-x^r\|_L^2\\
&\quad+\frac{1}{2}\|\Sigma F(x^{r+1}-x^r)\|^2-\|x^{r+1}-x^r\|_{H+F^T\Sigma^2 F}\\
&\overset{(3.99),(2.22)}{\le} -(x^{r+1}-x^r)^T\left(\frac{\Delta}{2}-\frac{L}{2M}+\Upsilon^2\right)(x^{r+1}-x^r).
\end{aligned}\tag{3.105}$$

Using the update rule of the dual variable, and combine the above inequality, we obtain

$$\begin{aligned}\mathsf{AL}^{r+1} - \mathsf{AL}^{r} &\leq -\frac{1}{2}\|x^{r+1} - x^{r}\|^2_{\Delta+2\Upsilon^2-L/M} + \langle \lambda^{r+1} - \lambda^{r}, Ax^{r+1}\rangle \\ &= -\frac{1}{2}\|x^{r+1} - x^{r}\|^2_{\Delta+2\Upsilon^2-L/M} + \|\Sigma^{-1}(\lambda^{r+1} - \lambda^{r})\|^2\end{aligned}$$

Combined with Lemma 3.3.1 we complete the first part.

The second part follows similar steps. The modifications are that H is replaced by Υ^2, and that there is an additional error term in the optimality condition; cf. (3.18b). **Q.E.D.**

3.6.3 Proof of Lemma 3.3.5 and Lemma 3.3.6

Proof. Using the optimality condition (3.99), we have

$$\langle F^T\lambda^{r+1} + \nabla f(x^r) + H(x^{r+1} - x^r), x^{r+1} - x\rangle = 0,\ \forall\ r \geq -1$$

This implies that for all $r \geq 0$

$$\langle F^T(\lambda^{r+1} - \lambda^r) + \nabla f(x^r) - \nabla f(x^{r-1}) + Hw^{r+1}, x^{r+1} - x^r\rangle = 0.$$

It follows that

$$\begin{aligned}&\frac{1}{2}\|\Sigma F x^{r+1}\|^2 + \frac{1}{2}\|x^{r+1} - x^r\|^2_H \leq \frac{1}{2}\|\Sigma F x^r\|^2 + \frac{1}{2}\|x^r - x^{r-1}\|_H - \frac{1}{2}\|w^{r+1}\|^2_H \quad (3.106)\\ &\quad + \frac{1}{2M}\|x^{r+1} - x^r\|^2_L + \frac{1}{2M}\|x^r - x^{r-1}\|^2_L, \quad \forall\ r \geq 0.\end{aligned}$$

Then combining Lemma 3.3.2 and (3.106), for all $r \geq 0$ we have

$$\begin{aligned}&P^{r+1} - P^r \leq -\left(\frac{c}{2} - 2\kappa\right)\|w^{r+1}\|^2_H \\ &-\frac{1}{2}(x^{r+1} - x^r)^T\left(\Delta + 2\Upsilon^2 - \frac{L}{M} - \frac{4\kappa}{M^2}L\Upsilon^{-2}L - \frac{2cL}{M}\right)(x^{r+1} - x^r).\end{aligned}$$

Therefore, in order to make the potential function decrease, we need to follow (3.44).

To show a similar result for the xFILTER, consider the following optimality condition

derived from (3.104)

$$\langle F^T\lambda^{r+1}+\nabla f(x^r)+\Upsilon^2(x^{r+1}-x^r)-\Upsilon^2R\epsilon^{r+1},x^{r+1}-x\rangle=0,\ \forall\ x.$$

Following similar steps as in (3.106), and use (3.39), we have

$$\begin{aligned}\frac{1}{2}\|\Sigma Fx^{r+1}\|^2+\frac{1}{2}\|x^{r+1}-x^r\|^2_{\Upsilon^2}\le\frac{1}{2}\|\Sigma Fx^r\|^2+\frac{1}{2}\|x^r-x^{r-1}\|_{\Upsilon^2}-\frac{1}{2}\|w^{r+1}\|^2_{\Upsilon^2}\\ +1/(2M)\|x^{r+1}-x^r\|^2_L+1/(2M)\|x^r-x^{r-1}\|^2_L\\ +1/4\|x^{r+1}-x^r\|^2_{\Upsilon^2R}+1/4\|x^r-x^{r-1}\|^2_{\Upsilon^2R},\quad\forall\ r\ge-1.\end{aligned}\tag{3.107}$$

Then combining Lemma 3.3.2, (3.106), and the estimate of the size of ϵ in (3.39), we have

$$\widetilde{P}^{r+1}-\widetilde{P}^r\le-\frac{1}{2}(x^{r+1}-x^r)^TV(x^{r+1}-x^r)-\left(\frac{\widetilde{c}}{2}-3\widetilde{\kappa}\right)\|w^{r+1}\|^2_{\Upsilon^2}.$$

with

$$V:=\left(\Upsilon^2R-(1+2\tilde{c})\frac{L}{M}-\frac{6\tilde{\kappa}}{M^2}L\Upsilon^{-2}L-\frac{\Upsilon^2R(24\tilde{\kappa}+6+16\tilde{c})}{16}\right).$$

Therefore in order to make the potential function decrease, we need to follow (3.46). **Q.E.D.**

3.6.4 Proof of Lemma 3.3.7

Proof. For D-GPDA, we can express the AL as (for all $r\ge0$)

$$\begin{aligned}&\mathsf{AL}^{r+1}-f(x^{r+1})=\langle\lambda^{r+1},\Sigma^{-2}(\lambda^{r+1}-\lambda^r)\rangle+\frac{1}{2}\|\Sigma Fx^{r+1}\|^2\\&=\frac{1}{2}\big(\|\Sigma^{-1}\lambda^{r+1}\|^2-\|\Sigma^{-1}\lambda^r\|^2+\|\Sigma^{-1}(\lambda^{r+1}-\lambda^r)\|^2+\|\Sigma Fx^{r+1}\|^2\big).\end{aligned}$$

Since $\inf_x f(x)=\underline{f}$ is lower bounded, let us define

$$\widehat{\mathsf{AL}}^{r+1}:=\mathsf{AL}^{r+1}-\underline{f},\ \widehat{f}(x):=f(x)-\underline{f}\ge0,\ \widehat{P}^{r+1}:=P^{r+1}-\underline{f}.$$

Therefore, summing over $r = -1 \cdots, T$, we obtain

$$\sum_{r=-1}^{T} \widehat{\mathsf{AL}}^{r+1} = \frac{1}{2}\left(\|\Sigma^{-1}\lambda^{T+1}\|^2 - \|\Sigma^{-1}\lambda^{-1}\|^2\right)$$
$$+ \sum_{r=-1}^{T}\left(\widehat{f}(x^{r+1}) + \frac{1}{2}\|\Sigma F x^{r+1}\|^2 + \frac{1}{2}\|\Sigma^{-1}(\lambda^{r+1} - \lambda^r)\|^2\right).$$

Using the initialization $\lambda^{-1} = 0$, then the above sum is lower bounded by zero. This fact implies that the sum of $\widehat{P}^{r+1}$ is also lower bounded by zero (note, the remaining terms in the potential function are all nonnegative)

$$\sum_{r=0}^{T} \widehat{P}^{r+1} \geq 0, \quad \forall\ T > 0,$$

Note that if the parameters of the system are chosen according to (3.44), then P^{r+1} is nonincreasing, which implies that its shifted version $\widehat{P}^{r+1}$ is also nonincreasing. Combined with the nonnegativity of the sum of the shifted potential function, we can conclude that

$$\widehat{P}^{r+1} \geq 0, \quad \text{and} \quad P^{r+1} \geq \inf f(x), \quad \forall\ r \geq 0. \tag{3.108}$$

Next we compute P^0. By using (2.22), we have

$$P^0 = \mathsf{AL}^0 + \frac{2\kappa}{M^2}\|\Upsilon^{-1}Lx^0\|^2 + \frac{c}{2}\left(\|x^0\|^2_{2\Delta+\Upsilon^2+L/M}\right) \tag{3.109}$$
$$\mathsf{AL}^0 \leq f(x^0) + 2\|\Sigma F x^0\|^2$$
$$x^0 \overset{(3.6)}{=} (2\Delta + \Upsilon^2)^{-1}\frac{1}{M}[\nabla f_1(0); \cdots; \nabla f_M(0)]$$
$$= (2\Delta + \Upsilon^2)^{-1}\frac{1}{M}d_0 \tag{3.110}$$

where in the last equality we have used the definition of d_0 in (2.48). Use the above relations, we have

$$P^0 \leq f(x^0) + (x^0)^T Z x^0 \tag{3.111}$$

with the matrix Z defined as

$$Z = \frac{2\kappa}{M^2} L\Upsilon^{-2}L + \frac{c(2\Delta + \Upsilon^2 + L/M)}{2} + 2F\Sigma^2 F \preceq 4c(2\Delta + \Upsilon^2)$$

where the last inequality follows from our choice of parameters in (3.44b), and the fact $c \geq 1$. Note that

$$\begin{aligned} 4c\|x^0\|^2_{2\Delta+\Upsilon^2} &\leq \frac{4c}{M^2} d_0^T (2\Delta + \Upsilon^2)^{-1}(2\Delta + \Upsilon^2)(2\Delta + \Upsilon^2)^{-1} d_0 \\ &= \frac{4c}{M^2} d_0^T (2\Delta + \Upsilon^2)^{-1} d_0 \leq \frac{2}{M} d_0^T L^{-1} d_0 \end{aligned} \tag{3.112}$$

where the last inequality comes from the choice of the parameters (3.44b), which implies that $2\Delta + \Upsilon^2 \succeq 2\frac{Lc}{M}$. These constants combined with (3.109) shows the desired result.

For xFILTER, the proof for the lower boundedness is the same. To bound the size of $\widetilde{P}^0$, first note that we again have

$$\mathsf{AL}^{r+1} - f(x^{r+1}) = \frac{1}{2}\big(\|\Sigma^{-1}\lambda^{r+1}\|^2 - \|\Sigma^{-1}\lambda^r\|^2 + \|\Sigma^{-1}(\lambda^{r+1} - \lambda^r)\|^2 + \|\Sigma F x^{r+1}\|^2\big)\,.$$

By letting $r = -1$, and use the fact that $x^{-1} = 0$ and $\lambda^{-1} = 0$, we obtain

$$\mathsf{AL}^0 - f(x^0) = \frac{1}{2}\big(2\|\Sigma^{-1}\lambda^0\|^2 + \|\Sigma F x^0\|^2\big) = \frac{3}{2}\|\Sigma^{-1}\lambda^0\|^2. \tag{3.113}$$

Then we have

$$\begin{aligned} \widetilde{P}^0 &= \mathsf{AL}^0 + \frac{3\widetilde{\kappa}}{M^2}\|\Upsilon^{-1}Lx^0\|^2 + \frac{3}{8}\tilde{\kappa}\|x^0\|^2_{\Upsilon^2 R} \\ &\quad + \frac{\widetilde{c}}{2}\left(\|\Sigma F x^0\|^2 + \|x^0\|^2_{\Upsilon^2 + \Upsilon^2 R/4 + L/M}\right), \end{aligned} \tag{3.114}$$

$$\mathsf{AL}^0 \leq f(x^0) + 2\|\Sigma F x^0\|^2,\ x^{-1} = 0,\ \lambda^{-1} = 0, \tag{3.115}$$

$$x^0 \overset{(3.18b)}{=} R^{-1}\Upsilon^{-2}\nabla f(0) - \epsilon^0, \quad \widetilde{\epsilon}^{-1} \overset{(3.35)}{=} R^{-1}\Upsilon^{-2}\nabla f(0). \tag{3.116}$$

Use the above relation, we have

$$\widetilde{P}^0 \leq f(x^0) + (x^0)^T \widetilde{Z} x^0 \tag{3.117}$$

with the matrix Z defined as

$$\widetilde{Z} = \frac{3\tilde{\kappa}}{M^2} L\Upsilon^{-2}L + \left(\frac{3}{8}\tilde{\kappa} + \tilde{c}\right)\Upsilon^2 R + \frac{\tilde{c}L}{2M} + 2F\Sigma^2 F \preceq 3\Upsilon^2 R$$

where the last inequality follows from our choice of parameters in (3.46b). Therefore we have

$$\begin{aligned}
(x^0)^T \widetilde{Z} x^0 &\leq 3(x^0)^T \Upsilon^2 R x^0 \\
&\leq 3(\nabla f(0) - \Upsilon^2 R\epsilon^0)^T R^{-1}\Upsilon^{-2}(\nabla f(0) - \Upsilon^2 R\epsilon^0) \\
&\overset{\text{(i)}}{\leq} 6(\nabla f(0))^T R^{-1}\Upsilon^{-2}\nabla f(0) + 6(\epsilon^0)^T\Upsilon^2 R\epsilon^0 \\
&\leq 3M(\nabla f(0))^T L^{-1}\nabla f(0) + \frac{3}{8M}\|x^0\|^2_{\Upsilon^2 R} \qquad (3.118)
\end{aligned}$$

where in (i) we have used the Cauchy-Swartz inequality; the last inequality uses (3.39), the choice of the parameters (3.46b) (which implies $\Upsilon^2 R \geq 4L/M$). The above series of inequalities imply that

$$2\|x^0\|^2_{\Upsilon^2 R} \leq \left(3 - \frac{3}{8M}\right)\|x^0\|^2_{\Upsilon^2 R} \leq 3M(\nabla f(0))^T L^{-1}\nabla f(0).$$

Therefore overall we have

$$(x^0)^T \widetilde{Z} x^0 \leq 3(x^0)^T \Upsilon^2 R x^0 \leq 5M(\nabla f(0))^T L^{-1}\nabla f(0). \qquad (3.119)$$

Finally, by observing $\frac{1}{M^2} d_0^T d_0 = \|\nabla f(0)\|^2$, the desired result is obtained. **Q.E.D.**

3.6.5 Proof of Theorem 3.3.1

Proof. To show the first part, we consider the optimality condition (3.101), and multiply both sides of it by the all one vector, and use the fact that $1^T A^T = 0$ to obtain

$$1^T \nabla f(x^r) + 1^T H(x^{r+1} - x^r) = 0, \ \forall\ r \geq -1.$$

Squaring both sides and rearranging terms, we have

$$\begin{aligned}\left\|\frac{1}{M}\sum_{i=1}^{M}\nabla f_i(x_i^r)\right\|^2 &\leq (x^{r+1}-x^r)^T H 1 1^T H (x^{r+1}-x^r)\\ &\leq (x^{r+1}-x^r)^T H (x^{r+1}-x^r)\times 1^T H 1\\ &\leq \|x^{r+1}-x^r\|_H^2 \times \left(4\sum_{(i,j)i\sim j}\sigma_{ij}^2+\sum_{i=1}^{M}\beta_i^2\right),\ \forall\ r\geq -1.\end{aligned}$$

Combining with (3.45), we obtain, for all $r \geq 0$

$$\begin{aligned}\left\|\frac{1}{M}\sum_{i=1}^{M}\nabla f_i(x_i^r)\right\|^2 &\leq \|x^{r+1}-x^r\|_H^2\left(4\sum_{e\in\mathcal{E}}\sigma_e^2+\sum_{i=1}^{M}\beta_i^2\right)\\ &\leq 8\left(P^r-P^{r+1}\right)\left(4\sum_{e\in\mathcal{E}}\sigma_e^2+\sum_{i=1}^{M}\beta_i^2\right).\end{aligned}\tag{3.120}$$

where in the last inequality we used $2(\Delta+\Upsilon^2)\succeq H$. We then bound the consensus error. Lemma 3.3.1 implies

$$\begin{aligned}\|\Sigma F x^{r+1}\|^2 &\leq \kappa\left(\frac{2}{M^2}\|\Upsilon^{-1}L(x^r-x^{r-1})\|^2+2\|w^{r+1}\|_H^2\right)\\ &\overset{(3.44a)}{\leq} 2\kappa\left(4\|w^{r+1}\|_H^2+\frac{2}{M^2}\|\Upsilon^{-1}L(x^{r+1}-x^r)\|^2\right).\end{aligned}\tag{3.121}$$

Therefore

$$\|\Sigma F x^r\|^2 \leq 4\kappa\left(4\|w^{r+1}\|_H^2+\frac{2}{M^2}\|\Upsilon^{-1}L(x^{r+1}-x^r)\|^2\right)+2\|\Sigma F(x^{r+1}-x^r)\|^2.\tag{3.122}$$

Combining with (3.45), and using the fact that [cf. (3.44b)]

$$\Delta+\Upsilon^2\succeq\frac{8\kappa L\Upsilon^{-2}L}{M^2},\quad 2\Delta\succeq F\Sigma^2F\tag{3.123}$$

we have

$$\|\Sigma F x^r\|^2 \leq 16\kappa \|w^{r+1}\|_H^2 + 5\|x^{r+1} - x^r\|^2_{\Delta+\Upsilon^2} \overset{(3.45)}{\leq} 20(P^r - P^{r+1}). \tag{3.124}$$

Also note that by the definition of $e(T)$ we have

$$T \times e(T) \leq \sum_{r=1}^{T} \left(\|\Sigma F x^r\|^2 + \left\| \frac{1}{M} \sum_{i=1}^{M} \nabla f_i(x_i^r) \right\|^2 \right) \tag{3.125}$$

Then combining the above with (3.45) and (3.124), and the fact that the potential function is lower bounded by $\underline{f}$, we obtain the desired result.

To show the result for xFILTER, multiply both sides of the optimality condition (3.104) by the all one vector, and use the fact that $F1 = 0$ to obtain

$$1^T \nabla f(x^r) + 1^T \Upsilon^2 (x^{r+1} - x^r) = 1^T \Upsilon^2 R \epsilon^{r+1}. \tag{3.126}$$

Squaring both sides and rearranging terms we have

$$\begin{aligned}
\left\| \frac{1}{M} \sum_{i=1}^{M} \nabla f_i(x_i^r) \right\|^2 &\leq 2(x^{r+1} - x^r)^T \Upsilon^2 1 1^T \Upsilon^2 (x^{r+1} - x^r) + 2(\epsilon^{r+1})^T \Upsilon^2 R 1 1^T \Upsilon^2 R \epsilon^{r+1} \\
&\overset{(3.39)}{\leq} 2(x^{r+1} - x^r)^T \Upsilon^2 (x^{r+1} - x^r) \times 1^T \Upsilon^2 1 + M/(4M) \|x^{r+1} - x^r\|^2_{\Upsilon^2 R} \\
&\leq \|x^{r+1} - x^r\|^2_{\Upsilon^2 R} \times 2\left(1 + \sum_{i=1}^{M} \beta_i^2\right), \ \forall\ r \geq -1.
\end{aligned}$$

where in the last inequality we have used the fact that $\Upsilon^2 R = \Upsilon^2 + F^T \Sigma^2 F \succeq \Upsilon^2$.

To bound the consensus error, we first use (3.39) and obtain

$$\|\Upsilon^2 R(\epsilon^{r+1} - \epsilon^r)\|^2 \leq \frac{1}{4M} \|x^{r+1} - x^r\|^2_{\Upsilon^2 R} + \frac{1}{4M} \|x^r - x^{r-1}\|^2_{\Upsilon^2 R}.$$

Similarly as the first part, we use Lemma 3.3.1 and obtain

$$\begin{aligned}
&\|\Sigma F x^{r+1}\|^2 \\
&\leq 3\widetilde{\kappa}\left(\|x^{r+1}-x^r\|^2_{\frac{\Upsilon^2 R}{4M}}+\|w^{r+1}\|^2_{\Upsilon^2}+\|x^r-x^{r-1}\|^2_{\frac{\Upsilon^2 R}{4M}+\frac{L\Upsilon^{-2}L}{M^2}}\right) \\
&\leq 2\|x^{r+1}-x^r\|^2_{\Upsilon^2 R}+3\tilde{\kappa}\|w^{r+1}\|^2_{\Upsilon^2}+2\|x^r-x^{r-1}\|^2_{\Upsilon^2 R},\ \ \forall\ r\geq 0
\end{aligned} \tag{3.127}$$

where the last inequality comes from (3.46b), that

$$2\Upsilon^2 R \succeq 3\widetilde{\kappa}\left(\frac{L\Upsilon^{-2}L}{M^2}+\Upsilon^2 R\right). \tag{3.128}$$

By combining (3.127) and the following inequality

$$\|\Sigma F x^r\|^2 \leq 2\|\Sigma F(x^{r+1}-x^r)\|^2+2\|\Sigma F x^{r+1}\|^2,$$

we have

$$\begin{aligned}
\|\Sigma F x^r\|^2 &\leq 4\|x^{r+1}-x^r\|^2_{\Upsilon^2 R+F^T\Sigma^2 F}+6\tilde{\kappa}\|w^{r+1}\|^2_{\Upsilon^2}+4\|x^r-x^{r-1}\|^2_{\Upsilon^2 R} \\
&\overset{(3.47)}{\leq} 64(\widetilde{P}^r-\widetilde{P}^{r+1})+64(\widetilde{P}^{r-1}-\widetilde{P}^r),\quad \forall\ r\geq 1 \\
\|\Sigma F x^0\|^2 &\leq 64(\widetilde{P}^0-\widetilde{P}^1)+4\|x^0\|^2_{\Upsilon^2 R}.
\end{aligned}$$

So overall we have that

$$\begin{aligned}
&\sum_{r=0}^{T_r}\left(\left\|\frac{1}{M}\sum_{i=1}^{M}\nabla f_i(x_i^r)\right\|^2+\|\Sigma F x^r\|^2\right) \\
&\leq 64\left(1+\sum_{i=1}^{M}\beta_i^2+1\right)\sum_{r=1}^{T_r}((\widetilde{P}^r-\widetilde{P}^{r+1})+(\widetilde{P}^{r-1}-\widetilde{P}^r)) \\
&\quad +64(\widetilde{P}^0-\widetilde{P}^1)+4\|x^0\|^2_{\Upsilon^2 R} \\
&\leq 128\left(1+\sum_{i=1}^{M}\beta_i^2+2\right)(\widetilde{P}^0-\underline{f})+4\|x^0\|^2_{\Upsilon^2 R}.
\end{aligned} \tag{3.129}$$

where the last inequality utilizes the descent property of $\widetilde{P}^0$ in Lemma 3.3.6. Note that

from (3.113), (3.114) and use $\widetilde{c} = 8\widetilde{\kappa}$ in (3.46a), we obtain

$$\widetilde{P}^0 \geq f(x^0) + \widetilde{\kappa}\|x^0\|^2_{\Upsilon^2 R}. \tag{3.130}$$

Therefore From (3.49) and Lemma 3.3.7 we have that

$$\begin{aligned} 4\|x^0\|^2_{\Upsilon^2 R} &\leq \frac{4\left(\widetilde{P}^0 - f(x^0)\right)}{\widetilde{\kappa}} \\ &\overset{(3.49)}{\leq} \frac{4\left(f(x^0) + \frac{5}{M} d_0^\top L^{-1} d_0 - \underline{f}\right)}{\widetilde{\kappa}} := \frac{4\widetilde{C}_1}{\widetilde{\kappa}}. \end{aligned}$$

Combining the above two relations leads to

$$\begin{aligned} &\frac{1}{T_r}\sum_{r=0}^{T_r}\left(\Big\|\frac{\sum_{i=1}^M \nabla f_i(x_i^r)}{M}\Big\|^2 + \|\Sigma F x^r\|^2\right) \\ &\leq 128\left(\sum_{i=1}^M \beta_i^2 + 3 + \frac{1}{32\widetilde{\kappa}}\right)\widetilde{C}_1/T_r. \end{aligned}$$

This completes the proof. **Q.E.D.**

Chapter 4

Improved Stochastic Algorithms for Distributed Non-convex Learning

Many modern large-scale machine learning problems benefit from decentralized and stochastic optimization. Recent works have shown that utilizing both decentralized computing and local stochastic gradient estimates can outperform state-of-the-art centralized algorithms, in applications involving highly non-convex problems, such as training deep neural networks. In this work, we propose a decentralized stochastic algorithm to deal with certain smooth non-convex problems where there are m nodes in the system, and each node has a large number of samples (denoted as n). Differently from the majority of the existing decentralized learning algorithms for either stochastic or finite-sum problems, our focus is given to *both* reducing the total communication rounds among the nodes, while accessing the minimum number of local data samples. In particular, we propose an algorithm named D-GET (decentralized gradient estimation and tracking), which jointly performs decentralized gradient estimation (which estimates the local gradient using a subset of local samples) *and* gradient tracking (which tracks the global full gradient using local estimates). We show that, to achieve certain ϵ stationary solution of the deterministic finite sum problem, the proposed algorithm achieves an $\mathcal{O}(mn^{1/2}\epsilon^{-1})$ sample complexity and an $\mathcal{O}(\epsilon^{-1})$ communication complexity. These

bounds significantly improve upon the best existing bounds of $\mathcal{O}(mn\epsilon^{-1})$ and $\mathcal{O}(\epsilon^{-1})$, respectively. Similarly, for online problems, the proposed method achieves an $\mathcal{O}(m\epsilon^{-3/2})$ sample complexity and an $\mathcal{O}(\epsilon^{-1})$ communication complexity.

4.1 Introduction

Recent advances of decentralized optimization enable us to utilize distributed resources to significantly improve the computation efficiency [67, 68]. Compared to the typical parameter-server type distributed system with a fusion center, decentralized optimization has its unique advantages in preserving data privacy, enhancing network robustness, and improving the computation efficiency [66, 68–70]. Furthermore, in many emerging applications such as collaborative filtering [71], federated learning [72] and dictionary learning [73], the data is naturally collected in a decentralized setting, and it is not possible to transfer the distributed data to a central location. Therefore, decentralized computation has sparked considerable interest in both academia and industry.

Motivated by these facts, in this paper we consider the following decentralized optimization problem,

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^{md}} f(\mathbf{x}) &= \frac{1}{m}\sum_{i=1}^{m} f^i(\mathbf{x}_i), \\ \text{s.t.} \quad \mathbf{x}_i &= \mathbf{x}_k, \quad \forall (i,k) \in \mathcal{E}. \end{aligned} \tag{4.1}$$

where $f^i(\cdot) : \mathbb{R}^d \to \mathbb{R}$ denotes the loss function which is smooth (possibly non-convex), and m is the total number of such functions. We consider the scenario where each node $i \in [m] := \{1, \cdots, m\}$ can only access its local function $f^i(\cdot)$, and can communicate with its neighbors via an undirected and unweighted graph $\mathcal{G} = \{\mathcal{E}, \mathcal{V}\}$. And $\mathbf{x}$ stacks all the variables: $\mathbf{x} := [\mathbf{x}_1; \mathbf{x}_2; \cdots; \mathbf{x}_m] \in \mathbb{R}^{md}$.

In this work, we consider two typical representations of the local cost functions:

1. **Finite-Sum Setting:** Each $f^i(\cdot)$ is defined as the average cost of n local samples, that is:

$$f^i(\cdot) = \frac{1}{n}\sum_{j=1}^{n} f_j^i(\cdot), \forall i \tag{4.2}$$

where n is the total number of local samples at node i, $f_j^i(\cdot)$ denotes the cost for

jth data sample at ith node.

2. **Online Setting:** Each $f^i(\cdot)$ is defined as:

$$f^i(\cdot) = \mathbb{E}_{\xi \sim \mathcal{D}_i}[f^i_\xi(\cdot)], \forall i \tag{4.3}$$

where $\mathcal{D}_i$ denotes the data distribution at node i.

For the above decentralized non-convex problem (4.1), one essential task is to find an ϵ stationary solution $\mathbf{x}^* := [\mathbf{x}_1^*; \cdots; \mathbf{x}_m^*] \in \mathbb{R}^{md}$ such that the optimality gap h^* satisfies

$$\left\| \frac{1}{m} \sum_{i=1}^{m} \nabla f^i(\mathbf{x}_i^*) \right\|^2 + \frac{1}{m} \sum_{i}^{m} \left\| \mathbf{x}_i^* - \frac{1}{m} \sum_{i}^{m} \mathbf{x}_i^* \right\|^2 \leq \epsilon. \tag{4.4}$$

This solution quality measure encodes both the size of local gradient error for classical centralized non-convex problems and the consensus error for decentralized optimization.

Many modern decentralized methods can be applied to obtain the above mentioned ϵ stationary solution for problem (4.1). In the finite-sum setting (4.2), deterministic decentralized methods [3, 74–76], which process the local dataset in full batches, typically achieve $\mathcal{O}(\epsilon^{-1})$ communication complexity (i.e., $\mathcal{O}(\epsilon^{-1})$ rounds of message exchanges are required to obtain ϵ stationary solution), and $\mathcal{O}(mn\epsilon^{-1})$ sample complexity.[1] Meanwhile, stochastic methods [68, 77–79], which randomly pick subsets of local samples, achieve $\mathcal{O}(m\epsilon^{-2})$ sample and $\mathcal{O}(\epsilon^{-2})$ communication complexity. These complexity bounds indicate that, when the sample size is large (i.e., $\epsilon^{-1} = o(n)$), the stochastic methods are preferred for lower sample complexity, but the deterministic methods still achieve lower communication complexity. On the other hand, in the online setting (4.3), only stochastic methods can be applied, and those methods again achieve $\mathcal{O}(m\epsilon^{-2})$ sample and $\mathcal{O}(\epsilon^{-2})$ communication complexity [77].

4.1.1 Our Contribution

Compared with the majority of the existing decentralized learning algorithms for either stochastic or deterministic problems, the focus of this work is given to *both* reducing the total communication and sample complexity. Specifically, we propose a decentralized

[1] Note that for the finite sum problem (4.2), the "sample complexity" refers to the total number of samples *accessed* by the algorithms to compute sample gradient $\nabla f_j^i(\mathbf{x}_i)$'s. If the same sample $j \in [n_i]$ is accessed k times and each time the evaluated gradients are different, then the sample complexity increases by k.

Table 4.1: Comparison of algorithms on decentralized non-convex optimization

Algorithm	Constant Stepsize	Finite-Sum	Online	Communication
DGD [85]	✗	$\mathcal{O}(mn\epsilon^{-2})$	✗	$\mathcal{O}(\epsilon^{-2})$
SONATA [76]	✓	$\mathcal{O}(mn\epsilon^{-1})$	✗	$\mathcal{O}(\epsilon^{-1})$
Prox-PDA [74]	✓	$\mathcal{O}(mn\epsilon^{-1})$	✗	$\mathcal{O}(\epsilon^{-1})$
xFILTER [3]	✓	$\mathcal{O}(mn\epsilon^{-1})$	✗	$\mathcal{O}(\epsilon^{-1})$
PSGD [68]	✗	$\mathcal{O}(m\epsilon^{-2})$	$\mathcal{O}(m\epsilon^{-2})$	$\mathcal{O}(\epsilon^{-2})$
D^2 [77]	✓	$\mathcal{O}(m\epsilon^{-2})$	$\mathcal{O}(m\epsilon^{-2})$	$\mathcal{O}(\epsilon^{-2})$
GNSD [79]	✓	$\mathcal{O}(m\epsilon^{-2})$	$\mathcal{O}(m\epsilon^{-2})$	$\mathcal{O}(\epsilon^{-2})$
D-GET (this work)	✓	$\mathcal{O}(m\sqrt{n}\epsilon^{-1})$	$\mathcal{O}(m\epsilon^{-3/2})$	$\mathcal{O}(\epsilon^{-1})$
Lower Bound [3, 80]		$\mathcal{O}(\sqrt{mn}\epsilon^{-1})$	-	$\mathcal{O}(\epsilon^{-1})$

gradient estimation and tracking (D-GET) approach, which uses a subset of samples to estimate the local gradients (by utilizing modern variance reduction techniques [80,81]), while using the differences of past local gradients to track the global gradients (by leveraging the idea of decentralized gradient tracking [75,82]). Remarkably, the proposed approach enjoys a sample complexity of $\mathcal{O}(mn^{1/2}\epsilon^{-1})$ and communication complexity of $\mathcal{O}(\epsilon^{-1})$ for finite sum problem (4.2), which outperforms all existing decentralized methods. The sample complexity rate is $\sqrt{m}$ worse than the known sample complexity lower bound for centralized problem [80], and the communication complexity matches the existing communication lower bound [3] for decentralized non-convex optimization (in terms of the dependency in ϵ). Furthermore, the proposed approach is also able to achieve $\mathcal{O}(m\epsilon^{-3/2})$ sample complexity and $\mathcal{O}(\epsilon^{-1})$ communication complexity for the online problem (4.3), reducing the best existing bounds (such as those obtained in [77,79,83]) by factors of $\mathcal{O}(\epsilon^{-1/2})$ and $\mathcal{O}(\epsilon^{-1})$, respectively, through a more restrictive mean-squared smoothness assumption [84]. The rate $\mathcal{O}(m\epsilon^{-3/2})$ is m worse than the centralized stochastic lower bound $\mathcal{O}(\epsilon^{-3/2})$ for non-convex problems [84]. We illustrate the main results of this work and compare the gradient and communication cost for state-of-the-art decentralized non-convex optimization algorithms in Table 4.1.[2] Note that in Table 4.1, by *constant step-size* we mean that it is not dependent on the target accuracy ϵ, nor the iteration number.

[2]For batch algorithms DGD, NEXT, Prox-PDA and xFILTER, the bounds are obtained by multiplying their convergence rates with $m \times n$, since when applied to solve finite-sum problems, each iteration requires $\mathcal{O}(1)$ full gradient evaluation.

4.1.2 Related Works

Decentralized Optimization. Decentralized optimization has been extensively studied for convex problems and can be traced back to the 1980s [86]. We refer the readers to the recent survey [87] and the references therein for a complete review. When the problem becomes non-convex, many algorithms such as primal-dual based methods [74, 88], gradient tracking based methods [75, 89], and non-convex extensions of DGD methods [85] have been proposed, where the $\mathcal{O}(\epsilon^{-1})$ iteration and communication complexity have been shown. Note that the above algorithms all require $\mathcal{O}(1)$ full gradient evaluations per iteration, so when directly applied to solve problems where each $f^i(\cdot)$ takes the form in (4.2), they all require $\mathcal{O}(mn\epsilon^{-1})$ local data samples.

However, due to the requirement that each iteration of the algorithm needs a full gradient evaluation, the above batch methods can be computationally very demanding. One natural solution is to use the stochastic gradient to approximate the true gradient. Stochastic decentralized non-convex methods can be traced back to [8, 90], and recent advances including DSGD [91], PSGD [68], D^2 [77], GNSD [79] and stochastic gradient push [78]. However, the large variance coming from the stochastic gradient estimator and the use of diminishing step-size slow down the convergence, resulting at least $\mathcal{O}(m\epsilon^{-2})$ sample and $\mathcal{O}(\epsilon^{-2})$ communication cost.

Variance Reduction. Consider the following non-convex finite sum problem:

$$\min_{\mathbf{w}\in\mathbb{R}^d} f(\mathbf{w}) = \frac{1}{mn}\sum_{j=1}^{mn} f_j(\mathbf{w}).$$

If we assume that $f(\cdot)$ has Lipschitz gradient, and directly apply the vanilla gradient descent (GD) method on $f(\mathbf{w})$, then it requires $\mathcal{O}(mn\epsilon^{-1})$ gradient evaluations to reach $\|\nabla f(\mathbf{w})\|^2 \leq \epsilon$ [92]. When $m \times n$ is large, it is usually preferable to process a subset of data each time. In this case, stochastic gradient descent (SGD) can be used to achieve an $\mathcal{O}(\epsilon^{-2})$ convergence rate [93]. To bridge the gap between the GD and SGD, many variance reduced gradient estimators have been proposed, including SAGA [94] and SVRG [95]. The idea is to reduce the variance of the stochastic gradient estimators and substantially improves the convergence rate. In particular, the above approaches have been shown to achieve sample complexities of $\mathcal{O}((mn)^{2/3}\epsilon^{-1})$ for finite sum problems [96–98] and $\mathcal{O}(\epsilon^{-5/3})$ for online problem [98]. Recent works further improve the

above gradient estimators and achieve $\mathcal{O}((mn)^{1/2}\epsilon^{-1})$ sample complexity for finite sum problems [80, 99–101] and $\mathcal{O}(\epsilon^{-3/2})$ sample complexity for online problems [80, 100]. At the same time, the $\mathcal{O}((mn)^{1/2}\epsilon^{-1})$ sample complexity is shown to be optimal when $m \times n \leq \mathcal{O}(\epsilon^{-2})$ [80].

Decentralized Variance Reduction. The variance reduced decentralized optimization has been extensively studied for convex problems. The DSA proposed in [102] combines the algorithm design ideas from EXTRA [103] and SAGA [94], and achieves the first expected linear convergence for decentralized stochastic optimization. Recent works also include the DSBA [104], diffusion-AVRG [105], ADFS [106], SAL-Edge [107], GT-SAGA [108], Network-DANE [109], and [110], just to name a few. However, when the problem becomes non-convex, to the best of our knowledge, no algorithms with provable guarantees are available.

4.2 The Finite Sum Setting

In this section, we consider the non-convex decentralized optimization problem (4.1) with finite number of samples as defined in (4.2), which is restated below:

$$\min_{\mathbf{x}\in\mathbb{R}^{md}} f(\mathbf{x}) = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} f_j^i(\mathbf{x}_i), \tag{P1}$$
$$\text{s.t.} \quad \mathbf{x}_i = \mathbf{x}_k, \quad \forall (i,k) \in \mathcal{E}.$$

We make the following standard assumptions on the above problem:

Assumption 1. *The objective function has Lipschitz continuous gradient with constant L:*

$$\|\nabla f^i(\mathbf{x}_i) - \nabla f^i(\mathbf{x}_i')\| \leq L\|\mathbf{x}_i - \mathbf{x}_i'\|, \forall i \tag{4.5}$$

while the component function has average Lipschitz continuous gradient with constant L:

$$\mathbb{E}_j\|\nabla f_j^i(\mathbf{x}_i) - \nabla f_j^i(\mathbf{x}_i')\| \leq L\|\mathbf{x}_i - \mathbf{x}_i'\|, \forall i \tag{4.6}$$

Assumption 2. *The mixing matrix $\mathbf{W}$ is symmetric, and satisfying the following*

$$|\lambda_{\max}(\mathbf{W})| := \eta < 1, \quad \mathbf{W}\mathbf{1} = \mathbf{1}, \tag{4.7}$$

where $\lambda_{\max}(\mathbf{W})$ *denotes the second largest eigenvalue of* $\mathbf{W} \in \mathbb{R}^{m\times m}$.[3]

Remark 4.2.1. Note that many choices of mixing matrices satisfy the condition in Assumption 2. Here we give three commonly used mixing matrices [111, 112], where d_i denotes the degree of node i, and $d_{\max} = \max_i\{d_i\}$:

- Metropolis-Hasting Weight

$$w_{ij} = \begin{cases} \frac{1}{1+\max\{d_i,d_j\}}, & \text{if} \quad \{i,j\} \in \mathcal{E}, \\ 1 - \sum_{\{i,k\}\in\mathcal{E}} w_{ik}, & \text{if} \quad i = j, \\ 0, & \text{otherwise.} \end{cases} \tag{4.8}$$

- Maximum-Degree Weight

$$w_{ij} = \begin{cases} \frac{1}{d_{\max}}, & \text{if} \quad \{i,j\} \in \mathcal{E}, \\ 1 - \frac{d_i}{d_{\max}}, & \text{if} \quad i = j, \\ 0, & \text{otherwise.} \end{cases} \tag{4.9}$$

- Laplacian Weight

$$w_{ij} = \begin{cases} \gamma & \text{if} \quad \{i,j\} \in \mathcal{E}, \\ 1 - \gamma d_i, , & \text{if} \quad i = j, \\ 0 & \text{otherwise.} \end{cases} \tag{4.10}$$

  If we use $\mathcal{L}$ to denote the graph Laplacian matrix, and $\lambda_{\max}, \lambda_{\min}$ as the largest and second smallest eigenvalue, then one of the common choices of γ is $\frac{2}{\lambda_{\max}(\mathcal{L})+\lambda_{\min}(\mathcal{L})}$.

Next, let us formally define our communication and sample complexity measures.

Definition 4.2.1. (Sample Complexity) *The Incremental First-order Oracle (IFO) is defined as an operation in which, one node* $i \in [m]$ *takes a data sample* $j \in [n]$*, a point*

[3]For notation simplicity when dealing with mixing matrix multiplication, but without loss of generality, we will assume that the optimization variable $\mathbf{x}_i$ in (4.1) is a scalar. The results can be extended to vector case via the Kronecker product.

$\mathbf{w} \in \mathbb{R}^d$, *and returns the pair* $(f_j^i(\mathbf{w}), \nabla f_j^i(\mathbf{w}))$. *The sample complexity is defined as the total number of IFO calls required across the entire network to achieve an* ϵ *stationary solution defined in* (4.4).

Definition 4.2.2. (Communication Complexity) *In one round of communication, each node* $i \in [m]$ *is allowed to broadcast and received one* d*-dimensional vector to and from its neighbors, respectively. Then the communication complexity is defined as the total rounds of communications required to achieve an* ϵ *stationary solution defined in* (4.4).

4.2.1 Algorithm Design

In this section, we introduce the proposed algorithm named Decentralized Gradient Estimation and Tracking (D-GET), for solving problem (P1). To motivate our algorithm design, we can observe from our discussion in Section 4.1.2 that, the existing deterministic decentralized methods typically suffer from the high sample complexity, while the decentralized stochastic algorithms suffer from the high communication cost. Such a phenomenon inspires us to find a solution in between, which could simultaneously reduce the sample and the communication costs.

One natural solution is to incorporate the modern variance reduction techniques into the classical decentralized methods. Our idea is to use some variance reduced gradient estimator to track the full gradient of the entire problem, then perform decentralized gradient descent update. The gradient tracking step gives us fast convergence with a constant step-size, while the variance reduction method significantly reduces the variation of the estimated gradient.

Unfortunately, the decentralized methods and variance reduction techniques cannot be directly combined. Compared with the existing decentralized and variance reduction techniques in the literature, the key challenges in the algorithm design and analysis are given below:

1) Due to the decentralized nature of the problem, none of the nodes can access the full gradient of the original objective function. The (possibly uncontrollable) consensus error always exists during the whole process of implementing the decentralized algorithm. Therefore, it is not clear that the existing variance reduction methods could

be applied at each individual node effectively, since all of those require accurate global gradient evaluation from time to time.

2) It is then natural to integrate some procedure that is able to approximate the global gradient. For example, one straightforward way to perform gradient tracking is to introduce a new auxiliary variable $\mathbf{y}$ as the following [75, 79], which is updated by only using local estimated gradient and neighbors' parameters:

$$\begin{aligned}\mathbf{y}_i^r = \sum_{k\in\mathcal{N}_i} \mathbf{W}_{ik}\mathbf{y}_k^{r-1} + \frac{1}{|S_2^r|}\sum_{j\in S_2^r} \nabla f_j^i(\mathbf{x}_i^r) \\ - \frac{1}{|S_2^{r-1}|}\sum_{j\in S_2^{r-1}} \nabla f_j^i(\mathbf{x}_i^{r-1}),\end{aligned} \tag{4.11}$$

where S_2^r and S_2^{r-1} are the samples selected at the r and $r-1$th iterations, respectively. If the tracked $\mathbf{y}_i$'s were used in the (local) variance reduction procedure, there would be at least two main issues of decreasing the variance resulted from the tracked gradient as follows: *i*) at the early stage of implementing the decentralized algorithm, the consensus/tracking error may dominate the variance of the tracked gradient, since the message of the full gradient has not been sufficiently propagated through the network. Consequently, performing variance reduction on $\mathbf{y}_i$'s will not be able to increase the quality of the full gradient estimation; *ii*) even assuming that there was no consensus error. Since only the stochastic gradients, i.e., $\sum_{j\in\mathcal{S}_2^r} \nabla f_j^i(\mathbf{x}_i^r)$, were used in the tracking, the $\mathbf{y}_i^r$'s themselves had high variance, resulting that such (possibly low-quality) full gradient estimates may not be compatible to variance reduction methods as developed in the current literature (which often require full gradient evaluation from time to time).

The challenges discussed above suggest that it is non-trivial to design an algorithm that can be implemented in a fully decentralized manner, while still achieving the superior sample complexity and convergence rate achieved by state-of-the-art variance reduction methods. In this work, we propose an algorithm which uses a novel decentralized gradient estimation and tracking strategy, together with a number of other design choices, to address the issues raised above.

To introduce the algorithm, let us first define two auxiliary local variables $\mathbf{v}_i$ and $\mathbf{y}_i$, where $\mathbf{v}_i$ is designed to estimate the local full batch gradient $\frac{1}{n}\sum_{j=1}^n \nabla f_j^i(\mathbf{x}_i)$ by only using sample gradient $\nabla f_j^i(\mathbf{x}_i)'s$, while $\mathbf{y}_i$ is designed to track the global average gradient $\frac{1}{mn}\sum_{i=1}^m\sum_{j=1}^n \nabla f_j^i(\mathbf{x}_i)$ by utilizing $\mathbf{v}_i$'s. After the local and global gradient

estimates are obtained, the algorithm performs local update based on the direction of $\mathbf{y}_i$; see the main steps below.

1) Local update using estimated gradient ($\mathbf{x}$ update): Each local node i first combines its previous iterates $\mathbf{x}_i^{r-1}$ with its local neighbors $\mathbf{x}_k^{r-1},\ k \in \mathcal{N}_i$ (by using the kth row of weight matrix $\mathbf{W}$), then makes a prediction based on the gradient estimate $\mathbf{y}_i^{r-1}$, i.e.,

$$\mathbf{x}_i^r = \sum_{k\in\mathcal{N}_i} \mathbf{W}_{ik}\mathbf{x}_k^{r-1} - \alpha \mathbf{y}_i^{r-1}. \tag{4.12}$$

2) Estimate local gradients ($\mathbf{v}$ update): Depending on the iteration r, each local node i either directly calculates the full local gradient $\nabla f^i(\mathbf{x}_i^r)$ when $\text{mod}(r, q) = 0$

$$\mathbf{v}_i^r = \nabla f^i(\mathbf{x}_i^r), \tag{4.13}$$

or estimates its local gradient via an estimator $\mathbf{v}$ using $|S_2|$ random samples otherwise,

$$\mathbf{v}_i^r = \frac{1}{|S_2|}\sum_{j\in S_2}\left[\nabla f_j^i(\mathbf{x}_i^r) - \nabla f_j^i(\mathbf{x}_i^{r-1})\right] + \mathbf{v}_i^{r-1}, \tag{4.14}$$

where $q > 0$ is the interval in which local full gradient will be evaluated once.

3) Track global gradients ($\mathbf{y}$ update): Each local node i combines its previous local estimate $\mathbf{y}_i^{r-1}$ with its local neighbors $\mathbf{y}_k^{r-1}, k \in \mathcal{N}_i$, then makes a new estimation based on the fresh information $\mathbf{v}_i^r$, i.e.,

$$\mathbf{y}_i^r = \sum_{k\in\mathcal{N}_i} \mathbf{W}_{ik}\mathbf{y}_k^{r-1} + \mathbf{v}_i^r - \mathbf{v}_i^{r-1}. \tag{4.15}$$

In the following table, we summarize the proposed algorithm in a more compact form. Note that we use $\mathbf{x} \in \mathbb{R}^{md}$, $\mathbf{v} \in \mathbb{R}^{md}$, $\mathbf{y} \in \mathbb{R}^{md}$, $\nabla f(\mathbf{x}) \in \mathbb{R}^{md}$ and $\nabla f_j(\mathbf{x}) \in \mathbb{R}^{md}$ to denote the concatenation of the $\mathbf{x}_i \in \mathbb{R}^d$, $\mathbf{v}_i \in \mathbb{R}^d$, $\mathbf{y}_i \in \mathbb{R}^d$, $\nabla f^i(\mathbf{x}_i) \in \mathbb{R}^d$ and $\nabla f_j^i(\mathbf{x}_i) \in \mathbb{R}^d$ across all nodes.

Remark 4.2.2. This is a "double loop" algorithm, where each outer iteration (i.e., mod $(r, q) = 0$) is followed by $q-1$ inner iterations (i.e., mod $(r, q) \neq 0$). The inner loop estimates the local gradient via stochastic sampling at every iteration r, while the outer loop aims to reduce the estimation variance by recalculating the full batch gradient at every q iterations. The local communication, update, and tracking steps are performed

Input: $\mathbf{x}^0, \alpha, q, |S_2|$
$\mathbf{v}^0 = \nabla f(\mathbf{x}^0)$, $\mathbf{y}^0 = \nabla f(\mathbf{x}^0)$
for $r = 1, 2, \ldots, T$ **do**
  $\mathbf{x}^r = \mathbf{W}\mathbf{x}^{r-1} - \alpha \mathbf{y}^{r-1}$
  if $\mathrm{mod}(r, q) = 0$ **then**
    Calculate the full gradient
    $\mathbf{v}^r = \nabla f(\mathbf{x}^r)$
  else
    Each node draws S_2 samples from $[n]$ with replacement
    $\mathbf{v}^r = \frac{1}{|S_2|} \sum_{j \in S_2} \left[\nabla f_j(\mathbf{x}^r) - \nabla f_j(\mathbf{x}^{r-1}) \right] + \mathbf{v}^{r-1}$
  end if
  $\mathbf{y}^r = \mathbf{W}\mathbf{y}^{r-1} + \mathbf{v}^r - \mathbf{v}^{r-1}$
end for
Output: $\mathbf{x}^R$ where $R \in [0, T]$ is the uniformly distributed random variable.

Algorithm 3: D-GET Algorithm for finite sum problem (P1)

at both inner and outer iterations.

Remark 4.2.3. In D-GET, the total communication rounds is in the same order as the total number of iterations, since only two rounds of communications are performed per iteration, via broadcasting the local variable $\mathbf{x}_i^{r-1}$ and $\mathbf{y}_i^{r-1}$ to their neighbors, and combining local $\mathbf{x}_k^{r-1}$ and $\mathbf{y}_k^{r-1}$'s, $k \in \mathcal{N}_i$. On the other hand, the total number of samples used per iteration is either $m|S_2|$ (where inner iterations are executed) or mn (where outer iterations are executed).

Remark 4.2.4. Note that our $\mathbf{x}$ and $\mathbf{y}$ updates are reminiscent of the classical gradient tracking methods [75], and $\mathbf{v}$ update takes a similar form as the SARAH/SPIDER estimator [80, 81]. However, it is non-trivial to directly combine the gradient tracking and the variance reduction together, as we mentioned at the beginning of Section 4.2.1. The proposed D-GET uses a number of design choices to address these challenges. For example, two vectors $\mathbf{v}$ and $\mathbf{y}$ are used to respectively estimate the local and global gradients, in a way that the local gradient estimates do not depend on the (potentially inaccurate) global tracked gradients; to reduce the variance in $\mathbf{y}$, we occasionally use the full local gradient to perform tracking, etc. Nevertheless, the key challenge in the

analysis is to properly bound the accumulated errors from the two estimates $\mathbf{v}$ and $\mathbf{y}$.

4.2.2 Convergence Analysis

To facilitate our analysis, we first define the average iterates $\bar{\mathbf{x}}$ and $\bar{\mathbf{y}}$ among all m nodes,

$$\bar{\mathbf{x}}^r := \frac{1}{m}\sum_i \mathbf{x}_i^r, \quad \bar{\mathbf{v}}^r := \frac{1}{m}\sum_i \mathbf{v}_i^r, \quad \bar{\mathbf{y}}^r := \frac{1}{m}\sum_i \mathbf{y}_i^r. \tag{4.16a}$$

We use r to denote the overall iteration number. The total number of outer iterations until iteration r as below:

$$n_r := \lfloor r/q \rfloor + 1, \text{ such that } (n_r - 1)q \leq r \leq n_r q - 1.$$

Next, we outline the proof steps of the convergence rate analysis.

Step 1. We first show that the variance of our local and global gradient estimators can be bounded via $\mathbf{x}$ and $\mathbf{y}$ iterates. The bounds to be given below is tighter than the classical analysis of decentralized stochastic methods, which assume the variance are bounded by some universal constant [68, 77, 91]. This is an important step to obtain lower sample/communication complexity, since later we can show that the right-hand-side (RHS) of our bound shrinks as the iteration progresses.

Lemma 4.2.1. *(Bounded Variance) Under Assumption 1 - 2, the sequence generated by the inner loop of Algorithm 3 satisfies the following inequalities (for all $(n_r - 1)q \leq r \leq n_r q - 1$):*

$$\begin{aligned} &\mathbb{E}\|\bar{\mathbf{y}}^r - \frac{1}{m}\sum_{i=1}^m \nabla f^i(\mathbf{x}_i^r)\|^2 \\ \leq &\frac{8L^2}{m|S_2|}\sum_{t=(n_r-1)q}^{r} \mathbb{E}\|\mathbf{x}^t - \mathbf{1}\bar{\mathbf{x}}^t\|^2 + \frac{4\alpha^2 L^2}{m|S_2|}\sum_{t=(n_r-1)q}^{r} \mathbb{E}\|\mathbf{y}^t - \mathbf{1}\bar{\mathbf{y}}^t\|^2 \\ + &\frac{4\alpha^2 L^2}{|S_2|}\sum_{t=(n_r-1)q}^{r} \mathbb{E}\|\bar{\mathbf{y}}^t\|^2 + \mathbb{E}\|\bar{\mathbf{y}}^{(n_r-1)q} - \frac{1}{m}\sum_{i=1}^m \nabla f^i(\mathbf{x}_i{}^{(n_r-1)q})\|^2. \end{aligned} \tag{4.17}$$

$$\mathbb{E}\|\mathbf{v}^r - \nabla f(\mathbf{x}^r)\|^2 \leq \frac{L^2}{|S_2|} \sum_{t=(n_r-1)q}^{r} \mathbb{E}\|\mathbf{x}^{t+1} - \mathbf{x}^t\|^2 + \mathbb{E}\|\mathbf{v}^{(n_r-1)q} - \nabla f(\mathbf{x}^{(n_r-1)q})\|^2. \tag{4.18}$$

Step 2. We then study the descent on $\mathbb{E}[f(\bar{\mathbf{x}}^r)]$, which is the expected value of the cost function evaluated on the average iterates.

Lemma 4.2.2. *(Descent Lemma) Suppose Assumptions 1 - 2 hold, and for any $r \geq 0$ satisfying mod$(r, q) = 0$, the following holds for some $\epsilon_1 \geq 0$:*

$$\mathbb{E}\left[\|\bar{\mathbf{y}}^r - \frac{1}{mn}\sum_{i=1}^{m}\sum_{k=1}^{n}\nabla f_k^i(\mathbf{x}_i^r)\|^2\right] \leq \epsilon_1. \tag{4.19}$$

Algorithm 3 ensures the following relation for all $r \geq 0$,

$$\begin{aligned}\mathbb{E}[f(\bar{\mathbf{x}}^{r+1})] \leq\ & \mathbb{E}[f(\bar{\mathbf{x}}^0)] \\ & - \left(\frac{\alpha}{2} - \frac{\alpha^2 L}{2} - \frac{4\alpha^3 L^2 q}{|S_2|}\right)\sum_{t=0}^{r}\mathbb{E}\|\bar{\mathbf{y}}^t\|^2 \\ & + \left(\frac{\alpha L^2}{m} + \frac{8\alpha L^2 q}{m|S_2|}\right)\sum_{t=0}^{r}\mathbb{E}\|\mathbf{x}^t - \mathbf{1}\bar{\mathbf{x}}^t\|^2 \\ & + \frac{4\alpha^3 L^2 q}{m|S_2|}\sum_{t=0}^{r}\mathbb{E}\|\mathbf{y}^t - \mathbf{1}\bar{\mathbf{y}}^t\|^2 + \alpha(r+1)\epsilon_1.\end{aligned} \tag{4.20}$$

A key observation from Lemma 4.2.2 is that, in the RHS of (4.20), besides the negative term in $\mathbb{E}\|\bar{\mathbf{y}}^k\|^2$, we also have several extra terms (such as $\mathbb{E}\|\mathbf{x}^k - \mathbf{1}\bar{\mathbf{x}}^k\|^2$ and $\mathbb{E}\|\mathbf{y}^k - \mathbf{1}\bar{\mathbf{y}}^k\|^2$) that cannot be made negative. Therefore, we need to find some potential function that is strictly descending per iteration.

Note that ϵ_1 in (4.19) comes from the variance of $\mathbf{v}^r$ in estimating the full local gradient at each outer loop n_r. For Algorithm 3, where we calculate a full batch gradient per outer loop in step (4.37), $\epsilon_1 = 0$. However, we still would like to include ϵ_1 in the above result because, later when we analyze the online version (where such a variance will no longer be zero), we can re-use the above result.

Step 3. Next, we introduce the contraction property, which combined with $\mathbb{E}[f(\bar{\mathbf{x}}^r)]$ will be used to construct the potential function.

Lemma 4.2.3. *(Iterates Contraction) Using the Assumption 2 on* $\mathbf{W}$ *and applying Algorithm 3, we have the following contraction property of the iterates:*

$$\begin{aligned}
&\mathbb{E}\|\mathbf{x}^{r+1} - \mathbf{1}\bar{\mathbf{x}}^{r+1}\|^2 \qquad (4.21)\\
\leq&(1+\beta)\eta^2\mathbb{E}\|\mathbf{x}^r - \mathbf{1}\bar{\mathbf{x}}^r\|^2 + (1+\frac{1}{\beta})\alpha^2\mathbb{E}\|\mathbf{y}^r - \mathbf{1}\bar{\mathbf{y}}^r\|^2,\\
&\mathbb{E}\|\mathbf{y}^{r+1} - \mathbf{1}\bar{\mathbf{y}}^{r+1}\|^2 \qquad (4.22)\\
\leq&(1+\beta)\eta^2\mathbb{E}\|\mathbf{y}^r - \mathbf{1}\bar{\mathbf{y}}^r\|^2 + (1+\frac{1}{\beta})\mathbb{E}\|\mathbf{v}^{r+1} - \mathbf{v}^r\|^2,
\end{aligned}$$

where β *is some constant such that* $(1+\beta)\eta^2 < 1$.

If we further assume for all $r \geq 0$ *satisfying* $mod(r, q) = 0$*, the following holds for some* $\epsilon_2 \geq 0$*:*

$$\mathbb{E}\|\mathbf{v}^r - \nabla f(\mathbf{x}^r)\|^2 \leq \epsilon_2. \qquad (4.23)$$

Then we have the following bound:

$$\begin{aligned}
&\sum_{t=0}^{r}\mathbb{E}\|\mathbf{v}^{t+1} - \mathbf{v}^t\|^2 \leq 48L^2\sum_{t=0}^{r}\mathbb{E}\|\mathbf{x}^t - \mathbf{1}\bar{\mathbf{x}}^t\|^2\\
&+ 24L^2\alpha^2\sum_{t=0}^{r}\mathbb{E}\|\mathbf{y}^t - \mathbf{1}\bar{\mathbf{y}}^t\|^2\\
&+ 24L^2\alpha^2\sum_{t=0}^{r}\mathbb{E}\|\mathbf{1}\bar{\mathbf{y}}^t\|^2 + 6(r+1)\epsilon_2,\ \forall\ r \geq 0. \qquad (4.24)
\end{aligned}$$

Again, ϵ_2 comes from the variance of the estimating the local gradient in each outer loop, and we have $\epsilon_2 = 0$ for Algorithm 3. Note that (4.21) can also be written as following

$$\begin{aligned}
&\mathbb{E}\|\mathbf{x}^{r+1} - \mathbf{1}\bar{\mathbf{x}}^{r+1}\|^2 - \mathbb{E}\|\mathbf{x}^r - \mathbf{1}\bar{\mathbf{x}}^r\|^2\\
\leq&\left((1+\beta)\eta^2 - 1\right)\mathbb{E}\|\mathbf{x}^r - \mathbf{1}\bar{\mathbf{x}}^r\|^2\\
&+ (1+\frac{1}{\beta})\alpha^2\mathbb{E}\|\mathbf{y}^r - \mathbf{1}\bar{\mathbf{y}}^r\|^2. \qquad (4.25)
\end{aligned}$$

One key observation here is that we have $(1+\beta)\eta^2 - 1 < 0$ by properly choosing β.

Therefore, the RHS of the above equation can be made negative by properly selecting the step-size α.

Step 4. This step combines the descent estimates obtained in Step 2-3 to construct a potential function, by using a conic combination of $\mathbb{E}[f(\bar{\mathbf{x}}^r)]$, $\mathbb{E}\|\mathbf{x}^r - \mathbf{1}\bar{\mathbf{x}}^r\|^2$ and $\mathbb{E}\|\mathbf{y}^r - \mathbf{1}\bar{\mathbf{y}}^r\|^2$.

Lemma 4.2.4. *(Potential Function) Constructing the following potential function*

$$H(\mathbf{x}^r) := \mathbb{E}[f(\bar{\mathbf{x}}^r)] + \frac{1}{m}\mathbb{E}\|\mathbf{x}^r - \mathbf{1}\bar{\mathbf{x}}^r\|^2 + \frac{\alpha}{m}\mathbb{E}\|\mathbf{y}^r - \mathbf{1}\bar{\mathbf{y}}^r\|^2.$$

Under Assumption 1 - 2 and Algorithm 3, if we further pick $q = |S_2|$ and define ϵ_1 and ϵ_2 as in (4.19) *and* (4.23)*, we have*

$$\begin{aligned} & H(\mathbf{x}^{r+1}) - H(\mathbf{x}^0) \\ \leq & - C_1 \sum_{t=0}^{r} \mathbb{E}\|\bar{\mathbf{y}}^t\|^2 - C_2 \sum_{t=0}^{r} \frac{1}{m}\mathbb{E}\|\mathbf{x}^t - \mathbf{1}\bar{\mathbf{x}}^t\|^2 \\ & - C_3 \sum_{t=0}^{r} \frac{1}{m}\mathbb{E}\|\mathbf{y}^t - \mathbf{1}\bar{\mathbf{y}}^t\|^2 + \epsilon_3, \end{aligned}$$

where

$$C_1 := \left(\frac{1}{2} - \frac{\alpha L}{2} - 4\alpha^2 L^2 - 24(1+\frac{1}{\beta})\alpha^2 L^2\right)\alpha, \tag{4.26a}$$

$$C_2 := \left(1 - (1+\beta)\eta^2 - 48\alpha(1+\frac{1}{\beta})L^2 - 9\alpha L^2\right), \tag{4.26b}$$

$$\begin{aligned} C_3 :=& \alpha - (1+\beta)\alpha\eta^2 - (1+\frac{1}{\beta})\alpha^2 \\ & - 24(1+\frac{1}{\beta})\alpha^3 L^2 - 4\alpha^3 L^2, \end{aligned} \tag{4.26c}$$

$$\epsilon_3 := \alpha(r+1)(\epsilon_1 + 6\frac{1}{m}(1+\frac{1}{\beta})\epsilon_2). \tag{4.26d}$$

Step 5. We can then properly choose the step-size α, and make C_1, C_2, C_3 to be positive. Therefore, our solution quality measure $\mathbb{E}\|\frac{1}{m}\sum_{i=1}^m \nabla f^i(\mathbf{x}_i^r)\|^2 + \frac{1}{m}\mathbb{E}\|\mathbf{x}^r - \mathbf{1}\bar{\mathbf{x}}^r\|^2$ can be expressed as the difference of the potential functions and the proof is complete.

Theorem 4.2.1. *Consider problem* (P1) *and under Assumption 1 - 2, if we pick*

$\alpha = \min\{K_1, K_2, K_3\}$ and $q = |S_2| = \sqrt{n}$, then we have following results by applying Algorithm 3,

$$\begin{aligned}&\frac{1}{T}\sum_{t=0}^{T}\mathbb{E}\|\frac{1}{m}\sum_{i=1}^{m}\nabla f^i(\mathbf{x}_i^t)\|^2+\frac{1}{T}\sum_{t=0}^{T}\frac{1}{m}\mathbb{E}\|\mathbf{x}^t-\mathbf{1}\bar{\mathbf{x}}^t\|^2\\ \leq& C_0\cdot\frac{\mathbb{E}f(\mathbf{x}^0)-\underline{f}}{T},\end{aligned}$$

where $\underline{f}$ denotes the lower bound of f, and the constants are defined as following

$$\begin{aligned}K_1:=&\frac{-\frac{L}{2}+\sqrt{(\frac{L}{2})^2+48(1+\frac{1}{\beta})L^2+8L^2}}{48(1+\frac{1}{\beta})L^2+8L^2},\\ K_2:=&\frac{1-(1+\beta)\eta^2}{48(1+\frac{1}{\beta})L^2+9L^2},\\ K_3:=&\frac{-(1+\frac{1}{\beta})}{48(1+\frac{1}{\beta})L^2+8L^2}\\ &+\frac{\sqrt{(1+\frac{1}{\beta})^2+4(1-(1+\beta)\eta^2)(24(1+\frac{1}{\beta})+4L^2)}}{48(1+\frac{1}{\beta})L^2+8L^2},\\ C_0:=&\left(\frac{8\alpha^2L^2+2}{C_1}+\frac{16L^2+1}{mC_2}+\frac{8\alpha^2L^2}{mC_3}\right),\end{aligned}$$

in which η denotes the second largest eigenvalue of the mixing matrix from (4.7)*, β denotes a constant satisfying $1-(1+\beta)\eta^2>0$, for example, $\beta=(1-\eta^2)/(2\eta^2)$, and C_1, C_2, C_3 are defined in* (4.26a)-(4.26c)*.*

By directly applying the above result, we have the upper bound on gradient and communication cost by properly choosing T based on ϵ.

Corollary 1. *To achieve the following ϵ stationary solution of problem* (P1) *by Algorithm 3,*

$$\frac{1}{T}\sum_{t=0}^{T}\mathbb{E}\|\frac{1}{m}\sum_{i=1}^{m}\nabla f^i(\mathbf{x}_i^t)\|^2+\frac{1}{T}\sum_{t=0}^{T}\frac{1}{m}\mathbb{E}\|\mathbf{x}^t-\mathbf{1}\bar{\mathbf{x}}^t\|^2\leq\epsilon,$$

the total number of iterations T and communication rounds required are both in the order of $\mathcal{O}(\epsilon^{-1})$, and the total number of samples evaluated across the network is in the

order of $\mathcal{O}(mn + mn^{1/2}\epsilon^{-1})$.

Note that our metric is evaluated on individual variable $\mathbf{x}_i$ and our algorithm also output $\mathbf{x}_i$ as each agent i's final solution. If we choose to use the average $\bar{\mathbf{x}} = \frac{1}{m}\sum \mathbf{x}_i$ as our final solution, one can show that the metric on average iterates $\bar{\mathbf{x}}$ is also small through simple derivations.

Corollary 2. *To achieve the following* ϵ *stationary solution of problem* (P1) *by Algorithm 3,*

$$\frac{1}{T}\sum_{t=0}^{T}\mathbb{E}\|\frac{1}{m}\sum_{i=1}^{m}\nabla f^i(\bar{\mathbf{x}}^t)\|^2 + \frac{1}{T}\sum_{t=0}^{T}\frac{1}{m}\mathbb{E}\|\mathbf{x}^t - \mathbf{1}\bar{\mathbf{x}}^t\|^2 \leq \epsilon,$$

the total number of iterations T *and communication rounds required are both in the order of* $\mathcal{O}(\epsilon^{-1})$*, and the total number of samples evaluated across the network is in the order of* $\mathcal{O}(mn + mn^{1/2}\epsilon^{-1})$.

If we further take expectation over the iteration t, we can have the convergence guarantee on our algorithm output $\mathbf{x}_i^R$ (or $\bar{\mathbf{x}}^R$ similarly), where $R \in [0, T]$ is the uniformly distributed random variable.

Corollary 3. *To achieve the following* ϵ *stationary solution of problem* (P1) *by Algorithm 3,*

$$\mathbb{E}\|\frac{1}{m}\sum_{i=1}^{m}\nabla f^i(\mathbf{x}_i^R)\|^2 + \frac{1}{m}\mathbb{E}\|\mathbf{x}^R - \mathbf{1}\bar{\mathbf{x}}^R\|^2 \leq \epsilon,$$

the total number of iterations T *and communication rounds required are both in the order of* $\mathcal{O}(\epsilon^{-1})$*, and the total number of samples evaluated across the network is in the order of* $\mathcal{O}(mn + mn^{1/2}\epsilon^{-1})$*. The expectation* $\mathbb{E}$ *here is taken over the iteration* R *and the randomness from the random sampling step* (4.14).

4.3 The Online Setting

In this section, we discuss the online setting (4.3) for solving problem (4.1), where the problem can either be expressed as the following

$$\min_{\mathbf{x}\in\mathbb{R}^{md}} f(\mathbf{x}) = \frac{1}{m}\sum_{i=1}^{m} \mathbb{E}_{\xi\sim\mathcal{D}_i}\left[f_\xi^i(\mathbf{x}_i)\right], \tag{P2}$$
$$\text{s.t.} \quad \mathbf{x}_i = \mathbf{x}_j, \quad \forall (i,j)\in\mathcal{E},$$

where ξ represents the data drawn from the data distribution $\mathcal{D}_i$ at the ith node, or in form (P1) such that the number of samples n is too large to calculate the full batch even occasionally. In either one of these scenarios, full batch evaluations at the local nodes are no longer performed for each outer iteration.

The above setting has been well-studied for the centralized problem (with large or even infinite number of samples). For example, in SCSG [98], a batch size $\mathcal{O}(\epsilon^{-1})$ is used when the sample size is large or the target accuracy $\mathcal{O}(\epsilon)$ is moderate, improving the rate to $\mathcal{O}(\epsilon^{-5/3})$ from $\mathcal{O}(\epsilon^{-2})$ compared to the vanilla SGD [93]. Recently, SPIDER [80] further improves the results to $\mathcal{O}(\epsilon^{-3/2})$, while the SpiderBoost [100] uses a constant step-size and is amenable to solve non-smooth problem at this rate.

4.3.1 The Proposed Algorithm

To begin with, we first introduce two additional commonly used assumptions in the online learning setting, together with our Assumption 1 and 2.

Assumption 3. *At each iteration, samples are independently collected, and the stochastic gradient is an unbiased estimate of the true gradient:*

$$\mathbb{E}_\xi[\nabla f_\xi^i(\mathbf{x}_i)] = \nabla f^i(\mathbf{x}_i), \forall i. \tag{4.27}$$

Assumption 4. *The variance between the stochastic gradient and the true gradient is bounded:*

$$\mathbb{E}_\xi[\|\nabla f_\xi^i(\mathbf{x}_i) - \nabla f^i(\mathbf{x}_i)\|^2] \le \sigma^2, \forall i. \tag{4.28}$$

To present our algorithms, note that compared to problem (P1), the main difference of having the expectation in (P2) is that the full batch gradient evaluation is no longer feasible. Therefore, we need to slightly revise our algorithm in Section 4.2 and redesign the local gradient estimation step (i.e., the $\mathbf{v}$ update). Specifically, different from (4.13) where we sample the full batch, here we randomly draw S_1 samples, the size of which is inversely proportional to the desired accuracy ϵ. We have the following updates on $\mathbf{v}$:

Depending on the iteration r, each local node i either estimates its local gradient using $|S_1|$ random samples when $\mathrm{mod}(r,q)=0$,

$$\mathbf{v}_i^r = \frac{1}{|S_1|}\sum_{\xi\in S_1}\nabla f_\xi^i(\mathbf{x}_i^r), \tag{4.29}$$

or uses $|S_2|$ random samples otherwise,

$$\mathbf{v}_i^r = \frac{1}{|S_2|}\sum_{\xi\in S_2}\left[\nabla f_\xi^i(\mathbf{x}_i^r) - \nabla f_\xi^i(\mathbf{x}_i^{r-1})\right] + \mathbf{v}_i^{r-1}. \tag{4.30}$$

It is easy to check that the following relation on average iterates is obvious when $\mathrm{mod}(r,q)=0$ and $\bar{\mathbf{y}}^0=\bar{\mathbf{v}}^0$,

$$\bar{\mathbf{y}}^r = \bar{\mathbf{v}}^r = \frac{1}{m|S_1|}\sum_{i=1}^{m}\sum_{\xi\in S_1}\nabla f_\xi^i(\mathbf{x}_i^r). \tag{4.31}$$

The rest of the updates on $\mathbf{x}$ and $\mathbf{y}$ are same as the finite sum setting; see Algorithm 4 below for details.

Input: $\mathbf{x}^0, \alpha, q, |S_1|, |S_2|$
Draw S_1 samples with replacement
$\mathbf{v}^0 = \frac{1}{|S_1|}\sum_{\xi\in S_1}\nabla f_\xi(\mathbf{x}^0)$, $\mathbf{y}^0=\mathbf{v}^0$
for $r=1,2,\ldots$ **do**
  $\mathbf{x}^r = \mathbf{W}\mathbf{x}^{r-1} - \alpha\mathbf{y}^{r-1}$
  if $\mathrm{mod}(r,q)=0$ **then**
    Draw S_1 samples with replacement
    $\mathbf{v}^r = \frac{1}{|S_1|}\sum_{\xi\in S_1}\nabla f_\xi(\mathbf{x}^r)$
  else
    Draw S_2 samples with replacement
    $\mathbf{v}^r = \frac{1}{|S_2|}\sum_{\xi\in S_2}\left[\nabla f_\xi(\mathbf{x}^r) - \nabla f_\xi(\mathbf{x}^{r-1})\right] + \mathbf{v}^{r-1}$
  end if
  $\mathbf{y}^r = \mathbf{W}\mathbf{y}^{r-1} + \mathbf{v}^r - \mathbf{v}^{r-1}$
end for
Output: $\mathbf{x}^R$ where $R\in[0,T]$ is the uniformly distributed random variable.

Algorithm 4: D-GET Algorithm (global view) (online)

4.3.2 Convergence Analysis

The analysis follows the same steps as described in Section 4.2.2 and it is easy to verify that our Lemma 4.2.1 to Lemma 4.2.4 still hold true for Algorithm 4. However, for online setting where we no longer sample a full batch, the variance ϵ_1 and ϵ_2 cannot be eliminated. The lemma given below provides the bounds on ϵ_1 and ϵ_2.

Lemma 4.3.1. *(Bounded Variance) Under Assumption 1 to 4, the sequence generated by the outer loop of Algorithm 4 satisfies the following relations (for all r such that $mod(r, q)=0$)*

$$\mathbb{E}\|\mathbf{v}^r - \nabla f(\mathbf{x}^r)\|^2 \leq \frac{m\sigma^2}{|S_1|},$$
$$\mathbb{E}\|\bar{\mathbf{y}}^r - \frac{1}{m}\sum_{i=1}^{m}\nabla f^i(\mathbf{x}_i^r)\|^2 \leq \frac{\sigma^2}{|S_1|}.$$

By using the above lemma, we can then choose the sample size inversely proportional to the targeted accuracy and obtain our final results.

Theorem 4.3.1. *Suppose Assumption 1 - 4 hold, and pick the following parameters for problem* (P2)*:*

$$\alpha = \min\{K_1, K_2, K_3\}, \quad q = |S_2| = \sqrt{|S_1|},$$
$$|S_1| = (4C_0\alpha(7 + \frac{6}{\beta})\sigma^2 + 8\sigma^2)/\epsilon.$$

Then we have the following result by applying Algorithm 4,

$$\frac{1}{T}\sum_{t=0}^{T}\mathbb{E}\|\frac{1}{m}\sum_{i=1}^{m}\nabla f^i(\mathbf{x}_i^t)\|^2 + \frac{1}{T}\sum_{t=0}^{T}\frac{1}{m}\mathbb{E}\|\mathbf{x}^t - \mathbf{1}\bar{\mathbf{x}}^t\|^2$$
$$\leq C_0 \cdot \frac{\mathbb{E}f(\mathbf{x}^0) - \underline{f}}{T} + \frac{\epsilon}{2}.$$

Corollary 4. *By using Algorithm 4, to achieve the ϵ stationary solution of problem* (4.1)*, i.e.,*

$$\frac{1}{T}\sum_{t=0}^{T}\mathbb{E}\|\frac{1}{m}\sum_{i=1}^{m}\nabla f^i(\mathbf{x}_i^t)\|^2 + \frac{1}{T}\sum_{t=0}^{T}\frac{1}{m}\mathbb{E}\|\mathbf{x}^t - \mathbf{1}\bar{\mathbf{x}}^t\|^2 \leq \epsilon,$$

the total number of iterations T and communication rounds required are both in the

order of $\mathcal{O}(\epsilon^{-1})$, and the total sample complexity is in the order of $\mathcal{O}(m\epsilon^{-3/2})$.

4.4 Experimental Results

In this section, we demonstrate the performance of the proposed algorithms on two classical smooth non-convex problems: a) decentralized logistic regression with non-convex regularizer and b) non-convex robust linear regression, the detailed objective functions are stated as the following:

a) Decentralized logistic regression with non-convex regularizer

The problem is stated as follows

$$\min_{\mathbf{x}\in\mathbb{R}^{nd}} f(\mathbf{x}) := \frac{1}{m}\sum_{i=1}^{m}\left[\frac{1}{n}\sum_{j=1}^{n}\ell(\mathbf{x}_i^\top \mathbf{v}_j^i, \mathbf{y}_j^i) + \sum_{k=1}^{d}\frac{\alpha \mathbf{x}_{i,k}^2}{1+\mathbf{x}_{i,k}^2}\right], \tag{4.32}$$

where $\mathbf{v}_j^i \in \mathbb{R}^d, \forall i, j$ denote the features of node i and sample j, $\mathbf{y}_j^i \in \{+1, -1\}, \forall i$ denote the labels of node i and sample j, and the loss function is defined as the cross-entropy loss

$$\ell(\mathbf{x}_i^\top \mathbf{v}_j^i, \mathbf{y}_j^i) := -\mathbf{y}_j^i \log\left(\frac{1}{1+e^{-\mathbf{x}_i^\top \mathbf{v}_j^i}}\right) - \left(1-\mathbf{y}_j^i\right)\log\left(1-\frac{1}{1+e^{-\mathbf{x}_i^\top \mathbf{v}_j^i}}\right). \tag{4.33}$$

The regularization parameter for non-convex term is set to $\alpha = 0.01$ in our simulation.

b) Non-convex robust linear regression

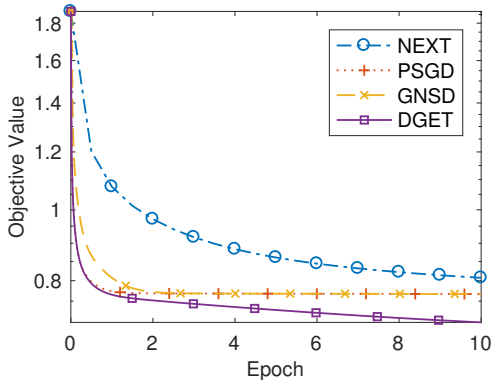
The problem is stated as follows

$$\min_{\mathbf{x}\in\mathbb{R}^{nd}} f(\mathbf{x}) := \frac{1}{m}\sum_{i=1}^{m}\frac{1}{n}\sum_{j=1}^{n}\ell(\mathbf{y}_j^i - \mathbf{x}_i^T \mathbf{v}_j^i), \tag{4.34}$$

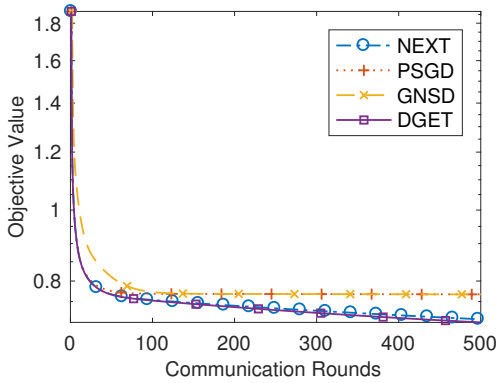
where $\mathbf{v}_j^i \in \mathbb{R}^d$ denotes the features of node i and sample j, $\mathbf{y}_i \in \{+1, -1\}$ denotes the labels of node i and sample j, and the loss function is defined as the following

$$\ell(u) := \log\left(\frac{u^2}{2}+1\right). \tag{4.35}$$

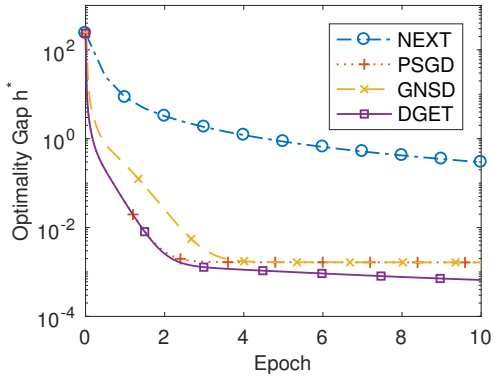
Next, we provide experimental results on the above mentioned decentralized logistic regression and robust linear regression problems. In particular, we demonstrate the



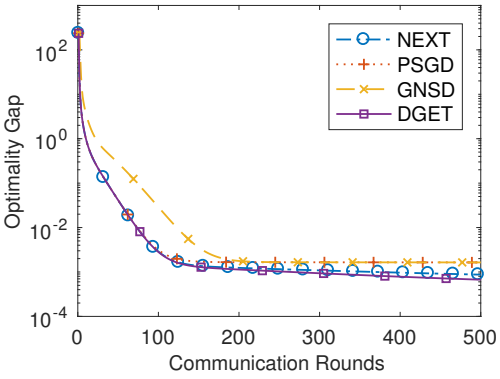
(a) Loss Function v.s. Sample Complexity



(b) Loss Function v.s. Communication Complexity



(c) Optimality Gap v.s. Sample Complexity

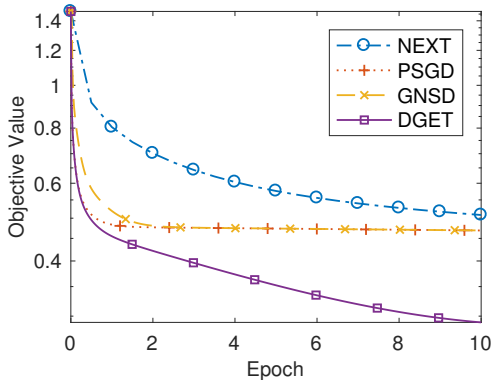


(d) Optimality Gap v.s. Communication Complexity

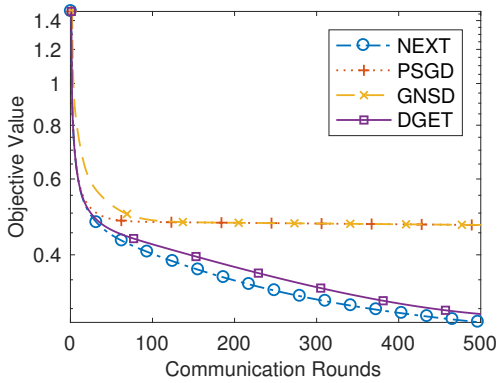
Figure 4.1: Logistic regression on the a9a dataset over the path graph

performance of the algorithms in terms of the loss function as stated in (4.32) and (4.34) and the optimality gap defined in (4.4). For all algorithms considered, we set their learning rates to be 0.01. For each experiment, we initialize all the algorithms at the same point generated randomly from the normal distribution. Also, we choose a fixed mini-batch size 64 and set the epoch length q to be $n/64$ such that all algorithms pass over the entire dataset once in each epoch.

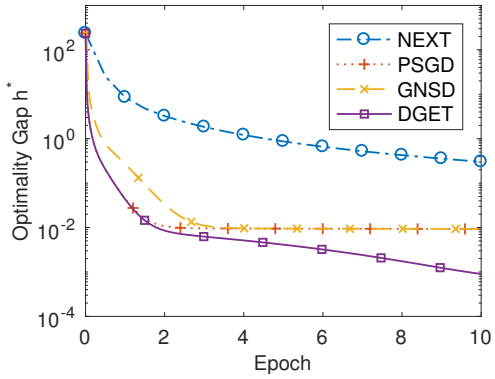
The simulation results in terms of both sample complexity and the communication complexity averaged over 10 realizations on the a9a dataset ($n = 32561, d = 123$) are shown in Figure 4.1 and Figure 4.2, and the performance on the w8a dataset



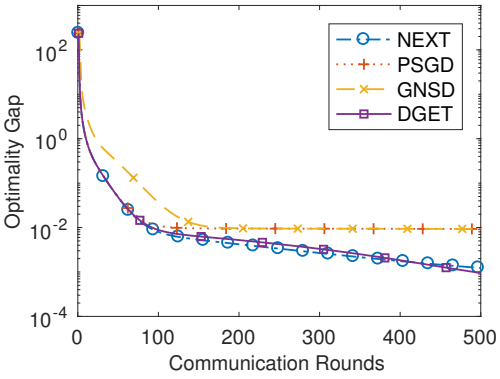
(a) Loss Function v.s. Sample Complexity



(b) Loss Function v.s. Communication Complexity



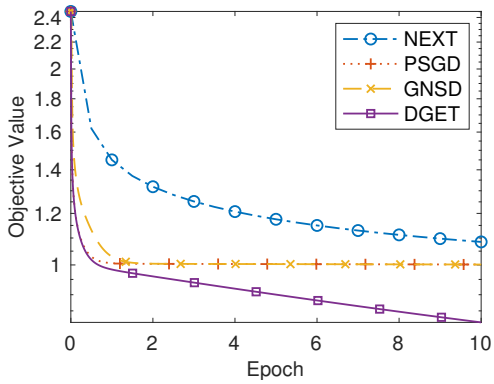
(c) Optimality Gap v.s. Sample Complexity



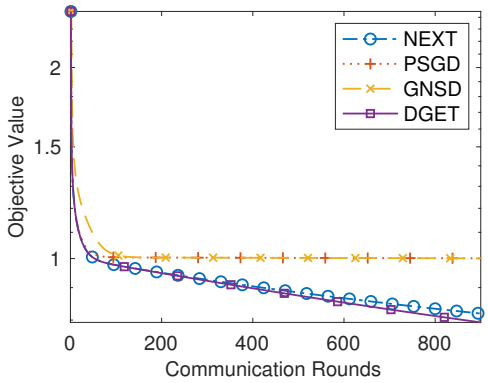
(d) Optimality Gap v.s. Communication Complexity

Figure 4.2: Robust linear regression on the a9a datatset over the path graph

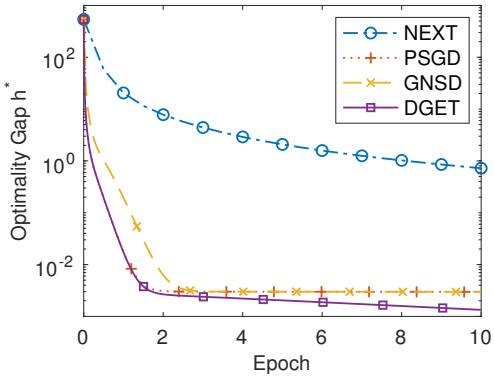
($n = 49749, d = 300$) are reported in Figure 4.3 and Figure 4.4, where the x-axis denotes total number of required (a)(c) epochs and (b)(d) communication rounds. Then we compare the proposed D-GET with the NEXT [76], PSGD [68] and GNSD [79] over the path communication graph $\mathcal{E}$. It can be observed that the proposed D-GET could achieve much faster convergence in terms of sample complexity, while matches the communication complexity as the deterministic algorithms, as claimed in Theorem 4.2.1 and 4.3.1.



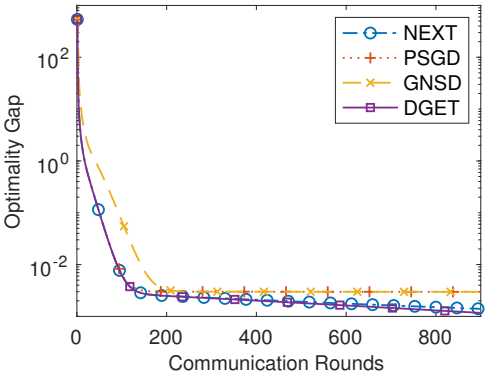
(a) Loss Function v.s. Sample Complexity



(b) Loss Function v.s. Communication Complexity



(c) Optimality Gap v.s. Sample Complexity



(d) Optimality Gap v.s. Communication Complexity

Figure 4.3: Logistic regression on the w8a dataset over the path graph

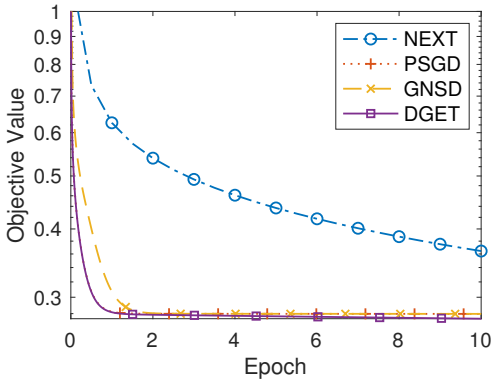
4.5 Proofs of Lemmas and Theorems

Before we formally conduct the analysis, we note three simple facts about Algorithm 3.

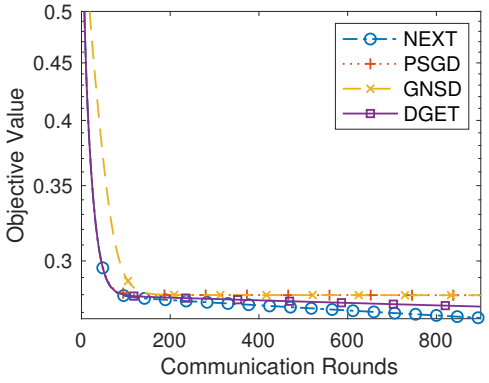
First, according to (4.12) and the definition (4.16a), the update rule of the average iterates can be expressed as:

$$\bar{\mathbf{x}}^r = \bar{\mathbf{x}}^{r-1} - \alpha \bar{\mathbf{y}}^{r-1}. \tag{4.36}$$

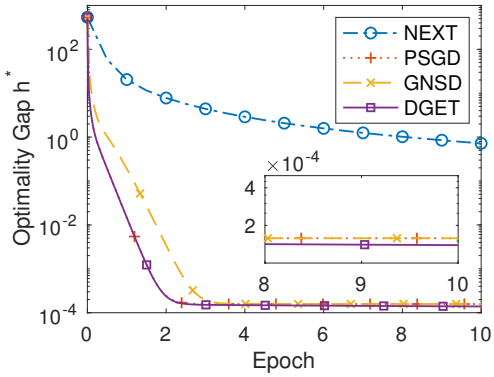
Second, if the iteration r satisfies $\mathrm{mod}(r, q) = 0$ (that is, when the outer iteration is executed), from (4.13) and (4.15) it is easy to check that the following relations hold



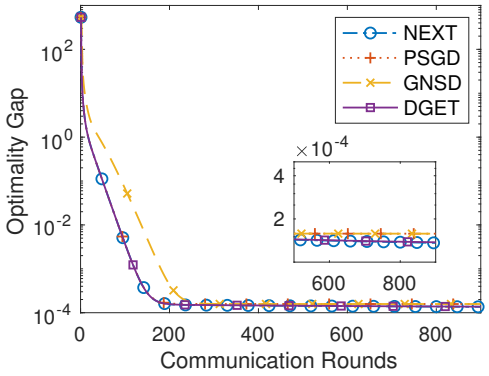
(a) Loss Function v.s. Sample Complexity



(b) Loss Function v.s. Communication Complexity



(c) Optimality Gap v.s. Sample Complexity



(d) Optimality Gap v.s. Communication Complexity

Figure 4.4: Robust linear regression on the w8a dataset over the path graph

(given $\mathbf{v}^0 = \mathbf{y}^0$):

$$\bar{\mathbf{v}}^r = \frac{1}{m}\sum_{i=1}^{m} \nabla f^i(\mathbf{x}_i^r) = \frac{1}{mn}\sum_{i=1}^{m}\sum_{k=1}^{n} \nabla f_k^i(\mathbf{x}_i^r), \tag{4.37}$$

$$\bar{\mathbf{y}}^r = \bar{\mathbf{v}}^r, \text{ if } \mathrm{mod}(r, q) = 0. \tag{4.38}$$

Third, if $\mathrm{mod}(r,q) \neq 0$, we have the following relations:

$$\bar{\mathbf{v}}^r = \frac{1}{m|S_2|}\sum_{i=1}^{m}\sum_{j\in S_2}\left[\nabla f_j^i(\mathbf{x}_i^r) - \nabla f_j^i(\mathbf{x}_i^{r-1})\right] + \bar{\mathbf{v}}^{r-1}, \tag{4.39}$$

$$\bar{\mathbf{y}}^r = \bar{\mathbf{y}}^{r-1} + \bar{\mathbf{v}}^r - \bar{\mathbf{v}}^{r-1}, \text{ if } \mathrm{mod}(r,q) \neq 0. \tag{4.40}$$

4.5.1 Proof of Lemma 4.2.1

Proof. Define $\mathbb{E}[\cdot|\mathcal{F}_r]$ as the expectation with respect to the random choice of sample j, conditioned on $\mathbf{x}^0, \cdots, \mathbf{x}^r$, $\mathbf{v}^0, \cdots, \mathbf{v}^{r-1}$ and $\mathbf{y}^0, \cdots, \mathbf{y}^{r-1}$.

First, we have the following identity (which holds true when $\mathrm{mod}(r,q) \neq 0$)

$$\begin{aligned}
&\mathbb{E}[\bar{\mathbf{v}}^r - \bar{\mathbf{v}}^{r-1}|\mathcal{F}_r] \\
&\overset{(4.39)}{=} \mathbb{E}\left[\frac{1}{m|S_2|}\sum_{i=1}^{m}\sum_{j\in S_2}\left[\nabla f_j^i(\mathbf{x}_i^r) - \nabla f_j^i(\mathbf{x}_i^{r-1})\right]\Big|\mathcal{F}_r\right] \\
&= \frac{1}{m}\sum_{i=1}^{m}\left[\nabla f^i(\mathbf{x}_i^r) - \nabla f^i(\mathbf{x}_i^{r-1})\right].
\end{aligned} \tag{4.41}$$

To see why the second equality holds, note that when $\mathbf{x}^r, \mathbf{y}^{r-1}, \mathbf{v}^{r-1}$ are known and fixed, the second expectation is taken over the random selection of S_2. The second equality follows because S_2 is sampled from $[n]$ uniformly with replacement, and it is an unbiased estimate of the averaged gradient.

Second, it is straightforward to obtain the following equality,

$$\begin{aligned}
&\left\|\bar{\mathbf{y}}^r - \frac{1}{m}\sum_{i=1}^{m}\nabla f^i(\mathbf{x}_i^r)\right\|^2 \\
&\overset{(4.40)}{=} \left\|\bar{\mathbf{y}}^{r-1} + \bar{\mathbf{v}}^r - \bar{\mathbf{v}}^{r-1} - \frac{1}{m}\sum_{i=1}^{m}\nabla f^i(\mathbf{x}_i^r)\right\|^2 \\
&= \left\|\bar{\mathbf{y}}^{r-1} - \frac{1}{m}\sum_{i=1}^{m}\nabla f^i(\mathbf{x}_i^{r-1})\right\|^2 + \left\|\bar{\mathbf{v}}^r - \bar{\mathbf{v}}^{r-1} + \frac{1}{m}\sum_{i=1}^{m}\nabla f^i(\mathbf{x}_i^{r-1}) - \frac{1}{m}\sum_{i=1}^{m}\nabla f^i(\mathbf{x}_i^r)\right\|^2 \\
&+ 2\left\langle \bar{\mathbf{y}}^{r-1} - \frac{1}{m}\sum_{i=1}^{m}\nabla f^i(\mathbf{x}_i^{r-1}), \bar{\mathbf{v}}^r - \bar{\mathbf{v}}^{r-1} + \frac{1}{m}\sum_{i=1}^{m}\nabla f^i(\mathbf{x}_i^{r-1}) - \frac{1}{m}\sum_{i=1}^{m}\nabla f^i(\mathbf{x}_i^r)\right\rangle,
\end{aligned} \tag{4.42}$$

where in the second equality, we add and subtract a term $\frac{1}{m}\sum_{i=1}^{m}\nabla f^i(\mathbf{x}_i^{r-1})$.

The cross term in (4.42) can be eliminated if we take the conditional expectation conditioning on $\mathcal{F}_r$. Since under $\mathcal{F}_r$, we have $\mathbf{x}^r, \mathbf{x}^{r-1}, \mathbf{v}^{r-1}, \mathbf{y}^{r-1}, \bar{\mathbf{y}}^{r-1}$ are all known and fixed. Further applying (4.41) we have

$$\mathbb{E}\left[\bar{\mathbf{v}}^r - \bar{\mathbf{v}}^{r-1} + \frac{1}{m}\sum_{i=1}^{m}\nabla f^i(\mathbf{x}_i^{r-1}) - \frac{1}{m}\sum_{i=1}^{m}\nabla f^i(\mathbf{x}_i^r)\middle|\mathcal{F}_r\right] = 0. \tag{4.43}$$

Further taking the full expectation on (4.42) we have

$$\begin{aligned}\mathbb{E}\left\|\bar{\mathbf{y}}^r - \frac{1}{m}\sum_{i=1}^{m}\nabla f^i(\mathbf{x}_i^r)\right\|^2 &\overset{(4.2)(4.39)(4.43)}{=} \mathbb{E}\left\|\bar{\mathbf{y}}^{r-1} - \frac{1}{m}\sum_{i=1}^{m}\nabla f^i(\mathbf{x}_i{}^{r-1})\right\|^2 \\ &+ \mathbb{E}\left\|\frac{1}{m|S_2|}\sum_{i=1}^{m}\sum_{j\in S_2}\nabla f_j^i(\mathbf{x}_i^r) - \frac{1}{m|S_2|}\sum_{i=1}^{m}\sum_{j\in S_2}\nabla f_j^i(\mathbf{x}_i^{r-1}) + \frac{1}{mn}\sum_{i=1}^{m}\sum_{k=1}^{n}\nabla f_k^i(\mathbf{x}_i{}^{r-1}) - \frac{1}{mn}\sum_{i=1}^{m}\sum_{k=1}^{n}\nabla f_k^i(\mathbf{x}_i^r)\right\|^2.\end{aligned} \tag{4.44}$$

Since each $j \in S_2$ is sampled from $[n]$ uniformly, we have the following conditional expectation:

$$\mathbb{E}_j\left[\frac{1}{m}\sum_{i=1}^{m}\nabla f_j^i(\mathbf{x}_i^r) - \frac{1}{m}\sum_{i=1}^{m}\nabla f_j^i(\mathbf{x}_i^{r-1}) - \frac{1}{mn}\sum_{i=1}^{m}\sum_{k=1}^{n}\nabla f_k^i(\mathbf{x}_i^r) + \frac{1}{mn}\sum_{i=1}^{m}\sum_{k=1}^{n}\nabla f_k^i(\mathbf{x}_i{}^{r-1})\middle|\mathcal{F}_r\right] = 0. \tag{4.45}$$

Then the second term of RHS of (4.44) can be further bounded through following

(where $\mathbb{E}[\cdot]$ is the full expectation)

$$\begin{aligned}
&\mathbb{E}\left\|\frac{1}{m|S_2|}\sum_{i=1}^{m}\sum_{j\in S_2}\nabla f_j^i(\mathbf{x}_i^r)-\frac{1}{m|S_2|}\sum_{i=1}^{m}\sum_{j\in S_2}\nabla f_j^i(\mathbf{x}_i^{r-1})-\frac{1}{mn}\sum_{i=1}^{m}\sum_{k=1}^{n}\nabla f_k^i(\mathbf{x}_i^r)+\frac{1}{mn}\sum_{i=1}^{m}\sum_{k=1}^{n}\nabla f_k^i(\mathbf{x}_i{}^{r-1})\right\|^2\\
=&\frac{1}{|S_2|^2}\mathbb{E}\left\|\sum_{j\in S_2}\left(\frac{1}{m}\sum_{i=1}^{m}\nabla f_j^i(\mathbf{x}_i^r)-\frac{1}{m}\sum_{i=1}^{m}\nabla f_j^i(\mathbf{x}_i^{r-1})-\frac{1}{mn}\sum_{i=1}^{m}\sum_{k=1}^{n}\nabla f_k^i(\mathbf{x}_i^r)+\frac{1}{mn}\sum_{i=1}^{m}\sum_{k=1}^{n}\nabla f_k^i(\mathbf{x}_i{}^{r-1})\right)\right\|^2\\
\overset{(i)}{=}&\frac{1}{|S_2|^2}\sum_{j\in S_2}\mathbb{E}\left\|\frac{1}{m}\sum_{i=1}^{m}\nabla f_j^i(\mathbf{x}_i^r)-\frac{1}{m}\sum_{i=1}^{m}\nabla f_j^i(\mathbf{x}_i^{r-1})-\frac{1}{mn}\sum_{i=1}^{m}\sum_{k=1}^{n}\nabla f_k^i(\mathbf{x}_i^r)+\frac{1}{mn}\sum_{i=1}^{m}\sum_{k=1}^{n}\nabla f_k^i(\mathbf{x}_i{}^{r-1})\right\|^2\\
\overset{(ii)}{\leq}&\frac{1}{|S_2|}\mathbb{E}\left\|\frac{1}{m}\sum_{i=1}^{m}\nabla f_j^i(\mathbf{x}_i^r)-\frac{1}{m}\sum_{i=1}^{m}\nabla f_j^i(\mathbf{x}_i^{r-1})\right\|^2\\
\overset{(iii)}{\leq}&\frac{1}{m|S_2|}\mathbb{E}\left[\sum_{i=1}^{m}\|\nabla f_j^i(\mathbf{x}_i^r)-\nabla f_j^i(\mathbf{x}_i^{r-1})\|^2\right]\\
\overset{(iv)}{\leq}&\frac{L^2}{m|S_2|}\mathbb{E}\|\mathbf{x}^r-\mathbf{x}^{r-1}\|^2,
\end{aligned}\tag{4.46}$$

In step (i) of the above relation, we use the fact that for two random variables u_i, u_j which are independent conditioning on $\mathcal{F}$, the following holds

$$\mathbb{E}[\langle u_\ell, u_j\rangle] = \mathbb{E}_{\mathcal{F}}\mathbb{E}[\langle u_\ell, u_j\rangle \mid \mathcal{F}] = \mathbb{E}_{\mathcal{F}}\langle \mathbb{E}[u_\ell \mid \mathcal{F}], \mathbb{E}[u_j \mid \mathcal{F}]\rangle. \tag{4.47}$$

Plugging u_j as below and u_ℓ similarly,

$$u_j = \frac{1}{m}\sum_{i=1}^{m}\nabla f_j^i(\mathbf{x}_i^r)-\frac{1}{m}\sum_{i=1}^{m}\nabla f_j^i(\mathbf{x}_i^{r-1})-\frac{1}{mn}\sum_{i=1}^{m}\sum_{k=1}^{n}\nabla f_k^i(\mathbf{x}_i^r)+\frac{1}{mn}\sum_{i=1}^{m}\sum_{k=1}^{n}\nabla f_k^i(\mathbf{x}_i{}^{r-1}) \tag{4.48}$$

and note that (4.45) holds true, we can show that the cross terms in the step before (i) can all be eliminated. In step (ii) of (4.46), we use the property that $\mathbb{E}\|w_j - \mathbb{E}(w_j)\|^2 \leq \mathbb{E}\|w_j\|^2$ and $\mathbb{E}\|w_j\|^2 = \mathbb{E}\|w_k\|^2$ with $w_j := \frac{1}{m}\sum_{i=1}^{m}\nabla f_j^i(\mathbf{x}_i^r) - \frac{1}{m}\sum_{i=1}^{m}\nabla f_j^i(\mathbf{x}_i^{r-1})$; in (iii) we use the Jensen's inequality, and the last inequality (iv) follows Assumption 1.

Therefore, by combining (4.44) and (4.46), we have for all $(n_r-1)q+1 \leq r \leq n_r q-1$,

$$\mathbb{E}\left\|\bar{\mathbf{y}}^r-\frac{1}{m}\sum_{i=1}^{m}\nabla f^i(\mathbf{x}_i^r)\right\|^2 \leq \mathbb{E}\left\|\bar{\mathbf{y}}^{r-1}-\frac{1}{m}\sum_{i=1}^{m}\nabla f^i(\mathbf{x}_i{}^{r-1})\right\|^2+\frac{L^2}{m|S_2|}\mathbb{E}\left\|\mathbf{x}^r-\mathbf{x}^{r-1}\right\|^2. \tag{4.49}$$

Next, note that we have the following bound on $\mathbb{E}\|\mathbf{x}^r - \mathbf{x}^{r-1}\|^2$ for all $r \geq 1$:

$$\begin{aligned}\mathbb{E}\|\mathbf{x}^r - \mathbf{x}^{r-1}\|^2 &\overset{(4.12)}{=} \mathbb{E}\|\mathbf{W}\mathbf{x}^{r-1} - \alpha\mathbf{y}^{r-1} - \mathbf{x}^{r-1}\|^2 \\ &\overset{(i)}{\leq} 2\mathbb{E}\|\mathbf{W}\mathbf{x}^{r-1} - \mathbf{x}^{r-1}\|^2 + 2\alpha^2\mathbb{E}\|\mathbf{y}^{r-1}\|^2 \\ &\overset{(ii)}{\leq} 2\mathbb{E}\|(\mathbf{W}-\mathbf{I})(\mathbf{x}^{r-1} - \mathbf{1}\bar{\mathbf{x}}^{r-1})\|^2 + 2\alpha^2\mathbb{E}\|\mathbf{y}^{r-1}\|^2 \\ &\overset{(iii)}{\leq} 8\mathbb{E}\|\mathbf{x}^{r-1} - \mathbf{1}\bar{\mathbf{x}}^{r-1}\|^2 + 4\alpha^2\mathbb{E}\|\mathbf{y}^{r-1} - \mathbf{1}\bar{\mathbf{y}}^{r-1}\|^2 + 4\alpha^2\mathbb{E}\|\mathbf{1}\bar{\mathbf{y}}^{r-1}\|^2, \ \forall\ r \geq 1\end{aligned} \tag{4.50}$$

where in (i) we apply the Cauchy-Schwarz inequality, (ii) follows that $\mathbf{W1} = \mathbf{1}$ from Assumption 2, and (iii) applies the fact $\|\mathbf{W}-\mathbf{I}\| \leq \|\mathbf{W}\| + \|\mathbf{I}\| \leq 2$ (due to Assumption 2 and the Cauchy-Schwarz inequality).

Telescoping the above inequality (4.49) over the n_r-th inner loop, that is from $(n_r - 1)q + 1$ to r, we obtain the following series of inequalities

$$\begin{aligned}&\mathbb{E}\|\bar{\mathbf{y}}^r - \frac{1}{m}\sum_{i=1}^m \nabla f^i(\mathbf{x}_i^r)\|^2 \\ \leq&\frac{L^2}{m|S_2|}\sum_{t=(n_r-1)q+1}^{r}\mathbb{E}\|\mathbf{x}^t - \mathbf{x}^{t-1}\|^2 + \mathbb{E}\|\bar{\mathbf{y}}^{(n_r-1)q} - \frac{1}{m}\sum_{i=1}^m \nabla f^i(\mathbf{x}_i{}^{(n_r-1)q})\|^2 \\ \overset{(4.50)}{\leq}&\frac{8L^2}{m|S_2|}\sum_{t=(n_r-1)q+1}^{r}\mathbb{E}\|\mathbf{x}^{t-1} - \mathbf{1}\bar{\mathbf{x}}^{t-1}\|^2 + \frac{4\alpha^2L^2}{m|S_2|}\sum_{t=(n_r-1)q+1}^{r}\mathbb{E}\|\mathbf{y}^{t-1} - \mathbf{1}\bar{\mathbf{y}}^{t-1}\|^2 \\ &+ \frac{4\alpha^2L^2}{m|S_2|}\sum_{t=(n_r-1)q+1}^{r}\mathbb{E}\|\mathbf{1}\bar{\mathbf{y}}^{t-1}\|^2 + \mathbb{E}\|\bar{\mathbf{y}}^{(n_r-1)q} - \frac{1}{m}\sum_{i=1}^m \nabla f^i(\mathbf{x}_i{}^{(n_r-1)q})\|^2 \\ \overset{(i)}{\leq}&\frac{8L^2}{m|S_2|}\sum_{t=(n_r-1)q}^{r}\mathbb{E}\|\mathbf{x}^t - \mathbf{1}\bar{\mathbf{x}}^t\|^2 + \frac{4\alpha^2L^2}{m|S_2|}\sum_{t=(n_r-1)q}^{r}\mathbb{E}\|\mathbf{y}^t - \mathbf{1}\bar{\mathbf{y}}^t\|^2 \\ &+ \frac{4\alpha^2L^2}{|S_2|}\sum_{t=(n_r-1)q}^{r}\mathbb{E}\|\bar{\mathbf{y}}^t\|^2 + \mathbb{E}\|\bar{\mathbf{y}}^{(n_r-1)q} - \frac{1}{m}\sum_{i=1}^m \nabla f^i(\mathbf{x}_i{}^{(n_r-1)q})\|^2,\end{aligned}$$

where in (i) we change the index in the summation, and add three non-negative terms (one for each sum). This concludes the first part of this lemma.

Next we show that (4.18) holds true. First, by using the same argument as in (4.41),

we can obtain the following

$$\mathbb{E}[\mathbf{v}^r - \mathbf{v}^{r-1}|\mathcal{F}_r] \overset{(4.13)}{=} \mathbb{E}\left[\frac{1}{|S_2|}\sum_{j\in S_2}\left[\nabla f_j(\mathbf{x}^r) - \nabla f_j(\mathbf{x}^{r-1})\right]\Big|\mathcal{F}_r\right] = \nabla f(\mathbf{x}^r) - \nabla f(\mathbf{x}^{r-1}).$$

By using the above fact, and that conditioning on $\mathcal{F}_r$, $\mathbf{x}^r, \mathbf{x}^{r-1}$ and $\mathbf{v}^{r-1}$, we obtain the following:

$$\mathbb{E}[\langle \mathbf{v}^{r-1} - \nabla f(\mathbf{x}^{r-1}), \mathbf{v}^r - \mathbf{v}^{r-1} - \nabla f(\mathbf{x}^r) + \nabla f(\mathbf{x}^{r-1})\rangle|\mathcal{F}_r] = 0. \tag{4.51}$$

Then it is straightforward to obtain following:

$$\begin{aligned}
&\mathbb{E}\|\mathbf{v}^r - \nabla f(\mathbf{x}^r)\|^2 \\
&= \mathbb{E}\|\mathbf{v}^{r-1} - \nabla f(\mathbf{x}^{r-1}) + \mathbf{v}^r - \mathbf{v}^{r-1} - \nabla f(\mathbf{x}^r) + \nabla f(\mathbf{x}^{r-1})\|^2 \\
&\overset{(4.51)}{=} \mathbb{E}\|\mathbf{v}^{r-1} - \nabla f(\mathbf{x}^{r-1})\|^2 + \mathbb{E}\|\mathbf{v}^r - \mathbf{v}^{r-1} - \nabla f(\mathbf{x}^r) + \nabla f(\mathbf{x}^{r-1})\|^2 \\
&\overset{(4.13)}{=} \mathbb{E}\|\mathbf{v}^{r-1} - \nabla f(\mathbf{x}^{r-1})\|^2 \\
&\qquad + \mathbb{E}\|\frac{1}{|S_2|}\sum_{j\in S_2}\nabla f_j(\mathbf{x}^r) - \frac{1}{|S_2|}\sum_{j\in S_2}\nabla f_j(\mathbf{x}^{r-1}) - \nabla f(\mathbf{x}^r) + \nabla f(\mathbf{x}^{r-1})\|^2 \\
&\overset{(i)}{\leq} \mathbb{E}\|\mathbf{v}^{r-1} - \nabla f(\mathbf{x}^{r-1})\|^2 + \frac{1}{|S_2|}\mathbb{E}\|\nabla f_j(\mathbf{x}^r) - \nabla f_j(\mathbf{x}^{r-1})\|^2 \\
&\overset{(ii)}{\leq} \mathbb{E}\|\mathbf{v}^{r-1} - \nabla f(\mathbf{x}^{r-1})\|^2 + \frac{L^2}{|S_2|}\mathbb{E}\|\mathbf{x}^r - \mathbf{x}^{r-1}\|^2,
\end{aligned}$$

where (i) and (ii) follow the similar arguments as in (4.46).

Telescoping the above inequality over r from $(n_r - 1)q + 1$ to r, we obtain that

$$\begin{aligned}
\mathbb{E}\|\mathbf{v}^r - \nabla f(\mathbf{x}^r)\|^2 &\leq \mathbb{E}\|\mathbf{v}^{(n_r-1)q} - \nabla f(\mathbf{x}^{(n_r-1)q})\|^2 + \frac{L^2}{|S_2|}\sum_{t=(n_r-1)q+1}^{r}\mathbb{E}\|\mathbf{x}^t - \mathbf{x}^{t-1}\|^2 \\
&\leq \mathbb{E}\|\mathbf{v}^{(n_r-1)q} - \nabla f(\mathbf{x}^{(n_r-1)q})\|^2 + \frac{L^2}{|S_2|}\sum_{t=(n_r-1)q}^{r}\mathbb{E}\|\mathbf{x}^{t+1} - \mathbf{x}^t\|^2.
\end{aligned}$$

This completes the proof of the second part of this lemma. **Q.E.D.**

4.5.2 Proof of Lemma 4.2.2

Proof. We first establish the relation of function values between the iterates. According to the gradient Lipschitz continuity (Assumption 1), we have

$$\begin{aligned}
f(\bar{\mathbf{x}}^{r+1}) &\leq f(\bar{\mathbf{x}}^r) + \langle \nabla f(\bar{\mathbf{x}}^r), \bar{\mathbf{x}}^{r+1} - \bar{\mathbf{x}}^r \rangle + \frac{L}{2}\|\bar{\mathbf{x}}^{r+1} - \bar{\mathbf{x}}^r\|^2 \\
&\overset{(i)}{=} f(\bar{\mathbf{x}}^r) - \alpha \langle \nabla f(\bar{\mathbf{x}}^r), \bar{\mathbf{y}}^r \rangle + \frac{\alpha^2 L}{2}\|\bar{\mathbf{y}}^r\|^2 \\
&\overset{(ii)}{=} f(\bar{\mathbf{x}}^r) - \alpha \langle \nabla f(\bar{\mathbf{x}}^r) - \bar{\mathbf{y}}^r, \bar{\mathbf{y}}^r \rangle - \alpha\|\bar{\mathbf{y}}^r\|^2 + \frac{\alpha^2 L}{2}\|\bar{\mathbf{y}}^r\|^2 \\
&\overset{(iii)}{\leq} f(\bar{\mathbf{x}}^r) + \frac{\alpha}{2}\|\nabla f(\bar{\mathbf{x}}^r) - \bar{\mathbf{y}}^r\|^2 - \frac{\alpha}{2}\|\bar{\mathbf{y}}^r\|^2 + \frac{\alpha^2 L}{2}\|\bar{\mathbf{y}}^r\|^2 \\
&\overset{(iv)}{\leq} f(\bar{\mathbf{x}}^r) - \left(\frac{\alpha}{2} - \frac{\alpha^2 L}{2}\right)\|\bar{\mathbf{y}}^r\|^2 + \alpha\|\nabla f(\bar{\mathbf{x}}^r) - \frac{1}{m}\sum_{i=1}^m \nabla f^i(\mathbf{x}_i^r)\|^2 \\
&\quad + \alpha\|\frac{1}{m}\sum_{i=1}^m \nabla f^i(\mathbf{x}_i^r) - \bar{\mathbf{y}}^r\|^2,
\end{aligned}$$

where we simply plug in the iterates (4.36) in (i), add and subtract a term $\bar{\mathbf{y}}^r$ in (ii), and apply the Cauchy-Schwarz inequality in (iii) and (iv).

Then the third term can be further quantified as below,

$$\begin{aligned}
&\mathbb{E}\|\nabla f(\bar{\mathbf{x}}^r) - \frac{1}{m}\sum_{i=1}^m \nabla f^i(\mathbf{x}_i^r)\|^2 \\
\overset{(i)}{\leq} &\frac{1}{m}\sum_{i=1}^m \mathbb{E}\|\nabla f^i(\bar{\mathbf{x}}^r) - \nabla f^i(\mathbf{x}_i^r)\|^2 \\
\overset{(ii)}{\leq} &\frac{1}{m}\sum_{i=1}^m L^2\mathbb{E}\|\mathbf{x}_i^r - \bar{\mathbf{x}}_i^r\|^2 \\
= &\frac{L^2}{m}\mathbb{E}\|\mathbf{x}^r - \mathbf{1}\bar{\mathbf{x}}^r\|^2
\end{aligned}$$

where in (i) we use the Jensen's inequality and in (ii) we use the Lipschitz Assumption 1.

Taking expectation on both sides and combining with (4.17) in Lemma 4.2.1, we

have

$$
\begin{aligned}
&\mathbb{E}[f(\bar{\mathbf{x}}^{r+1})] \\
&\leq \mathbb{E}[f(\bar{\mathbf{x}}^{r})] - \left(\frac{\alpha}{2} - \frac{\alpha^2 L}{2}\right)\mathbb{E}\|\bar{\mathbf{y}}^{r}\|^2 + \frac{\alpha L^2}{m}\mathbb{E}\|\mathbf{x}^{r} - \mathbf{1}\bar{\mathbf{x}}^{r}\|^2 \\
&+ \frac{8\alpha L^2}{m|S_2|}\sum_{t=(n_r-1)q}^{r}\mathbb{E}\|\mathbf{x}^{t} - \mathbf{1}\bar{\mathbf{x}}^{t}\|^2 + \frac{4\alpha^3 L^2}{m|S_2|}\sum_{t=(n_r-1)q}^{r}\mathbb{E}\|\mathbf{y}^{t} - \mathbf{1}\bar{\mathbf{y}}^{t}\|^2 \\
&+ \frac{4\alpha^3 L^2}{|S_2|}\sum_{t=(n_r-1)q}^{r}\mathbb{E}\|\bar{\mathbf{y}}^{t}\|^2 + \alpha\mathbb{E}\|\bar{\mathbf{y}}^{(n_r-1)q} - \frac{1}{m}\sum_{t=1}^{m}\nabla f^{t}(\mathbf{x}_t{}^{(n_r-1)q})\|^2 \\
&\leq \mathbb{E}[f(\bar{\mathbf{x}}^{r})] - \left(\frac{\alpha}{2} - \frac{\alpha^2 L}{2}\right)\mathbb{E}\|\bar{\mathbf{y}}^{r}\|^2 + \frac{\alpha L^2}{m}\mathbb{E}\|\mathbf{x}^{r} - \mathbf{1}\bar{\mathbf{x}}^{r}\|^2 + \frac{8\alpha L^2}{m|S_2|}\sum_{t=(n_r-1)q}^{r}\mathbb{E}\|\mathbf{x}^{t} - \mathbf{1}\bar{\mathbf{x}}^{t}\|^2 \\
&+ \frac{4\alpha^3 L^2}{m|S_2|}\sum_{t=(n_r-1)q}^{r}\mathbb{E}\|\mathbf{y}^{t} - \mathbf{1}\bar{\mathbf{y}}^{t}\|^2 + \frac{4\alpha^3 L^2}{|S_2|}\sum_{t=(n_r-1)q}^{r}\mathbb{E}\|\bar{\mathbf{y}}^{t}\|^2 + \alpha\epsilon_1,
\end{aligned}
$$

where in the last inequality we use the definition of ϵ_1 in (4.19).

Next, telescoping over one inner loop, that is r from $(n_r - 1)q$ to r, we have

$$
\begin{aligned}
&\mathbb{E}[f(\bar{\mathbf{x}}^{r+1})] \leq \mathbb{E}[f(\bar{\mathbf{x}}^{(n_r-1)q})] - \left(\frac{\alpha}{2} - \frac{\alpha^2 L}{2}\right)\sum_{t=(n_r-1)q}^{r}\mathbb{E}\|\bar{\mathbf{y}}^{t}\|^2 + \frac{\alpha L^2}{m}\sum_{t=(n_r-1)q}^{r}\mathbb{E}\|\mathbf{x}^{t} - \mathbf{1}\bar{\mathbf{x}}^{t}\|^2 \\
&+ \frac{8\alpha L^2}{m|S_2|}\sum_{t=(n_r-1)q}^{r}\sum_{k=(n_r-1)q}^{t}\mathbb{E}\|\mathbf{x}^{k} - \mathbf{1}\bar{\mathbf{x}}^{k}\|^2 + \frac{4\alpha^3 L^2}{m|S_2|}\sum_{t=(n_r-1)q}^{r}\sum_{k=(n_r-1)q}^{t}\mathbb{E}\|\mathbf{y}^{k} - \mathbf{1}\bar{\mathbf{y}}^{k}\|^2 \\
&+ \frac{4\alpha^3 L^2}{|S_2|}\sum_{t=(n_r-1)q}^{r}\sum_{k=(n_r-1)q}^{t}\mathbb{E}\|\bar{\mathbf{y}}^{k}\|^2 + \alpha\sum_{t=(n_r-1)q}^{r}\epsilon_1 \\
&\overset{(i)}{\leq} \mathbb{E}[f(\bar{\mathbf{x}}^{(n_r-1)q})] - \left(\frac{\alpha}{2} - \frac{\alpha^2 L}{2} - \frac{4\alpha^3 L^2 q}{|S_2|}\right)\sum_{t=(n_r-1)q}^{r}\mathbb{E}\|\bar{\mathbf{y}}^{t}\|^2 + \frac{\alpha L^2}{m}\sum_{t=(n_r-1)q}^{r}\mathbb{E}\|\mathbf{x}^{t} - \mathbf{1}\bar{\mathbf{x}}^{t}\|^2 \\
&+ \frac{8\alpha L^2 q}{m|S_2|}\sum_{t=(n_r-1)q}^{r}\mathbb{E}\|\mathbf{x}^{t} - \mathbf{1}\bar{\mathbf{x}}^{t}\|^2 + \frac{4\alpha^3 L^2 q}{m|S_2|}\sum_{t=(n_r-1)q}^{r}\mathbb{E}\|\mathbf{y}^{t} - \mathbf{1}\bar{\mathbf{y}}^{t}\|^2 + \alpha\sum_{t=(n_r-1)q}^{r}\epsilon_1,
\end{aligned}
$$

where (i) follows the fact that, for any sequence $\{a^i\}$, and an index $r \leq n_r q - 1$, we

have

$$\sum_{t=(n_r-1)q}^{r}\sum_{k=(n_r-1)q}^{t} a^k \leq \sum_{t=(n_r-1)q}^{r}\sum_{k=(n_r-1)q}^{r} a^k \leq q\sum_{k=(n_r-1)q}^{r} a^k. \tag{4.52}$$

Then utilizing the fact that

$$\begin{aligned}&\mathbb{E}[f(\bar{\mathbf{x}}^{r+1})] - \mathbb{E}[f(\bar{\mathbf{x}}^0)] \\ =&\mathbb{E}[f(\bar{\mathbf{x}}^{r+1})] - \mathbb{E}[f(\bar{\mathbf{x}}^{(n_r-1)q})] + \cdots + \mathbb{E}[f(\bar{\mathbf{x}}^{2q})] - \mathbb{E}[f(\bar{\mathbf{x}}^{q})] + \mathbb{E}[f(\bar{\mathbf{x}}^{q})] - \mathbb{E}[f(\bar{\mathbf{x}}^{0})],\end{aligned} \tag{4.53}$$

we have

$$\begin{aligned}\mathbb{E}[f(\bar{\mathbf{x}}^{r+1})] \leq \mathbb{E}[f(\bar{\mathbf{x}}^0)] - \left(\frac{\alpha}{2} - \frac{\alpha^2 L}{2} - \frac{4\alpha^3 L^2 q}{|S_2|}\right)\sum_{t=0}^{r}\mathbb{E}\|\bar{\mathbf{y}}^t\|^2 + \frac{\alpha L^2}{m}\sum_{t=0}^{r}\mathbb{E}\|\mathbf{x}^t - \mathbf{1}\bar{\mathbf{x}}^t\|^2 \\ + \frac{8\alpha L^2 q}{m|S_2|}\sum_{t=0}^{r}\mathbb{E}\|\mathbf{x}^t - \mathbf{1}\bar{\mathbf{x}}^t\|^2 + \frac{4\alpha^3 L^2 q}{m|S_2|}\sum_{t=0}^{r}\mathbb{E}\|\mathbf{y}^t - \mathbf{1}\bar{\mathbf{y}}^t\|^2 + \alpha(r+1)\epsilon_1,\end{aligned}$$

which completes the proof. **Q.E.D.**

4.5.3 Proof of Lemma 4.2.3

Proof. First, using the Assumption 2 on $\mathbf{W}$, we can obtain the contraction property of the iterates disagreement, i.e.,

$$\|\mathbf{W}\mathbf{x}^r - \mathbf{1}\bar{\mathbf{x}}^r\| = \|\mathbf{W}(\mathbf{x}^r - \mathbf{1}\bar{\mathbf{x}}^r)\| \leq \eta\|\mathbf{x}^r - \mathbf{1}\bar{\mathbf{x}}^r\|. \tag{4.54}$$

To see why the inequality holds true, note that $\mathbf{1}^T(\mathbf{x}^r - \mathbf{1}\bar{\mathbf{x}}^r) = 0$, that is, $\mathbf{x}^r - \mathbf{1}\bar{\mathbf{x}}^r$ is orthogonal $\mathbf{1}$, which is the eigenvector corresponding to the largest eigenvalue of $\mathbf{W}$. Combining with the fact that $|\lambda_{\max}(\mathbf{W})| = \eta < 1$, we obtain the above inequality.

Then applying the definition of $\mathbf{x}$ iterates (4.12) and the Cauchy-Schwartz inequality,

we have

$$\begin{aligned}\|\mathbf{x}^{r+1}-\mathbf{1}\bar{\mathbf{x}}^{r+1}\|^2 &\overset{(4.12)}{=}\|\mathbf{W}\mathbf{x}^r-\alpha\mathbf{y}^r-\mathbf{1}(\bar{\mathbf{x}}^r-\alpha\bar{\mathbf{y}}^r)\|^2\\ &\leq(1+\beta)\|\mathbf{W}\mathbf{x}^r-\mathbf{1}\bar{\mathbf{x}}^r\|^2+\left(1+\frac{1}{\beta}\right)\alpha^2\|\mathbf{y}^r-\mathbf{1}\bar{\mathbf{y}}^r\|^2\\ &\overset{(4.54)}{\leq}(1+\beta)\eta^2\|\mathbf{x}^r-\mathbf{1}\bar{\mathbf{x}}^r\|^2+\left(1+\frac{1}{\beta}\right)\alpha^2\|\mathbf{y}^r-\mathbf{1}\bar{\mathbf{y}}^r\|^2,\end{aligned}$$

where β is some constant parameter to be tuned later. Then, taking expectation on the both sides of the above inequality we are able to obtain (4.21).

Similarly, we have

$$\begin{aligned}\|\mathbf{y}^{r+1}-\mathbf{1}\bar{\mathbf{y}}^{r+1}\|^2 &\overset{(4.15)}{=}\|\mathbf{W}\mathbf{y}^r+\mathbf{v}^{r+1}-\mathbf{v}^r-\mathbf{1}(\bar{\mathbf{y}}^r+\bar{\mathbf{v}}^{r+1}-\bar{\mathbf{v}}^r)\|^2\\ &\leq(1+\beta)\|\mathbf{W}\mathbf{y}^r-\mathbf{1}\bar{\mathbf{y}}^r\|^2+\left(1+\frac{1}{\beta}\right)\left\|\left(\mathbf{I}-\frac{\mathbf{1}\mathbf{1}^T}{m}\right)(\mathbf{v}^{r+1}-\mathbf{v}^r)\right\|^2\\ &\leq(1+\beta)\eta^2\|\mathbf{y}^r-\mathbf{1}\bar{\mathbf{y}}^r\|^2+\left(1+\frac{1}{\beta}\right)\|\mathbf{v}^{r+1}-\mathbf{v}^r\|^2,\end{aligned}$$

where in the last inequality we also use $\|\mathbf{I}-\frac{1}{m}\mathbf{1}\mathbf{1}^T\|<1$.

After taking expectation on the both sides of the above inequality and combining the following inequalities, the proof for (4.22) is complete.

To further bound the term $\|\mathbf{v}^{r+1}-\mathbf{v}^r\|^2$, consider that we have $(n_r-1)q\leq r\leq n_rq-1$, that is r is taken within one inner loop. We will divide the analysis into two cases.

Case 1) For all $(n_r-1)q\leq r\leq n_rq-2$, we have $\mathrm{mod}(r+1,q)\neq 0$ and the following is straightforward:

$$\begin{aligned}\mathbb{E}\|\mathbf{v}^{r+1}-\mathbf{v}^r\|^2 &\overset{(4.13)}{=}\mathbb{E}\|\frac{1}{|S_2|}\sum_{j\in S_2}\left[\nabla f_j(\mathbf{x}^{r+1})-\nabla f_j(\mathbf{x}^r)\right]\|^2\\ &\overset{(i)}{\leq}\frac{1}{|S_2|}\mathbb{E}\sum_{j\in S_2}\|\nabla f_j(\mathbf{x}^{r+1})-\nabla f_j(\mathbf{x}^r)\|^2\\ &\overset{(ii)}{\leq}L^2\mathbb{E}\|\mathbf{x}^{r+1}-\mathbf{x}^r\|^2,\end{aligned}\tag{4.55}$$

where in (i) we use the Jensen's inequality and in (ii) we use Assumption 1.

Case 2) If $r = n_r q - 1$, we have $\text{mod}(r+1, q) = 0$. Therefore,

$$\begin{aligned}\mathbb{E}\|\mathbf{v}^{r+1} - \mathbf{v}^r\|^2 &= \mathbb{E}\|\mathbf{v}^{r+1} - \nabla f(\mathbf{x}^{r+1}) + \nabla f(\mathbf{x}^{r+1}) - \nabla f(\mathbf{x}^r) + \nabla f(\mathbf{x}^r) - \mathbf{v}^r\|^2 \\ &\overset{(i)}{\leq} 3\mathbb{E}\|\mathbf{v}^{r+1} - \nabla f(\mathbf{x}^{r+1})\|^2 + 3\mathbb{E}\|\nabla f(\mathbf{x}^{r+1}) - \nabla f(\mathbf{x}^r)\|^2 + 3\mathbb{E}\|\nabla f(\mathbf{x}^r) - \mathbf{v}^r\|^2 \\ &\overset{(ii)}{\leq} 3\epsilon_2 + 3L^2\mathbb{E}\|\mathbf{x}^{r+1} - \mathbf{x}^r\|^2 + 3\sum_{t=(n_r-1)q}^{r} \frac{L^2}{|S_2|}\mathbb{E}\|\mathbf{x}^{t+1} - \mathbf{x}^t\|^2 + 3\epsilon_2,\end{aligned} \tag{4.56}$$

where in (i) we use the Cauchy-Schwarz inequality; in (ii) we apply (4.18) from Lemma 4.2.1, Assumption 1, and $\mathbb{E}\|\mathbf{v}^r - \nabla f(\mathbf{x}^r)\|^2 \leq \epsilon_2$ for all $\text{mod}(r, q) = 0$.

Next, telescoping $\|\mathbf{v}^{r+1} - \mathbf{v}^r\|^2$ over r from $(n_r - 1)q$ to r. Since $r \leq n_r q - 1$, we have at most one follows (4.56) and all the rest follow (4.55). Therefore, we obtain

$$\begin{aligned}\sum_{t=(n_r-1)q}^{r} \mathbb{E}\|\mathbf{v}^{t+1} - \mathbf{v}^t\|^2 &\leq \sum_{t=(n_r-1)q}^{r} L^2\mathbb{E}\|\mathbf{x}^{t+1} - \mathbf{x}^t\|^2 + 6\epsilon_2 + 2L^2\mathbb{E}\|\mathbf{x}^{r+1} - \mathbf{x}^r\|^2 \\ &\quad + \sum_{t=(n_r-1)q}^{r} \frac{3L^2}{|S_2|}\mathbb{E}\|\mathbf{x}^{t+1} - \mathbf{x}^t\|^2 \\ &\leq \sum_{t=(n_r-1)q}^{r} 6L^2\mathbb{E}\|\mathbf{x}^{t+1} - \mathbf{x}^t\|^2 + 6\epsilon_2.\end{aligned}$$

Through a similar step as (4.53), the following is obvious

$$\sum_{t=0}^{r} \mathbb{E}\|\mathbf{v}^{t+1} - \mathbf{v}^t\|^2 \leq 6(r+1)\epsilon_2 + \sum_{t=0}^{r} 6L^2\mathbb{E}\|\mathbf{x}^{t+1} - \mathbf{x}^t\|^2.$$

By combining (4.50), i.e.,

$$\mathbb{E}\|\mathbf{x}^{r+1} - \mathbf{x}^r\|^2 \leq 8\mathbb{E}\|\mathbf{x}^r - \mathbf{1}\bar{\mathbf{x}}^r\|^2 + 4\alpha^2\mathbb{E}\|\mathbf{y}^r - \mathbf{1}\bar{\mathbf{y}}^r\|^2 + 4\alpha^2\mathbb{E}\|\mathbf{1}\bar{\mathbf{y}}^r\|^2, \ \forall\ r \geq 0,$$

we complete the proof. **Q.E.D.**

4.5.4 Proof of Lemma 4.2.4

Proof. We first introduce an intermediate function $P(\mathbf{x}^r)$ to facilitate the analysis,

$$P(\mathbf{x}^r) := \mathbb{E}\|\mathbf{x}^r - \mathbf{1}\bar{\mathbf{x}}^r\|^2 + \alpha\mathbb{E}\|\mathbf{y}^r - \mathbf{1}\bar{\mathbf{y}}^r\|^2.$$

Obviously, we have $H(\mathbf{x}^r) = \mathbb{E}[f(\bar{\mathbf{x}}^r)] + \frac{1}{m}P(\mathbf{x}^r)$.

By applying (4.21) and (4.22) in Lemma 4.2.3 we have

$$\begin{aligned}
&P(\mathbf{x}^{r+1}) - P(\mathbf{x}^r) \\
&\leq (1+\beta)\eta^2\mathbb{E}\|\mathbf{x}^r - \mathbf{1}\bar{\mathbf{x}}^r\|^2 + (1+\frac{1}{\beta})\alpha^2\mathbb{E}\|\mathbf{y}^r - \mathbf{1}\bar{\mathbf{y}}^r\|^2 + \alpha(1+\beta)\eta^2\mathbb{E}\|\mathbf{y}^r - \mathbf{1}\bar{\mathbf{y}}^r\|^2 \\
&+ \alpha(1+\frac{1}{\beta})\mathbb{E}\|\mathbf{v}^{r+1} - \mathbf{v}^r\|^2 - \mathbb{E}\|\mathbf{x}^r - \mathbf{1}\bar{\mathbf{x}}^r\|^2 - \alpha\mathbb{E}\|\mathbf{y}^r - \mathbf{1}\bar{\mathbf{y}}^r\|^2 \\
&= -\left(1-(1+\beta)\eta^2\right)\mathbb{E}\|\mathbf{x}^r - \mathbf{1}\bar{\mathbf{x}}^r\|^2 - \left(\alpha - \alpha(1+\beta)\eta^2 - (1+\frac{1}{\beta})\alpha^2\right)\mathbb{E}\|\mathbf{y}^r - \mathbf{1}\bar{\mathbf{y}}^r\|^2 \\
&+ \alpha(1+\frac{1}{\beta})\mathbb{E}\|\mathbf{v}^{r+1} - \mathbf{v}^r\|^2.
\end{aligned}$$

Next, summing over the iteration from 0 to r we obtain

$$\begin{aligned}
P(\mathbf{x}^{r+1}) - P(\mathbf{x}^0) \leq &-\left(1-(1+\beta)\eta^2\right)\sum_{t=0}^{r}\mathbb{E}\|\mathbf{x}^t - \mathbf{1}\bar{\mathbf{x}}^t\|^2 \\
&-\left(\alpha - \alpha(1+\beta)\eta^2 - (1+\frac{1}{\beta})\alpha^2\right)\sum_{t=0}^{r}\mathbb{E}\|\mathbf{y}^t - \mathbf{1}\bar{\mathbf{y}}^t\|^2 \\
&+\alpha(1+\frac{1}{\beta})\sum_{t=0}^{r}\mathbb{E}\|\mathbf{v}^{t+1} - \mathbf{v}^t\|^2.
\end{aligned} \tag{4.57}$$

If we further pick $q = |S_2|$, then Lemma 4.2.2 becomes

$$\begin{aligned}
\mathbb{E}[f(\bar{\mathbf{x}}^{r+1})] \leq &\mathbb{E}[f(\bar{\mathbf{x}}^0)] - \left(\frac{\alpha}{2} - \frac{\alpha^2 L}{2} - 4\alpha^3L^2\right)\sum_{t=0}^{r}\mathbb{E}\|\bar{\mathbf{y}}^t\|^2 \\
&+\frac{9\alpha L^2}{m}\sum_{t=0}^{r}\mathbb{E}\|\mathbf{x}^t - \mathbf{1}\bar{\mathbf{x}}^t\|^2 + \frac{4\alpha^3L^2}{m}\sum_{t=0}^{r}\mathbb{E}\|\mathbf{y}^t - \mathbf{1}\bar{\mathbf{y}}^t\|^2 + \alpha(r+1)\epsilon_1.
\end{aligned} \tag{4.58}$$

Similarly, equation (4.24) of Lemma 4.2.3 becomes

$$\sum_{t=0}^{r}\mathbb{E}\|\mathbf{v}^{t+1}-\mathbf{v}^{t}\|^2 \tag{4.59}$$

$$\leq 48L^2\sum_{t=0}^{r}\mathbb{E}\|\mathbf{x}^t-\mathbf{1}\bar{\mathbf{x}}^t\|^2+24L^2\alpha^2\sum_{t=0}^{r}\mathbb{E}\|\mathbf{y}^t-\mathbf{1}\bar{\mathbf{y}}^t\|^2+24L^2\alpha^2\sum_{t=0}^{r}\mathbb{E}\|\mathbf{1}\bar{\mathbf{y}}^t\|^2+6(r+1)\epsilon_2.$$

Therefore, combine (4.57), (4.58) and (4.59), we have

$$\begin{aligned}
H(\mathbf{x}^{r+1})-H(\mathbf{x}^0)\leq&-\left(\frac{\alpha}{2}-\frac{\alpha^2L}{2}-4\alpha^3L^2-24(1+\frac{1}{\beta})\alpha^3L^2\right)\sum_{t=0}^{r}\mathbb{E}\|\bar{\mathbf{y}}^t\|^2\\
&-\left(1-(1+\beta)\eta^2-48\alpha(1+\frac{1}{\beta})L^2-9\alpha L^2\right)\frac{1}{m}\sum_{t=0}^{r}\mathbb{E}\|\mathbf{x}^t-\mathbf{1}\bar{\mathbf{x}}^t\|^2\\
&-\left(\alpha-\alpha(1+\beta)\eta^2-(1+\frac{1}{\beta})\alpha^2-24(1+\frac{1}{\beta})\alpha^3L^2-4\alpha^3L^2\right)\frac{1}{m}\sum_{t=0}^{r}\mathbb{E}\|\mathbf{y}^t-\mathbf{1}\bar{\mathbf{y}}^t\|^2\\
&+\alpha(r+1)(\epsilon_1+6\frac{1}{m}(1+\frac{1}{\beta})\epsilon_2).
\end{aligned}$$

This completes the proof. **Q.E.D.**

4.5.5 Proof of Theorem 4.2.1

Proof. To begin with, we notice that by applying the update rule from Algorithm 3, then for all $\mathrm{mod}(r,q)=0$, the following holds true

$$\mathbb{E}\|\mathbf{v}^r-\nabla f(\mathbf{x}^r)\|^2\overset{(4.13)}{=}0, \tag{4.60}$$

$$\mathbb{E}\|\bar{\mathbf{y}}^r-\frac{1}{mn}\sum_{i=1}^{m}\sum_{k=1}^{n}\nabla f_k^i(\mathbf{x}_i^r)\|^2\overset{(4.37)}{=}0, \tag{4.61}$$

which implies $\epsilon_1=\epsilon_2=0$ for Lemma 4.2.2, Lemma 4.2.3, and Lemma 4.2.4.

Next, if we further pick β such that $1-(1+\beta)\eta^2>0$ and choose $0<\alpha<$

$\min\{K_1, K_2, K_3\}$, we can rewrite Lemma 4.2.4 as below

$$H(\mathbf{x}^{r+1}) - H(\mathbf{x}^0) \leq -C_1 \sum_{t=0}^{r} \mathbb{E}\|\bar{\mathbf{y}}^t\|^2 - C_2 \sum_{t=0}^{r} \frac{1}{m}\mathbb{E}\|\mathbf{x}^t - \mathbf{1}\bar{\mathbf{x}}^t\|^2 - C_3 \sum_{t=0}^{r} \frac{1}{m}\mathbb{E}\|\mathbf{y}^t - \mathbf{1}\bar{\mathbf{y}}^t\|^2, \tag{4.62}$$

where $C_1 > 0, C_2 > 0, C_3 > 0$.

Therefore the upper bound of the optimality gap can be quantified as the following

$$\begin{aligned}
&\frac{1}{T}\sum_{t=0}^{T} \mathbb{E}\|\frac{1}{m}\sum_{i=1}^{m} \nabla f^i(\mathbf{x}_i^t)\|^2 + \frac{1}{T}\sum_{t=0}^{T} \frac{1}{m}\mathbb{E}\|\mathbf{x}^t - \mathbf{1}\bar{\mathbf{x}}^t\|^2 \\
\leq &\frac{2}{T}\sum_{t=0}^{T} \mathbb{E}\|\bar{\mathbf{y}}^t\|^2 + \frac{2}{T}\sum_{t=0}^{T} \mathbb{E}\|\bar{\mathbf{y}}^t - \frac{1}{m}\sum_{i=1}^{m} \nabla f^i(\mathbf{x}_i^t)\|^2 + \frac{1}{T}\sum_{t=0}^{T} \frac{1}{m}\mathbb{E}\|\mathbf{x}^t - \mathbf{1}\bar{\mathbf{x}}^t\|^2,
\end{aligned}$$

where we use the Cauchy-Schwarz inequality.

Applying (4.17) from Lemma 4.2.1 with $\mathbb{E}\|\bar{\mathbf{y}}^{(n_r-1)q} - \frac{1}{m}\sum_{i=1}^{m} \nabla f^i(\mathbf{x}_i{}^{(n_r-1)q})\|^2 = 0$, telescoping over r from 0 to T (follows similar reasoning as (4.52) and (4.53)), and using the choice of $|S_2| = q = \sqrt{n}$, we have

$$\begin{aligned}
&\sum_{t=0}^{r} \mathbb{E}\|\bar{\mathbf{y}}^t - \frac{1}{m}\sum_{i=1}^{m} \nabla f^i(\mathbf{x}_i^t)\|^2 \\
\leq &\frac{8L^2}{m}\sum_{t=0}^{r} \mathbb{E}\|\mathbf{x}^t - \mathbf{1}\bar{\mathbf{x}}^t\|^2 + \frac{4\alpha^2 L^2}{m}\sum_{t=0}^{r} \mathbb{E}\|\mathbf{y}^t - \mathbf{1}\bar{\mathbf{y}}^t\|^2 + 4\alpha^2 L^2 \sum_{t=0}^{r} \mathbb{E}\|\bar{\mathbf{y}}^t\|^2.
\end{aligned}$$

Combining the above two inequalities we can obtain

$$\begin{aligned}
&\frac{1}{T}\sum_{t=0}^{T} \mathbb{E}\|\frac{1}{m}\sum_{i=1}^{m} \nabla f^i(\mathbf{x}_i^t)\|^2 + \frac{1}{T}\sum_{t=0}^{T} \frac{1}{m}\mathbb{E}\|\mathbf{x}^t - \mathbf{1}\bar{\mathbf{x}}^t\|^2 \\
\leq &\left(\frac{16L^2}{mT} + \frac{1}{mT}\right)\sum_{t=0}^{T} \mathbb{E}\|\mathbf{x}^t - \mathbf{1}\bar{\mathbf{x}}^t\|^2 + \frac{8\alpha^2 L^2}{mT}\sum_{t=0}^{T} \mathbb{E}\|\mathbf{y}^t - \mathbf{1}\bar{\mathbf{y}}^t\|^2 + \left(\frac{8\alpha^2 L^2}{T} + \frac{2}{T}\right)\sum_{t=0}^{T} \|\bar{\mathbf{y}}^t\|^2.
\end{aligned}$$

Further combining with (4.62), we have

$$\begin{aligned}&\frac{1}{T}\sum_{t=0}^{T}\mathbb{E}\|\frac{1}{m}\sum_{i=1}^{m}\nabla f^i(\mathbf{x}_i^t)\|^2+\frac{1}{T}\sum_{t=0}^{T}\frac{1}{m}\mathbb{E}\|\mathbf{x}^t-\mathbf{1}\bar{\mathbf{x}}^t\|^2\\ \leq& C_0\cdot\frac{H(\mathbf{x}^0)-H(\mathbf{x}^{T+1})}{T}\leq C_0\cdot\frac{\mathbb{E}[f(\mathbf{x}^0)]-\underline{f}}{T},\end{aligned}\tag{4.63}$$

where

$$C_0:=\left(\frac{8\alpha^2L^2+2}{C_1}+\frac{16L^2+1}{mC_2}+\frac{8\alpha^2L^2}{mC_3}\right),$$

and the last inequality follows from

$$\begin{aligned}H(\mathbf{x}^0)&:=\mathbb{E}[f(\bar{\mathbf{x}}^0)]+\mathbb{E}\|\mathbf{x}^0-\mathbf{1}\bar{\mathbf{x}}^0\|^2+\alpha\mathbb{E}\|\mathbf{y}^0-\mathbf{1}\bar{\mathbf{y}}^0\|^2=\mathbb{E}[f(\bar{\mathbf{x}}^0)],\\ H(\mathbf{x}^r)&:=\mathbb{E}[f(\bar{\mathbf{x}}^r)]+\mathbb{E}\|\mathbf{x}^r-\mathbf{1}\bar{\mathbf{x}}^r\|^2+\alpha\mathbb{E}\|\mathbf{y}^r-\mathbf{1}\bar{\mathbf{y}}^r\|^2\geq\mathbb{E}[f(\bar{\mathbf{x}}^r)]\geq\underline{f}.\end{aligned}$$

This completes the proof. **Q.E.D.**

4.5.6 Proof of Corollary 1

Proof. If we pick $T=\lfloor C_0\cdot\frac{\mathbb{E}f(\mathbf{x}^0)-\underline{f}}{\epsilon}\rfloor+1\geq C_0\cdot\frac{\mathbb{E}f(\mathbf{x}^0)-\underline{f}}{\epsilon}$, then we can obtain following from Theorem 4.2.1

$$\begin{aligned}&\frac{1}{T}\sum_{t=0}^{T}\mathbb{E}\|\frac{1}{m}\sum_{i=1}^{m}\nabla f^i(\mathbf{x}_i^t)\|^2+\frac{1}{T}\sum_{t=0}^{T}\frac{1}{m}\mathbb{E}\|\mathbf{x}^t-\mathbf{1}\bar{\mathbf{x}}^t\|^2\\ &\leq C_0\cdot\frac{\mathbb{E}f(\mathbf{x}^0)-\underline{f}}{T}\leq\epsilon.\end{aligned}$$

Therefore, the total samples needed will be the sum of outer loop complexity ($\lceil\frac{T}{q}\rceil$ times full (n) gradient evaluations per node) plus inner loop complexity (T times $|S_2|$ gradient evaluations per node), by letting $q=|S_2|=\sqrt{n}$, we conclude that the total

samples needed are

$$
\begin{aligned}
& m \times \left(\lceil \frac{T}{q} \rceil \cdot n + T \cdot |S_2| \right) \\
& \leq m \times \left(\left(\frac{T}{\sqrt{n}} + 1 \right) n + T\sqrt{n} \right) \\
& \leq m \left(n + 2C_0 \cdot \frac{\mathbb{E}f(\mathbf{x}^0) - \underline{f}}{\epsilon} \sqrt{n} + 2\sqrt{n} \right) \\
& = \mathcal{O}\left(m \times \left(n + \frac{\sqrt{n}}{\epsilon} \right) \right).
\end{aligned}
$$

This completes the proof. **Q.E.D.**

4.5.7 Proof of Corollary 2

Proof. In previous proof of Theorem 4.2.1 and Corollary 1 we already show the convergence respect to *both* gradient size and consensus error

$$
\frac{1}{T} \sum_{t=0}^{T} \mathbb{E} \| \frac{1}{m} \sum_{i=1}^{m} \nabla f^i(\mathbf{x}_i^t) \|^2 + \frac{1}{T} \sum_{t=0}^{T} \frac{1}{m} \mathbb{E} \| \mathbf{x}^t - \mathbf{1}\bar{\mathbf{x}}^t \|^2.
$$

To show our results also holds true when the gradient metric is evaluated at the "average" of the output $\bar{\mathbf{x}} = \frac{1}{m} \sum_{i=1}^{m} \mathbf{x}_i$, we only need following relations (by using Jensen's inequality).

$$
\begin{aligned}
\| \frac{1}{m} \sum_{i=1}^{m} \nabla f^i(\bar{\mathbf{x}}) \|^2 =& \| \frac{1}{m} \sum_{i=1}^{m} \nabla f^i(\bar{\mathbf{x}}) - \frac{1}{m} \sum_{i=1}^{m} \nabla f^i(\mathbf{x}_i) + \frac{1}{m} \sum_{i=1}^{m} \nabla f^i(\mathbf{x}_i) \|^2 && (4.64) \\
\overset{\text{(i)}}{\leq} & 2 \| \frac{1}{m} \sum_{i=1}^{m} \nabla f^i(\bar{\mathbf{x}}) - \frac{1}{m} \sum_{i=1}^{m} \nabla f^i(\mathbf{x}_i) \|^2 + 2 \| \frac{1}{m} \sum_{i=1}^{m} \nabla f^i(\mathbf{x}_i) \|^2 && (4.65) \\
\overset{\text{(ii)}}{\leq} & \frac{2}{m} \sum_{i=1}^{m} \| \nabla f^i(\bar{\mathbf{x}}) - \nabla f^i(\mathbf{x}_i) \|^2 + 2 \| \frac{1}{m} \sum_{i=1}^{m} \nabla f^i(\mathbf{x}_i) \|^2 && (4.66) \\
\overset{\text{(iii)}}{\leq} & \frac{2L^2}{m} \sum_{i=1}^{m} \| \mathbf{x}_i - \bar{\mathbf{x}} \|^2 + 2 \| \frac{1}{m} \sum_{i=1}^{m} \nabla f^i(\mathbf{x}_i) \|^2 && (4.67)
\end{aligned}
$$

where in (i) we use the Cauchy-Swahitz inequality, in (ii) we use the Jensen's inequality to move the average outside the Euclidean norm, and the last inequality uses the Assumption 1. The proof is complete by further combining (4.63) in Theorem 4.2.1 and similar reasoning in Corollary 1. **Q.E.D.**

4.5.8 Proof of Lemma 4.3.1

Proof.

First recall the definition of $\mathbb{E}[\cdot|\mathcal{F}_r]$ in Lemma 4.2.1, which is the expectation with respect to the random choice of sample ξ, conditioning on $\mathbf{x}^0, \cdots, \mathbf{x}^r$, $\mathbf{v}^0, \cdots, \mathbf{v}^{r-1}$ and $\mathbf{y}^0, \cdots, \mathbf{y}^{r-1}$.

Let us define a random variable u_ξ as below and u_ℓ similarly,

$$u_\xi = \frac{1}{m}\sum_{i=1}^{m} \nabla f_\xi^i(\mathbf{x}_i^r) - \frac{1}{m}\sum_{i=1}^{m} \nabla f^i(\mathbf{x}_i^r). \tag{4.68}$$

Note that u_ξ and u_ℓ are independent random variables conditioning on $\mathcal{F}$. Further, we have the following from Assumption 3

$$\mathbb{E}_\xi \left[\frac{1}{m}\sum_{i=1}^{m} \nabla f_\xi^i(\mathbf{x}_i^r) - \frac{1}{m}\sum_{i=1}^{m} \nabla f^i(\mathbf{x}_i^r) \middle| \mathcal{F}_r \right] = 0. \tag{4.69}$$

Therefore we have

$$\mathbb{E}[\langle u_\xi, u_\ell \rangle] = \mathbb{E}_{\mathcal{F}}\mathbb{E}[\langle u_\xi, u_\ell \rangle \mid \mathcal{F}] = \mathbb{E}_{\mathcal{F}}\langle \mathbb{E}[u_\xi \mid \mathcal{F}], \mathbb{E}[u_\ell \mid \mathcal{F}] \rangle = 0. \tag{4.70}$$

Following the update rule from Algorithm 4, we have the following relations for all

$\mathrm{mod}(r, q) = 0$

$$\begin{aligned}
\mathbb{E}\left\|\bar{\mathbf{y}}^r - \frac{1}{m}\sum_{i=1}^{m}\nabla f^i(\mathbf{x}_i^r)\right\|^2 &\overset{(4.31)}{=} \mathbb{E}\left\|\frac{1}{m|S_1|}\sum_{i=1}^{m}\sum_{\xi\in S_1}\nabla f_\xi^i(\mathbf{x}_i^r) - \frac{1}{m}\sum_{i=1}^{m}\nabla f^i(\mathbf{x}_i^r)\right\|^2 \\
&\overset{(i)}{=} \frac{1}{|S_1|^2}\mathbb{E}\left\|\sum_{\xi\in S_1}\left(\frac{1}{m}\sum_{i=1}^{m}\nabla f_\xi^i(\mathbf{x}_i^r) - \frac{1}{m}\sum_{i=1}^{m}\nabla f^i(\mathbf{x}_i^r)\right)\right\|^2 \\
&\overset{(ii)}{=} \frac{1}{|S_1|^2}\mathbb{E}\sum_{\xi\in S_1}\left\|\frac{1}{m}\sum_{i=1}^{m}\nabla f_\xi^i(\mathbf{x}_i^r) - \frac{1}{m}\sum_{i=1}^{m}\nabla f^i(\mathbf{x}_i^r)\right\|^2 \\
&\overset{(iii)}{=} \frac{1}{|S_1|}\mathbb{E}\left\|\frac{1}{m}\sum_{i=1}^{m}\nabla f_\xi^i(\mathbf{x}_i^r) - \frac{1}{m}\sum_{i=1}^{m}\nabla f^i(\mathbf{x}_i^r)\right\|^2 \\
&\overset{(iv)}{\leq} \frac{1}{m|S_1|}\sum_{i=1}^{m}\mathbb{E}\left\|\nabla f_\xi^i(\mathbf{x}_i^r) - \nabla f^i(\mathbf{x}_i^r)\right\|^2 \\
&\leq \frac{\sigma^2}{|S_1|},
\end{aligned}$$

where in (i) we take out the constant $|S_1|$; in (ii) we eliminate the cross terms via (4.70); in (iii) we use the fact that the following term

$$\mathbb{E}\|\frac{1}{m}\sum_{i=1}^{m}\nabla f_\xi^i(\mathbf{x}_i^r) - \frac{1}{m}\sum_{i=1}^{m}\nabla f^i(\mathbf{x}_i^r)\|^2 \tag{4.71}$$

are equal across different samples ξ; in (iv) we use the Jensen's inequality, and the last inequality follows the Assumption 4.

Similarly, we have

$$\begin{aligned}\mathbb{E}\|\mathbf{v}^r - \nabla f(\mathbf{x}^r)\|^2 \overset{(4.30)}{=} &\mathbb{E}\left\|\frac{1}{|S_1|}\sum_{\xi\in S_1}\nabla f_\xi(\mathbf{x}^r) - \nabla f(\mathbf{x}^r)\right\|^2 \\ =&\frac{1}{|S_1|}\mathbb{E}\|\nabla f_\xi(\mathbf{x}^r) - \nabla f(\mathbf{x}^r)\|^2 \\ \leq&\frac{1}{|S_1|}\sum_{i=1}^{m}\mathbb{E}\|\nabla f_\xi^i(\mathbf{x}^r) - \nabla f^i(\mathbf{x}^r)\|^2 \\ \leq&\frac{m\sigma^2}{|S_1|}.\end{aligned}$$

This completes the proof. **Q.E.D.**

4.5.9 Proof of Theorem 4.3.1

Proof.

Note that it is easy to check that Lemma 4.2.1, Lemma 4.2.2, Lemma 4.2.3 and Lemma 4.2.4 still hold true. And the quantity ϵ_1 and ϵ_2 can be determined by Lemma 4.3.1, i.e., $\epsilon_1 = \frac{\sigma^2}{|S_1|}$ and $\epsilon_2 = \frac{m\sigma^2}{|S_1|}$. Therefore, Lemma 4.2.4 can be rewritten as below if we follow Algorithm 4,

$$H(\mathbf{x}^{r+1}) - H(\mathbf{x}^0) \leq -C_1\sum_{t=0}^{r}\mathbb{E}\|\bar{\mathbf{y}}^t\|^2 - C_2\sum_{t=0}^{r}\frac{1}{m}\mathbb{E}\|\mathbf{x}^t - \mathbf{1}\bar{\mathbf{x}}^t\|^2 - C_3\sum_{t=0}^{r}\frac{1}{m}\mathbb{E}\|\mathbf{y}^t - \mathbf{1}\bar{\mathbf{y}}^t\|^2 + \epsilon_3, \tag{4.72}$$

with $\epsilon_3 = \alpha(r+1)(1+6(1+\frac{1}{\beta}))\frac{\sigma^2}{|S_1|}$.

Therefore the upper bound of the optimality gap can be derived in a similar way as

Theorem 4.2.1,

$$\begin{aligned}
&\frac{1}{T}\sum_{t=0}^{T}\mathbb{E}\|\frac{1}{m}\sum_{i=1}^{m}\nabla f^i(\mathbf{x}_i^t)\|^2+\frac{1}{T}\sum_{t=0}^{T}\frac{1}{m}\mathbb{E}\|\mathbf{x}^t-\mathbf{1}\bar{\mathbf{x}}^t\|^2\\
\leq&\frac{2}{T}\sum_{t=0}^{T}\mathbb{E}\|\bar{\mathbf{y}}^t\|^2+\frac{2}{T}\sum_{t=0}^{T}\mathbb{E}\|\bar{\mathbf{y}}^t-\frac{1}{m}\sum_{i=1}^{m}\nabla f^i(\mathbf{x}_i^t)\|^2+\frac{1}{T}\sum_{t=0}^{T}\frac{1}{m}\mathbb{E}\|\mathbf{x}^t-\mathbf{1}\bar{\mathbf{x}}^t\|^2\\
\leq&\left(\frac{16L^2}{mT}+\frac{1}{mT}\right)\sum_{t=0}^{T}\mathbb{E}\|\mathbf{x}^t-\mathbf{1}\bar{\mathbf{x}}^t\|^2+\frac{8\alpha^2L^2}{mT}\sum_{t=0}^{T}\mathbb{E}\|\mathbf{y}^t-\mathbf{1}\bar{\mathbf{y}}^t\|^2\\
&+\left(\frac{8\alpha^2L^2}{T}+\frac{2}{T}\right)\sum_{t=0}^{T}\|\bar{\mathbf{y}}^t\|^2+\frac{2}{T}\sum_{t=0}^{T}\frac{\sigma^2}{|S_1|}.
\end{aligned}$$

Further combining (4.72) we have

$$\begin{aligned}
&\frac{1}{T}\sum_{t=0}^{T}\mathbb{E}\|\frac{1}{m}\sum_{i=1}^{m}\nabla f^i(\mathbf{x}_i^t)\|^2+\frac{1}{T}\sum_{t=0}^{T}\frac{1}{m}\mathbb{E}\|\mathbf{x}^t-\mathbf{1}\bar{\mathbf{x}}^t\|^2\\
\leq&C_0\left(\frac{H(\mathbf{x}^0)-H(\mathbf{x}^{T+1})+\epsilon_3}{T}\right)+\frac{2T+2}{T}\frac{\sigma^2}{|S_1|}\\
\leq&C_0\cdot\frac{\mathbb{E}[f(\mathbf{x}^0)]-\underline{f}}{T}+C_0\cdot\frac{\alpha(T+1)(7+\frac{6}{\beta})\sigma^2}{T|S_1|}+\frac{2T+2}{T}\frac{\sigma^2}{|S_1|}.
\end{aligned}$$

After picking $|S_1|=\frac{4C_0\alpha(7+\frac{6}{\beta})\sigma^2+8\sigma^2}{\epsilon}$, we complete the proof. **Q.E.D.**

4.5.10 Proof of Corollary 4.3.1

Proof.

If we pick the following constants for Algorithm 4:

$$\begin{aligned}
&|S_1|=\frac{4C_0\alpha(7+\frac{6}{\beta})\sigma^2+8\sigma^2}{\epsilon},\ q=|S_2|=\sqrt{|S_1|},\\
&T=2\lfloor C_0\cdot\frac{\mathbb{E}f(\mathbf{x}^0)-\underline{f}}{\epsilon}\rfloor+2\geq 2C_0\cdot\frac{\mathbb{E}f(\mathbf{x}^0)-\underline{f}}{\epsilon}
\end{aligned}$$

then from Theorem 4.3.1 we can obtain

$$\frac{1}{T}\sum_{t=0}^{T}\mathbb{E}\|\frac{1}{m}\sum_{i=1}^{m}\nabla f^i(\mathbf{x}_i^t)\|^2+\frac{1}{T}\sum_{t=0}^{T}\frac{1}{m}\mathbb{E}\|\mathbf{x}^t-\mathbf{1}\bar{\mathbf{x}}^t\|^2\leq\underbrace{C_0\cdot\frac{\mathbb{E}f(\mathbf{x}^0)-\underline{f}}{T}}_{\leq\frac{\epsilon}{2}}+\frac{\epsilon}{2}\leq\epsilon.$$

Therefore we have that the per-node sample evaluations are given as

$$\lceil\frac{T}{q}\rceil\cdot|S_1|+T\cdot|S_2|\leq\left(\frac{T}{\sqrt{|S_1|}}+1\right)|S_1|+T\sqrt{|S_1|}=O(\frac{1}{\epsilon}+\frac{1}{\epsilon^{3/2}}).$$

This completes the proof. **Q.E.D.**

Chapter 5

Applications in Wireless Resources Management

There has been a growing interest in developing data-driven, and in particular deep neural network (DNN) based methods for modern communication tasks. For a few popular tasks such as power control, beamforming, and MIMO detection, these methods achieve state-of-the-art performance while requiring less computational efforts, less resources for acquiring channel state information (CSI), etc. However, it is often challenging for these approaches to learn in a dynamic environment.

This work develops a new approach that enables data-driven methods to continuously learn and optimize resource allocation strategies in a dynamic environment. Specifically, we consider an "episodically dynamic" setting where the environment statistics change in "episodes", and in each episode the environment is stationary. We propose to build the notion of continual learning (CL) into wireless system design, so that the learning model can incrementally adapt to the new episodes, *without forgetting* knowledge learned from the previous episodes. Our design is based on a novel bilevel optimization formulation which ensures certain "fairness" across different data samples. We demonstrate the effectiveness of the CL approach by integrating it with two popular DNN based models for power control and beamforming, respectively, and testing using both synthetic and ray-tracing based data sets. These numerical results show that the proposed CL approach is not only able to adapt to the new scenarios quickly

and seamlessly, but importantly, it also maintains high performance over the previously encountered scenarios as well.

5.1 Introduction

Deep learning (DL) has been successful in many applications such as computer vision [113], natural language processing [114], and recommender system [115]; see [116] for an overview. Recent works have also demonstrated that deep learning can be applied in communication systems, either by replacing an individual function module in the system (such as signal detection [6, 117], channel decoding [118], channel estimation [5, 119]), or by jointly representing the entire system [120, 121] for achieving state-of-the-art performance. Specifically, deep learning is a data-driven method in which a large amount of training data is used to train a deep neural network (DNN) for a specific task (such as power control). Once trained, such a DNN model will replace conventional algorithms to process data in real time. Existing works have shown that when the real-time data follows similar distribution as the training data, then such an approach can generate high-quality solutions for non-trivial wireless tasks [4–6,117–119,122–127], while significantly reducing real-time computation, and/or requiring only a subset of channel state information (CSI).

Dynamic environment. However, it is often challenging to use these DNN based algorithms when the environment (such as CSI and user locations) keeps changing. There are three main reasons.

1) It is well-known that naive DL based methods typically suffer from severe performance deterioration when the environment changes, that is, when the real-time data follows a different distribution than those used in the training stage [125].

2) One can adopt the *transfer learning* and/or *online learning* paradigm, by updating the DNN model according to data generated from the new environment [125]. However, these approaches usually degrade or even overwrite the previously learned models [128,129]. Therefore they are sensitive to outlier because once adapted to a transient (outlier) environment/task, its performance on the existing environment/task can degrade significantly [130]. Such kinds of behavior are particularly undesirable for wireless resource allocation tasks, because the unstable model performance would cause large

outage probability for communication users.

3) If the entire DNN is periodically retrained using all the data seen so far [130], then the training can be time and memory consuming since the number of data needed keeps growing.

Due to these challenges, it is unclear how state-of-the-art DNN based communication algorithms could properly adapt to new environments quickly without experiencing significant performance loss over previously encountered environments. Ideally, one would like to design data-driven models that can adapt to the new environment *efficiently* (i.e., by using as little resource as possible), *seamlessly* (i.e., without knowing when the environment has been changed), *quickly* (i.e., adapt well using only a small amount of data), and *continually* (i.e., without forgetting the previously learned models).

Continual Learning. In the machine learning community, continual learning (CL) has recently been proposed to address the "catastrophic forgetting phenomenon". That is, the tendency of abruptly losing the previously learned models when the current environment information is incorporated [129]. Specifically, consider the setting where different "tasks" (e.g., different CSI distributions) are revealed sequentially. Then CL aims to retain the knowledge learned from the early tasks through one of the following mechanisms: 1) regularize the most important parameters [130,131]; 2) design dynamic neural network architectures and associate neurons with tasks [132–134]; or 3) introduce a small set of memory for later training rehearsal [135–137]. However, most of the above mentioned methods require the knowledge of the *task boundaries*, that is, the time stamp where an old task terminates and a new task begins. Unfortunately, such a setting does not suit wireless communication problems well, since the wireless environment usually changes continuously, without a precise changing point. Only limited recent CL works have focused on boundary-free environments [138–140], but they all focus on proposing general-purpose tools without considering any problem-specific structures. Therefore, it is unclear whether they will be effective in wireless communication tasks.

Contributions. The *main contribution* of this paper is that we introduce the notion of CL to data-driven wireless system design, and develop a tailored CL formulation together with a training algorithm. Specifically, we consider an "episodically dynamic" setting where the environment changes in *episodes*, and within each episode the distribution of the CSIs stays relatively stationary. Our goal is to design a learning model which

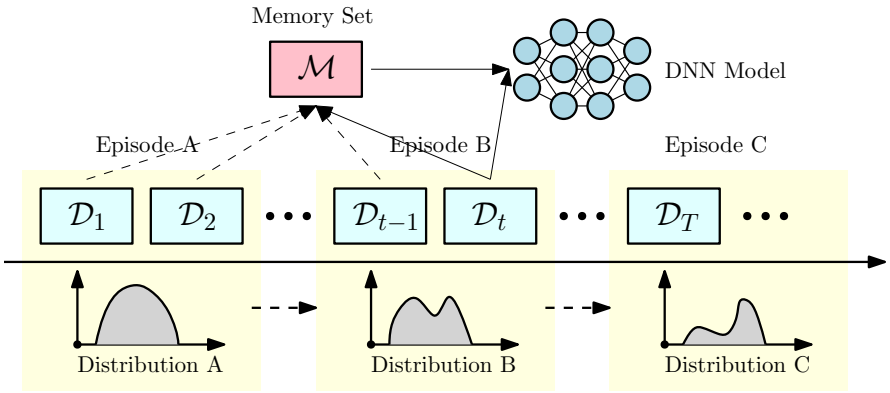


Figure 5.1: Proposed CL framework for the episodically dynamic environment. The data is feeding in a sequential manner (thus the system can only access D_t at time t) with changing episodes and distributions, and the model has a limited memory set $\mathcal{M}$ (which cannot store all data D_1 to D_t). To maintain the good performance over all experienced data from D_1 to D_t, the proposed framework optimizes the data-driven model at each time t, based on the mixture of the current data D_t and the memory set $\mathcal{M}$. The memory set $\mathcal{M}$ is then updated to incorporate the new data D_t.

can seamlessly and efficiently adapt to the changing environment, while maintaining the previously learned knowledge, and without knowing the episode boundaries.

Towards this end, we propose a CL framework for wireless systems, which incrementally adapts the DNN models by using data from the new episode as well as a limited but carefully selected subset of data from the previous episodes; see Fig. 5.1. Compared with the existing heuristic boundary-free CL algorithms [138–140], our approach is based upon a clearly defined optimization formulation that is tailored for the wireless resource allocation problem. In particular, our CL method is based on a bilevel optimization which selects a small set of important data samples into the working memory according to certain *data-sample fairness* criterion. We further relax the lower level of constrained non-convex bilevel problem using a smooth approximation, and propose and analyze practical (stochastic) algorithms for model training. Moreover, we demonstrate the effectiveness of our proposed framework by applying it to two popular DNN based models (one for power control and the other for beamforming). We test our CL approach using both synthetic and ray-tracing based data. To advocate reproducible research, the code of our implementation is available online at `https://github.com/Haoran-S/TSP_CL`.

5.2 Literature Review

5.2.1 Deep learning for Wireless Communication

Recently, DL has been used to generate high-quality solutions for non-trivial wireless communication tasks [4–6, 117–119, 122–127]. These approaches can be roughly divided into following two categories:

End-to-end Learning

For the classic resource allocation problems such as power control, the work [4] shows that DNNs can be exploited to learn the optimization algorithms such as WMMSE [141], in an end-to-end fashion. Subsequent works such as [122] and [123] show that unsupervised learning can be used to further improve the model performance. Different network structures, such as convolutional neural networks [122] and graph neural networks [124,142], and different modeling techniques, such as reinforcement learning [143], are also studied in the literature. Nevertheless, all the above mentioned methods belong to the category of end-to-end learning, where a black-box model (typically deep neural network) is applied to learn either the structure of some existing algorithms, or the optimal solution of a communication task.

Deep Unfolding

Alternatively, deep unfolding based methods [144] unfold existing optimization algorithms iteration by iteration and approximate the per-iteration behavior by one layer of the neural network. In the machine learning community, well-known works in this direction include the unfolding of the iterative soft-thresholding algorithm (ISTA) [145], unfolding of the non-negative matrix factorization methods [146], and the unfolding of the alternating direction method of multipliers (ADMM) [147]. Recently, the idea of unfolding has been used in communication task such as MIMO detection [148–150], channel coding [151], resource allocation [152], channel estimation [153], and beamforming problems [154]; see a recent survey paper [144].

5.2.2 Continual Learning

CL is originally proposed to improve reinforcement learning tasks [155] to help alleviate the catastrophic forgetting phenomenon, that is, the tendency of abruptly losing the knowledge about the previously learned task(s) when the current task information is incorporated [129]. It has later been broadly used to improve other machine learning models, and specifically the DNN models [128, 130]. Generally speaking, the CL paradigm can be classified into the following categories.

Regularization Based Methods

Based on the Bayesian theory and inspired by synaptic consolidation in Neuroscience, the regularization based methods penalize the most important parameters to retain the performance on old tasks [130]. Some most popular regularization approaches include Elastic Weight Consolidation (EWC) [130] and Learning without Forgetting (LwF) [131]. However, regularization or penalty based methods naturally introduce tradeoff between the performance of old and new tasks. If a large penalty is applied to prevent the model parameters from moving out of the optimal region of old tasks, the model may be hard to adapt to new tasks; if a small penalty is applied, it may not be sufficient to force the parameters to stay in the optimal region to retain the performance on old tasks.

Architectures Based Methods

By associating neurons with tasks (either explicitly or not), many different types of dynamic neural network architectures are proposed to address the catastrophic forgetting phenomenon [132]. However, due to the nature of the parameter isolation, architecture based methods usually require the knowledge of the task boundaries, and thus they are not suitable for wireless settings, where the environment change is often difficult to track.

Memory Based Methods

Tracing back to the 1990s, the memory (aka. rehearsal) based methods play an important role in areas such as reinforcement learning [156]. As its name suggests, memory

based methods store a small set of samples in memory for later training rehearsal, either through selecting and storing the most represented samples [135] or use generative models to generate representative samples [136]. However, all above methods require the knowledge of the task boundaries, which are not suitable for wireless settings. Only recently, the authors of [140] proposed boundary-free methods by selecting the samples through random reservoir sampling, which fills the memory set with data that is sampled from the streaming episodes uniformly at random. More complex mechanisms are also introduced recently to further increase the *sampling diversity*, where the diversity is measured by either the samples' stochastic gradient directions [139] or the samples' Euclidean distances [138].

5.2.3 Related methods

In this section, we discuss a few methods which also deal with streaming data, and compare them with the CL approach.

Online Learning

Online learning deals with the learning problems where the training data comes sequentially, and data distribution over time may or may not be consistent [157]. The ultimate goal of online learning is to minimize the cumulative loss over time, utilizing the previously learned knowledge. In particular, when data sampling is independently and identically distributed, online gradient descent is essentially the stochastic gradient descent method, and all classic complexity results can be applied. On the other hand, when data sampling is non-stationary and drifts over time, online learning methods are more likely to adapt to the most recent data, at the cost of degrading the performance on past data [158].

Transfer Learning (TL)

Different from online learning, TL is designed to apply the knowledge gained from one task to another task, based on the assumption that related tasks will benefit each other [159]. By transferring the learned knowledge from old tasks to new tasks, TL can quickly adapt to new tasks with fewer samples and less labeling effort. A typical

application is the model fine-tuning on a (potentially small) user-specific problem (e.g., MNIST classification) based on some offline pre-trained model using a comprehensive dataset (e.g. ImageNet dataset). By applying the gained knowledge from the original dataset, the model can adapt to the new dataset quickly with a few samples. Similar ideas have been applied in wireless settings recently [125,160,161] to deal with scenarios that network parameters changes. However, since the model is purely fine-tuned on the new dataset, after the knowledge transfer, the knowledge from the original model may be altered or overwritten, resulting in significant performance deterioration on the original problem [130].

5.3 The Episodic Wireless Environment

The focus of this paper is to design learning algorithms in a *dynamic* wireless environment, so that the data-driven models we build can seamlessly, efficiently, and continually adapt to new environments. This section provides details about our considered *dynamic* environment, and discuss potential challenges.

Specifically, we consider an "episodically dynamic" setting where the environment changes relatively slowly in "episodes", and during each episode the learners observe multiple *batches* of samples generated from *the same* stationary distribution; see Fig. 5.1. We use $\mathcal{D}_t$ to denote a small batch of data collected at time t, and assume that each episode k contains a set of T_k batches, and use $\mathcal{E}_k = \{\mathcal{D}_t\}_{t \in T_k}$ to denote the data collected in episode k. To have a better understanding about the setting, let us consider the following example. Again, we do not have knowledge about the episode boundaries.

5.3.1 A Motivating Example

Suppose a collection of base stations (BSs) run certain DNN based resource allocation algorithm to provide coverage for a given area (e.g., a shopping mall). The users' activities can contain two types of patterns: 1) regular but gradually changing patterns – such as daily commute for the employees and customers, and such a kind of pattern could slowly change from week to week (e.g., the store that people like to visit in the summer is different in winter); 2) irregular but important patterns – such as large events (e.g., promotion during the anniversary season), during which the distribution of user

population (and thus the CSI distribution) will be significantly different compared with their usual distributions, and more careful resource allocation has to be performed. The episode, in this case, can be defined as "a usual period of time", or "an unusual period of time that includes a particular event".

For illustration purposes, suppose that each BS solves a weighted sum-rate (WSR) maximization problem for single-input single-output (SISO) interference channel, with a maximum of K transmitter and receiver pairs. Let $h_{kk} \in \mathbb{C}$ denote the direct channel between transmitter k and receiver k, and $h_{kj} \in \mathbb{C}$ denote the interference channel from transmitter j to receiver k. The power control problem aims to maximize the weighted system throughput via allocating each transmitter's transmit power p_k. For a given snapshot of the network, the problem can be formulated as the following:

$$\begin{aligned} \max_{p_1,\ldots,p_K} \quad & R(\mathbf{p};\mathbf{h}) := \sum_{k=1}^{K} \alpha_k \log\left(1 + \frac{|h_{kk}|^2 p_k}{\sum_{j\neq k} |h_{kj}|^2 p_j + \sigma_k^2}\right) \\ \text{s.t.} \quad & 0 \leq p_k \leq P_{\max},\ \forall\, k = 1, 2, \ldots, K, \end{aligned} \tag{5.1}$$

where $P_{\max}$ denotes the power budget of each transmitter; $\{\alpha_k > 0\}$ are the weights. Problem (5.1) is known to be NP-hard [162] but can be effectively approximated by many optimization algorithms [141]. The data-driven methods proposed in recent works such as [4, 122–125] train DNNs using some pre-generated dataset. Here $\mathcal{D}_t$ can include a mini-batch of channels $\{h_{kj}\}$, and each episode can include a period of time where the channel distribution is stationary.

For illustration purposes, let us consider the following scenario. At the beginning of a period, a DNN model for solving problem (5.1) (pretrained using historical data, $\mathcal{D}_0$) is preloaded on the BSs to capture the regular patterns in the shopping mall area. The question is, what should the BSs do when the unexpected patterns appear? Say every morning a *morning model* is loaded to allocate resources up until noon. During this time the BSs can collect batches of data $\mathcal{D}_t, t = 1, 2, \cdots$. Then shall the BS update its *morning model* immediately to capture the dynamics of the user/demand distribution? If so, shall we use the entire data set, including the historical data and the real-time data, to re-train the neural network (which can be time-consuming), or shall we use TL to adapt to the new environment on the fly (which may result in overwriting the basic

morning model)?

To address the above questions, we propose to adopt the notion of CL, so that our model can incorporate the new data $\mathcal{D}_t$ on the fly, while keeping the knowledge acquired from $\mathcal{D}_{0:t-1}$. In the next section, we will detail our proposed CL formulation to achieve such a goal.

5.4 CL for Learning Wireless Resource

5.4.1 Memory-based CL

Our proposed method is based upon the notion of the *memory-based CL* proposed in [135, 138, 139], which allows the learner to collect a small subset of historical data for future re-training. The idea is, once $\mathcal{D}_t$ is received, we fill in memory $\mathcal{M}_t$ (with fixed size) with the *most representative samples* from all experienced episodes $\mathcal{D}_{0:t-1}$, and then train the neural network at each time t with the data $\mathcal{M}_t \cup \mathcal{D}_t$. Several major features of this approach are listed below:

• The learner does not need to know where a new episode starts (that is, the boundary-free setting) – it can keep updating $\mathcal{M}_t$ and keep training as data comes in.

• If one can control the size of the memory well, then the training complexity will be made much smaller than performing a complete training over the entire data set $\mathcal{D}_{0:t}$, and will be comparable with TL approach which uses $\mathcal{D}_t$.

• If the size of a given data batch $\mathcal{D}_t$ is very small, the learner is unlikely to *overfit* because the memory size is kept as fixed during the entire training process. This makes the algorithm more robust than the TL technique.

As mentioned before, existing memory-based CL methods include the random reservoir sampling algorithm [140], and sample diversity based methods [138,139,163]. However, these works have a number of drawbacks. First, for the reservoir sampling, if certain episode only contains a very small number of samples, then samples from this episode will be poorly represented in $\mathcal{M}_t$ because the probability of having samples from an episode in the memory is only dependent on the size of the episode. Second, for the diversity based methods, the approach is again heuristic, since it is not clear how the "diversity" measured by large gradient or Euclidean distances can be directly

linked to the quality of representation of the dataset. Third, and perhaps most importantly, the ways that the memory sets are selected are *independent* of the actual learning tasks at hand. The last property makes these algorithms broadly applicable to different kinds of learning tasks, but also prevents them from exploring application-specific structures. It is not clear whether, and how well these approaches will work for the wireless communication applications of interest in this paper.

5.4.2 The Proposed Approach

In this work, we propose a new memory-based CL formulation that is tailored to the wireless resource allocation problem. Our approach differs from the existing memory-based CL approaches discussed in the previous subsection, because we intend to directly use features of the learning problem at hand to build our memory selection mechanism.

To explain the proposed approach, let us begin with presenting two common ways of formulating the training problem for learning optimal wireless resource allocation. First, one can adopt an *unsupervised learning* approach, which directly optimizes some native measures of wireless system performance, such as the throughput, Quality of Service, or the user fairness [164, 165], and this approach does not need any labeled data. Specifically, a popular DNN training problem is given by

$$\min_{\mathbf{\Theta}} \sum_{i \in \mathcal{D}_{0:T}} \ell(\mathbf{\Theta}; \mathbf{h}^{(i)}), \tag{5.2}$$

where $\mathbf{h}^{(i)}$ is the ith CSI sample; $\mathbf{\Theta}$ is the DNN weight to be optimized; $\ell(\cdot)$ is the negative of the per-sample sum-rate function, that is: $\ell(\mathbf{\Theta}; \mathbf{h}^{(i)}) = -R(\pi(\mathbf{\Theta}; \mathbf{h}^{(i)}); \mathbf{h}^{(i)})$, where R is defined in (5.1) and $\pi(\mathbf{\Theta}; \mathbf{h}^{(i)})$ is the output of DNN which predicts the power allocation. The advantage of this class of unsupervised learning approach is that the system performance measure is directly built into the learning model, while the downside is that this approach can get stuck at low-quality local solutions due to the non-convex nature of DNN [166].

Secondly, it is also possible to use a supervised learning approach. Towards this end, we can generate some *labeled data* by executing a state-of-the-art optimization algorithm over all the training data samples [4]. Specifically, for each CSI vector $\mathbf{h}^{(i)}$, we can use algorithms such as the WMMSE [141] to solve problem (5.1) and obtain a

high-quality solution $\mathbf{p}^{(i)}$. Putting the $\mathbf{h}^{(i)}$ and $\mathbf{p}^{(i)}$ together yields the ith labeled data sample. Specifically, a popular supervised DNN training problem is given by

$$\min_{\boldsymbol{\Theta}} \sum_{i \in \mathcal{D}_{0:T}} \ell(\boldsymbol{\Theta}; \mathbf{h}^{(i)}, \mathbf{p}^{(i)}), \tag{5.3}$$

where $\ell(\cdot)$ can be the Mean Squared Error (MSE) loss, that is: $\ell(\boldsymbol{\Theta}; \mathbf{x}^{(i)}, \mathbf{p}^{(i)}) = \|\mathbf{p}^{(i)} - \pi(\boldsymbol{\Theta}, \mathbf{x}^{(i)})\|^2$. Such a supervised learning approach typically finds high-quality models [4,166], but often incurs significant computation cost since generating high-quality labels can be very time-consuming. Additionally, the quality of the learning model is usually limited by that of label-generating optimization algorithms.

Our idea is to leverage the advantages of both training approaches to construct a memory-based CL formulation. Specifically, we propose to select the most representative data samples $\mathbf{h}^{(i)}$'s into the working memory, by using a *sample fairness* criteria. That is, those data samples that have relatively low system performance are more likely to be selected into the memory. Meanwhile, the DNN is trained by performing either supervised or unsupervised learning over the selected data samples. We expect that as long as the learning model can perform well on these challenging and *under-performing* data samples, then it should work well for the rest of the samples in a given episode.

To proceed, let us first assume that the entire dataset $\mathcal{D}_{0:T}$ is available. Let us use $\ell(\cdot)$ to denote a function measuring the per-sample training loss, $u(\cdot)$ a loss function measuring system performance for one data sample, $\boldsymbol{\Theta}$ the weights to be trained, $\mathbf{x}^{(i)}$ the ith data sample and $\mathbf{p}^{(i)}$ the ith label. Let $\pi(\boldsymbol{\Theta}; \mathbf{x}^{(i)})$ denote the output of the neural network. Let us consider the following bilevel optimization problem

$$\min_{\boldsymbol{\Theta}} \quad \sum_{i \in \mathcal{D}_{0:T}} \lambda_*^{(i)}(\boldsymbol{\Theta}) \cdot \ell(\boldsymbol{\Theta}; \mathbf{x}^{(i)}, \mathbf{p}^{(i)}) \tag{5.4a}$$

$$\text{s.t.} \quad \boldsymbol{\lambda}_*(\boldsymbol{\Theta}) = \arg\max_{\boldsymbol{\lambda} \in \mathcal{B}} \sum_{i \in \mathcal{D}_{0:T}} \lambda^{(i)} \cdot u(\boldsymbol{\Theta}; \mathbf{x}^{(i)}, \mathbf{p}^{(i)}), \tag{5.4b}$$

where $\mathcal{B}$ denotes the simplex constraint

$$\mathcal{B} := \left\{ \boldsymbol{\lambda} \quad \Big| \sum_{i \in \mathcal{D}_{0:T}} \lambda^{(i)} = 1, \quad \lambda^{(i)} \geq 0, \quad \forall\, i \in \mathcal{D}_{0:T} \right\}.$$

In the above formulation, the upper level problem (5.4a) optimizes the *weighted* training performance across all data samples, and the lower level problem (5.4b) assigns larger weights to those data samples that have higher loss $u(\cdot)$ (or equivalently, lower system level performance). The lower level problem has a linear objective, so the optimal $\boldsymbol{\lambda}_*$ is always on the vertex of the simplex, and the non-zero elements in $\boldsymbol{\lambda}_*$ all have the same weight. Such a solution naturally selects a subset of data for the upper level training problem to optimize.

Remark 5.4.1. **(Choices of Loss Functions)** One feature of the above formulation is that we decompose the training problem and the data selection problem, so that we can have the flexibility of choosing different loss functions according to the applications at hand. Below we discuss a few alternatives.

First, the upper layer problem trains the DNN parameters $\boldsymbol{\Theta}$, so we can adopt any existing training formulation we discussed above. For example, if supervised learning is used, then one common training loss is the MSE loss:

$$\ell_{\text{MSE}}(\boldsymbol{\Theta}; \mathbf{x}^{(i)}, \mathbf{p}^{(i)}) = \|\mathbf{p}^{(i)} - \pi(\boldsymbol{\Theta}, \mathbf{x}^{(i)})\|^2. \tag{5.5}$$

Second, the lower level loss function $u(\cdot)$ can be chosen as some adaptive weighted negative sum-rate for the ith data sample, which is directly related to system performance

$$u(\boldsymbol{\Theta}, \mathbf{x}^{(i)}, \mathbf{p}^{(i)}) = -\alpha_i(\boldsymbol{\Theta}; \mathbf{x}^{(i)}, \mathbf{p}^{(i)}) \cdot R(\pi(\boldsymbol{\Theta}, \mathbf{x}^{(i)}); \mathbf{x}^{(i)}). \tag{5.6}$$

If we choose $\alpha_i(\boldsymbol{\Theta}; \mathbf{x}^{(i)}, \mathbf{p}^{(i)}) \equiv 1, \ \forall \ i$, then the channel realization that achieves the worst throughput by the current DNN model will always be selected, and the subsequent training problem will try to improve such "worst case" performance. Alternatively, when the achievable rates at samples across different episodes vary significantly (e.g., some episodes can have strong interference), then it is likely that the previous scheme will select data only from a few episodes. Alternatively, we can choose $\alpha_i(\boldsymbol{\Theta}; \mathbf{x}^{(i)}, \mathbf{p}^{(i)}) = 1/\bar{R}(\mathbf{x}^{(i)})$, where $\bar{R}(\mathbf{x}^{(i)})$ is the rate achievable by running some existing optimization algorithm on the sample $\mathbf{x}^{(i)}$. This way, the data samples that achieve the worst sum-rate "relative" to the state-of-the-art optimization algorithm is more likely to be selected. Empirically the ratio $R(\pi(\boldsymbol{\Theta}, \mathbf{x}^{(i)}); \mathbf{x}^{(i)})/\bar{R}(\mathbf{x}^{(i)})$ should be quite uniform across data

samples [4], so if there is one sample whose ratio is significantly lower than the rest, then we consider it as "underperforming" and select it into the memory.

Remark 5.4.2. **(Special Case)** As a special case of problem (5.4), one can choose $\ell(\cdot)$ to be the same as $u(\cdot)$. Then the bilevel problem reduces to the following minimax problem, which optimizes the *worst case* performance (measured by the loss $\ell(\cdot)$)) across all samples:

$$\min_{\boldsymbol{\Theta}} \max_{\boldsymbol{\lambda}\in\mathcal{B}} \quad \sum_{i\in\mathcal{D}_{0:T}} \lambda^{(i)} \cdot \ell(\boldsymbol{\Theta}; \mathbf{x}^{(i)}, \mathbf{p}^{(i)}). \tag{5.7}$$

When $\ell(\cdot)$ is taken as the negative per-sample sum-rate defined in (5.2), problem (5.7) is related to the classical minimax resource allocation [164,167,168], with the key difference that it does not achieve fairness *across users*, but rather to achieve fairness across *data samples*.

Compared to the original bilevel formulation (5.4), the minimax formulation (5.7) is more restrictive but its properties have been relatively better understood. Many recent works have been developed for solving this problem, such as the two-time-scale Gradient Descent Ascent (GDA) algorithm [169]; see [170] for a recent survey about related algorithms.

At this point, neither the bilevel problem (5.4) nor the minimax formulation (5.7) can be used to design CL strategy yet, because solving these problems requires the full data $\mathcal{D}_{0:T}$. To make these formulations useful for the considered CL setting, we make the following approximation. Suppose that at t-th time instance, we have the memory $\mathcal{M}_t$ and the new data set $\mathcal{D}_t$ available. Then, we propose to solve the following problem to select data points at time t:

$$\begin{aligned} \min_{\boldsymbol{\Theta}} \quad & \sum_{i\in\mathcal{M}_t\cup\mathcal{D}_t} \lambda_t^{(i)}(\boldsymbol{\Theta}) \cdot \ell(\boldsymbol{\Theta}; \mathbf{x}^{(i)}, \mathbf{p}^{(i)}) \\ \text{s.t.} \quad & \boldsymbol{\lambda}_t(\boldsymbol{\Theta}) = \arg\max_{\boldsymbol{\lambda}\in\mathcal{B}_t} \sum_{i\in\mathcal{M}_t\cup\mathcal{D}_t} \lambda^{(i)} \cdot u(\boldsymbol{\Theta}; \mathbf{x}^{(i)}, \mathbf{p}^{(i)}), \end{aligned} \tag{5.8}$$

where $\mathcal{B}_t$ denotes the simplex constraint

$$\mathcal{B}_t := \left\{ \boldsymbol{\lambda} \quad \Big| \sum_{i\in\mathcal{M}_t\cup\mathcal{D}_t} \lambda^{(i)} = 1, \quad \lambda^{(i)} \geq 0, \ \forall\ i \in \mathcal{M}_t \cup \mathcal{D}_t \right\}.$$

More specifically, at a given time t, we will collect M data points $j \in \mathcal{M}_t \cup \mathcal{D}_t$ whose corresponding $\lambda^{(j)}$'s are the largest. These data points will form the next memory $\mathcal{M}_{t+1}$, and problem (5.8) will be solved again. The entire procedure will be shown shortly in Algorithm 5.

5.4.3 Reformulation

In the previous section, we have proposed the CL framework and its optimization formulations. In this section, we will propose practical (stochastic) algorithms to solve those problems, and provide some basic analysis.

In general, at each time t, the non-convex bilevel problem (5.8) is very challenging to solve. Recent works on bilevel problems typically focus on solving problems with unconstrained and strongly convex inner problems [171]. However, there is no generic theoretical guarantee available when the outer problem is non-convex, and the inner problem is constrained. In this section, instead of directly solving the bilevel problem (5.8), we relax the original non-convex constrained lower level problem using a softmax function [116], which is a *smooth* approximation of the argmax function:

$$\begin{aligned} &\min_{\boldsymbol{\Theta}} \sum_{i \in \mathcal{M}_t \cup \mathcal{D}_t} \lambda_*^{(i)}(\boldsymbol{\Theta}) \cdot \ell(\boldsymbol{\Theta}; \mathbf{x}^{(i)}, \mathbf{p}^{(i)}) \\ &\text{s.t.} \quad \lambda_*^{(i)}(\boldsymbol{\Theta}) = \frac{e^{u(\boldsymbol{\Theta}; \mathbf{x}^{(i)}, \mathbf{p}^{(i)})}}{\sum_{j \in \mathcal{M}_t \cup \mathcal{D}_t} e^{u(\boldsymbol{\Theta}; \mathbf{x}^{(j)}, \mathbf{p}^{(j)})}} \in (0, 1), \quad \forall i. \end{aligned} \tag{5.9}$$

After using the above approximation, $\boldsymbol{\lambda}$ is now implicitly constrained and can be computed in a closed-form. It is clear that the obtained $\boldsymbol{\lambda}_*(\boldsymbol{\Theta})$ still allocates larger weights to larger loss values $u(\cdot)$. Further, we no longer need to solve two problems simultaneously, since we can easily obtain a single level problem by plugging the lower level problem into the upper problem.

Formally, at a given time t, problem (5.9) can be written as the following *compositional* optimization form:

$$\min_{\boldsymbol{\Theta}} \; F^t(\boldsymbol{\Theta}) = \bar{f}^t(\bar{g}^t(\boldsymbol{\Theta}); \boldsymbol{\Theta}), \tag{5.10}$$

where we have defined:

$$\bar{f}^t(\mathbf{z};\boldsymbol{\Theta}) := \frac{\sum_{i\in\mathcal{M}_t\cup\mathcal{D}_t} e^{u(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})}\cdot \ell(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})}{|\mathcal{M}_t\cup\mathcal{D}_t|\cdot\mathbf{z}}, \tag{5.11a}$$

$$\bar{g}^t(\boldsymbol{\Theta}) := \frac{1}{|\mathcal{M}_t\cup\mathcal{D}_t|}\cdot\sum_{i\in\mathcal{M}_t\cup\mathcal{D}_t} e^{u(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})}. \tag{5.11b}$$

5.4.4 Optimization Algorithms and Convergence

In this subsection, we will design algorithms for solving problem (5.10) (for a given time instance t). We first make the following standard assumptions.

Assumption 5 (Boundedness)**.** *The function value, the gradient and the Hessian of both upper level loss function $\ell(\cdot)$ and lower level loss function $u(\cdot)$ are bounded for all $\boldsymbol{\Theta}$, and for all $\mathbf{p}\in[0,\mathbf{1}\times P_{\max}]$, and all realizations of $\mathbf{h}$:*

$$\begin{aligned}
\|\ell(\boldsymbol{\Theta};\mathbf{x},\mathbf{p})\|\le C_{\ell_0}, \quad \|u(\boldsymbol{\Theta};\mathbf{x},\mathbf{p})\|\le C_{u_0},\\
\|\nabla_{\boldsymbol{\Theta}}\ell(\boldsymbol{\Theta};\mathbf{x},\mathbf{p})\|\le C_{\ell_1}, \quad \|\nabla_{\boldsymbol{\Theta}}u(\boldsymbol{\Theta};\mathbf{x},\mathbf{p})\|\le C_{u_1},\\
\left\|\nabla^2_{\boldsymbol{\Theta}}\ell(\boldsymbol{\Theta};\mathbf{x},\mathbf{p})\right\|\le C_{\ell_2}, \quad \left\|\nabla^2_{\boldsymbol{\Theta}}u(\boldsymbol{\Theta};\mathbf{x},\mathbf{p})\right\|\le C_{u_2}.
\end{aligned}$$

Remark 5.4.3. Assumption 5 is reasonable in our specific problems. We can show that it can be satisfied if we choose $\ell(\cdot)$ and $u(\cdot)$ as suggested in (5.5) and (5.6), and use a neural network $\pi(\cdot)$ that have bounded gradient and Hessian [172]. The details verifying Assumption 5 will be left to the supplemental material 5.6.2.

Since the compositional problem (5.10) is essentially a single level problem, we can update $\boldsymbol{\Theta}$ using the conventional gradient descent (GD) algorithm:

$$\begin{aligned}
\boldsymbol{\Theta}^{k+1} = \boldsymbol{\Theta}^k &- \alpha\nabla\bar{g}^t(\boldsymbol{\Theta}^k)\nabla_1\bar{f}^t(\bar{g}^t(\boldsymbol{\Theta}^k);\boldsymbol{\Theta}^k)\\
&- \alpha\nabla_2\bar{f}^t(\bar{g}^t(\boldsymbol{\Theta}^k);\boldsymbol{\Theta}^k),
\end{aligned} \tag{5.12}$$

where α is the stepsize, and the two gradients are defined as

$$\nabla_1\bar{f}^t(\mathbf{a},\cdot) := \frac{\partial\bar{f}^t(\mathbf{a},\cdot)}{\partial\mathbf{a}}, \quad \nabla_2\bar{f}^t(\cdot,\mathbf{b}) := \frac{\partial\bar{f}^t(\cdot,\mathbf{b})}{\partial\mathbf{b}},$$

for all $\mathbf{a},\mathbf{b}$ of appropriate sizes.

We have the following convergence result.

Theorem 5.4.1. *Suppose Assumption 5 hold, then the GD update* (5.12) *achieves the following convergence rate*

$$\min_{0\leq k\leq K} \|\nabla F^t(\boldsymbol{\Theta}^k)\|^2 \leq \frac{c_0 \cdot \bar{L} \cdot (F^t(\boldsymbol{\Theta}^0) - F^{t,*})}{K+1},$$

where c_0 is some universal positive constant, $\bar{L}$ is the Lipschitz constant of function $\nabla F^t(\boldsymbol{\Theta})$, and $F^{t,*}$ is the optimal value of $F^t(\cdot)$ as defined in (5.10).

Proof. From Assumption 5 we can conclude that the function F^t has Lipschitz continuous gradient with constant $\bar{L}$, where the proof and precise definition of $\bar{L}$ is relegated to Lemma 5.6.1 in the supplemental material 5.6.4. Then the desired result immediately follows from the classical gradient descent analysis on non-convex problems; see [173, Section 1.2.3].

However, the above update needs to evaluate $\bar{g}^t(\boldsymbol{\Theta})$, $\nabla\bar{g}^t(\boldsymbol{\Theta})$ and $\nabla\bar{f}^t(\mathbf{z},\boldsymbol{\Theta})$, and the evaluation of each term requires the entire dataset $\mathcal{M}_t \cup \mathcal{D}_t$. This practically means that we need to perform full GD to train a (potentially large) neural network, which is computationally expensive, and typically results in poor performance.

A more efficient solution is to perform a stochastic gradient descent (SGD) type update, which first samples a mini-batch of data, then computes stochastic gradients to update. To be specific, the algorithm samples a subset of data ξ and ϕ uniformaly randomly at each iteration from the dataset $\mathcal{M}_t \cup \mathcal{D}_t$. Then the sampled versions of $\bar{f}^t(\mathbf{z};\boldsymbol{\Theta})$ and $\bar{g}^t(\boldsymbol{\Theta})$ are given by:

$$f(\mathbf{z};\boldsymbol{\Theta};\xi) := \frac{\sum_{i\in\xi} e^{u(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})} \cdot \ell(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})}{|\xi|\cdot\mathbf{z}}, \tag{5.13a}$$

$$g(\boldsymbol{\Theta};\phi) := \frac{1}{|\phi|}\cdot\sum_{i\in\phi} e^{u(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})}, \tag{5.13b}$$

where the notations $|\phi|$ and $|\xi|$ denote the number of samples in the mini-batch ϕ and ξ, respectively. It is common to assume that the sampling mechanism can obtain ξ and ϕ randomly and independently, that is, the following unbiasedness property holds.

Assumption 6 (Unbiased Sampling)**.** *The sampling oracle satisfies the following relations, where z is a deterministic variable*

$$\begin{aligned}
\mathbb{E}_t\left[g(\boldsymbol{\Theta};\phi)\right] &= \bar{g}^t(\boldsymbol{\Theta}),\\
\mathbb{E}_t\left[\nabla g(\boldsymbol{\Theta};\phi)\right] &= \nabla \bar{g}^t(\boldsymbol{\Theta}),\\
\mathbb{E}_t\left[\nabla_1 f(\mathbf{z};\boldsymbol{\Theta};\xi)\right] &= \nabla_1 \bar{f}^t(\mathbf{z};\boldsymbol{\Theta})\\
\mathbb{E}_t\left[\nabla_2 f(\mathbf{z};\boldsymbol{\Theta};\xi)\right] &= \nabla_2 \bar{f}^t(\mathbf{z};\boldsymbol{\Theta}).
\end{aligned}$$

Note that we have used the simplified notation $\mathbb{E}_t[\cdot]$ to indicate that the expectation is taken over the sampling process from the data set $\mathcal{M}_t \cup \mathcal{D}_t$.

Based on the above assumption, problem (5.10) can be equivalently written as:

$$\min_{\boldsymbol{\Theta}} \; F^t(\boldsymbol{\Theta}) = \mathbb{E}_t[f(\mathbb{E}_t[g(\boldsymbol{\Theta};\phi)];\boldsymbol{\Theta};\xi)]. \tag{5.14}$$

Then we can write down the following stochastic update, where the update direction $\mathbf{d}^k$ is an unbiased estimator of $\nabla F^t(\boldsymbol{\Theta}^k)$:

$$\boldsymbol{\Theta}^{k+1} = \boldsymbol{\Theta}^k - \alpha \mathbf{d}^k, \tag{5.15}$$

where we have defined:

$$\mathbf{d}^k := \nabla g(\boldsymbol{\Theta}^k;\phi^k)\nabla_1 f\left(\mathbb{E}_t\left[g(\boldsymbol{\Theta}^k;\phi^k)\right];\boldsymbol{\Theta}^k;\xi^k\right) + \nabla_2 f\left(\mathbb{E}_t\left[g(\boldsymbol{\Theta}^k;\phi^k)\right];\boldsymbol{\Theta}^k;\xi^k\right).$$

Unfortunately, computing $\mathbf{d}^k$ is still costly due to the need to evaluate $\mathbb{E}_t\left[g(\boldsymbol{\Theta}^k;\phi^k)\right]$ (i.e., evaluating $\bar{g}^t(\boldsymbol{\Theta}^k)$), which still involves the full data. One can no longer directly replace $\mathbb{E}_t\left[g(\boldsymbol{\Theta}^k;\phi^k)\right]$ by its stochastic samples $g(\boldsymbol{\Theta}^k;\phi)$ because such an estimator is biased, that is:

$$\mathbb{E}_t[\nabla g(\boldsymbol{\Theta}^k;\phi^k)\nabla_1 f(g(\boldsymbol{\Theta}^k;\phi^k);\boldsymbol{\Theta}^k;\xi^k)] \neq \nabla \bar{g}^t(\boldsymbol{\Theta}^k)\nabla_1 \bar{f}^t(\bar{g}^t(\boldsymbol{\Theta}^k);\boldsymbol{\Theta}^k).$$

To proceed, we introduce an auxiliary sequence $\{\mathbf{y}^{k+1}\}$ to track $\mathbb{E}_t[g(\boldsymbol{\Theta}^k;\phi^k)]$. The

Input: Memory $\mathcal{M}_0 = \emptyset$, memory size M, max iterations R, step-sizes α, β
while *receive* $\mathcal{D}_t$ **do**
  Set $\mathcal{G}_t = \mathcal{M}_t \cup \mathcal{D}_t$
  for $k = 1 : K$ **do**
    $\boldsymbol{\Theta}^{k+1} \leftarrow \boldsymbol{\Theta}^k - \alpha_k \nabla g(\boldsymbol{\Theta}^k; \phi^k) \nabla_1 f(\mathbf{y}^{k+1}; \boldsymbol{\Theta}^k; \xi^k)$
      $-\alpha_k \nabla_2 f(\mathbf{y}^{k+1}; \boldsymbol{\Theta}^k; \xi^k)$
    $\mathbf{y}^{k+1} \leftarrow (1-\beta_k)\left(\mathbf{y}^k + g(\boldsymbol{\Theta}^k; \phi^k) - g(\boldsymbol{\Theta}^{k-1}; \phi^k)\right)$
      $+\beta_k g(\boldsymbol{\Theta}^k; \phi^k)$
  end
  if $|\mathcal{G}_t| < M$ **then**
    $\mathcal{M}_{t+1} = \mathcal{G}_t$
  else
    $\mathcal{I} = \mathrm{Top}_M(\{\lambda_t^{(i)}\}_{\forall i})$
    $\mathcal{M}_{t+1} = \{\mathcal{G}_t^{(i)}\}_{i \in \mathcal{I}}$
  end
end

Algorithm 5: The Proposed stochastic CL Algorithm

resulting SGD-type algorithm is given below:

$$\boldsymbol{\Theta}^{k+1} = \boldsymbol{\Theta}^k - \alpha_k \nabla g(\boldsymbol{\Theta}^k; \phi^k) \nabla_1 f(\mathbf{y}^{k+1}; \boldsymbol{\Theta}^k; \xi^k) - \alpha_k \nabla_2 f(\mathbf{y}^{k+1}; \boldsymbol{\Theta}^k; \xi^k), \tag{5.16a}$$

$$\mathbf{y}^{k+1} = (1-\beta_k)\left(\mathbf{y}^k + g(\boldsymbol{\Theta}^k; \phi^k) - g(\boldsymbol{\Theta}^{k-1}; \phi^k)\right) + \beta_k g(\boldsymbol{\Theta}^k; \phi^k), \tag{5.16b}$$

where $\{\alpha_k\}$ and $\{\beta_k\}$ are sequences of stepsizes. The rationale is that, if the auxiliary variable $\mathbf{y}^{k+1}$ can track the true value $\bar{g}^t(\boldsymbol{\Theta}^k)$ reasonably well, then (5.16a) will be able to approximate an unbiased estimator of the true gradient.

Finally, the overall stochastic algorithm for approximately solving problem (5.4) is given in Algorithm 5. For each time period t, we first solve the relaxed problem (5.9) in line 4-8, by performing the stochastic updates described in (5.16a) – (5.16b) for K times (where K is a predetermined number). Next, we construct the memory set $\mathcal{M}_t$ in line 9-13. We sort the elements of $\{\lambda_t^{(i)}\}$ (defined in (5.9)) and pick M largest elements' index set $\mathcal{I}$ (In line 12 of the table); Then we assign the data points associated with the index set $\mathcal{I}$ to the new memory set $\mathcal{M}_{t+1}$ (In line 13 of the table).

Remark 5.4.4. Note that the use of the auxiliary variable $\mathbf{y}$, first appeared in solving

stochastic compositional optimization problems in the form of

$$\min_{\mathbf{\Theta}} \mathbb{E}_{\xi}[f(\mathbb{E}_{\phi}[g(\mathbf{\Theta}, \phi)], \xi)], \tag{5.17}$$

see recent works [174,175], and the references therein. In particular, the authors of [175] provided the exact update form of (5.16b), and showed that the resulting algorithm enjoys the same sample efficiency as directly applying the SGD to solve problem (5.17).

However, problem (5.17) is not exactly the same as our problem (5.14) because our problem includes an extra variable $\mathbf{\Theta}$ in the definition of $f(\cdot)$, so the update (5.16a) includes an additional term $-\alpha_k \nabla_2 f(\mathbf{y}^{k+1}; \mathbf{\Theta}^k; \xi^k)$. Therefore, more refined analysis steps have to be taken compared to [175].

Below, we analyze the convergence of the $\mathbf{\Theta}$ and $\mathbf{y}$ updates given in line 4-8 of Algorithm 1. The following lemma is an immediate consequence of Assumption 5 and 6. Its proof is similar to Lemma 5.6.2 in the supplementary material Sec. 5.6.4.

Lemma 5.4.1. *Suppose Assumption 5 – 6 hold, then we have*
(1) The stochastic function $g(\mathbf{\Theta}; \phi)$ has bounded variance, that is, there exits a positive constant V_g such that:

$$\mathbb{E}_t \left[\|g(\mathbf{\Theta}; \phi) - \bar{g}^t(\mathbf{\Theta})\|^2 \right] \leq V_g, \ \forall \ \mathbf{\Theta},$$

where ϕ denotes the random data sampled from $\mathcal{M}_t \cup \mathcal{D}_t$, $\bar{g}^t$ and g are defined in (5.11) *and* (5.13)*, respectively.*
(2) The stochastic gradient of g is bounded in expectation, that is, there exists a positive constant C_g such that

$$\mathbb{E}_t \left[\|\nabla g(\mathbf{\Theta}; \phi)\| \right] \leq C_g, \ \forall \ \mathbf{\Theta}. \tag{5.18}$$

(3) Fixing any sample ϕ, the stochastic gradient of g is L_g-smooth, that is, for any $\mathbf{\Theta}, \mathbf{\Theta}' \in \mathbb{R}^d$, we have:

$$\|\nabla g(\mathbf{\Theta}; \phi) - \nabla g(\mathbf{\Theta}'; \phi)\| \leq L_g \|\mathbf{\Theta} - \mathbf{\Theta}'\|.$$

Next, we show that the tracking error of the auxiliary variable $\mathbf{y}$ is shrinking.

Lemma 5.4.2 (Tracking Error Contraction [175, Lemma 1])**.** *Consider $\mathcal{F}^k$ as the collection of random variables, i.e., $\mathcal{F}^k := \{\phi^0, \ldots, \phi^{k-1}, \xi^0, \ldots, \xi^{k-1}\}$. Suppose Assumption 5 and 6 hold, and $\mathbf{y}^{k+1}$ is generated by running iteration* (5.16b) *conditioned on $\mathcal{F}^k$. The mean square error of $\mathbf{y}^{k+1}$ satisfies*

$$\begin{aligned}&\mathbb{E}_t\left[\|\bar{g}^t(\boldsymbol{\Theta}^k) - \mathbf{y}^{k+1}\|^2 \mid \mathcal{F}^k\right]\\ \leq&(1-\beta_k)^2\|\bar{g}^t(\boldsymbol{\Theta}^{k-1}) - \mathbf{y}^k\|^2 + 4(1-\beta_k)^2 C_g^2\|\boldsymbol{\Theta}^k - \boldsymbol{\Theta}^{k-1}\|^2 + 2\beta_k^2 V_g^2,\end{aligned}$$

where C_g and V_g are defined in Lemma 5.4.1.

Proof. The above analysis is the same as the one presented for solving stochastic compositional optimization problems in the form of (5.17) (see [175, Lemma 1]). We include the steps in the supplemental material 5.6.3 for completeness.

We are now ready to show our main results about the convergence of the sequence $\{(\boldsymbol{\Theta}^k, \mathbf{y}^k)\}_{k=1}^K$ in Algorithm 1.

Theorem 5.4.2 (Convergence Analysis)**.** *Consider Algorithm 5, and fix a time instance t. Let K be the total number of iterations used at time t to update the tuple $\{(\boldsymbol{\Theta}^k, \mathbf{y}^k)\}$. Suppose Assumptions 5 and 6 hold, and that the sequence of the auxiliary variable $\{\mathbf{y}^k\}$ is bounded away from zero, i.e., $\|\mathbf{y}^k\| \geq C_y$, $\forall\ k$, for some positive constant C_y. Let us choose the stepsizes as $\alpha_k = \beta_k/L_0$, $\forall\ k$, for some appropriately chosen $L_0 > 0$ (defined in* (5.21) *in the Appendix). Then the iterates $\{\boldsymbol{\Theta}^k\}$ generated by the algorithm satisfies:*

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}_t[\|\nabla F^t(\boldsymbol{\Theta}^k)\|^2] \leq \frac{2F^t(\boldsymbol{\Theta}^0) + 2\tilde{C}}{\sqrt{K}},$$

where $\tilde{C}$ is some universal constant, dependent on Assumption 5, 6 and C_y.

Proof. The full proof is relegated to Appendix 5.6.1, and $\tilde{C}$ is defined in (5.22).

Remark 5.4.5. The key idea of the proposed method is to use an auxiliary variable $\mathbf{y}$ to track the expected value $\mathbb{E}_t[g(\boldsymbol{\Theta}; \phi)]$ (or equivalently $\bar{g}^t(\boldsymbol{\Theta})$). Lemma 5.4.2 shows that the tracking error $\|\mathbf{y} - \bar{g}^t(\boldsymbol{\Theta})\|$ is shrinking given that α and β are small. Theorem 5.4.2 implies that, for a given time instance t, the sequence $\{(\boldsymbol{\Theta}^k, \mathbf{y}^k)\}_{k=1}^K$ converges in the order of $\mathcal{O}(K^{-\frac{1}{2}})$, which is the same order achieved by generic SGD methods for non-compositional non-convex problems.

Note that compared with Theorem 5.4.1, we have made an additional assumption that the size of the iterates $\{\mathbf{y}^k\}$ is bounded away from zero. Although such an assumption cannot be verified *a priori*, in our numerical result it appears to always hold. Intuitively, this assumption makes sense since $\mathbf{y}$ tracks $\bar{g}^t(\cdot)$, and $\bar{g}^t(\cdot)$ is bounded away from zero by its definition (5.11). Therefore, as long as the tracking error is small (cf. Lemma 5.4.2), we can assume $\mathbf{y}^k$ to be bounded away from zero.

Nonetheless, we would like to emphasize that, the main contribution of this work is the development of the CL formulation and approximation problem (5.9), as well as a set of practical algorithms for solving them. The convergence analysis helps us justify our design principle, but ultimately the efficiency of the proposed formulation and algorithms has to be tested in practice. This is what we plan to do in the next section.

5.5 Experimental Results

In this section, we illustrate the performance of the proposed CL framework. We choose two applications where the end-to-end learning based DNN is used: 1) power control for weighted sum-rate (WSR) maximization problem [4] with single-input single-output (SISO) interference channel defined in (5.1); 2) coordinated beamforming problem for the millimeter wave system [176], with up to 256 antennas per BS.

5.5.1 Simulation Setup

The experiments are conducted on Ubuntu 18.04 with Python 3.6, PyTorch 1.6.0, and MATLAB R2019b on one computer with two 8-core Intel Haswell processors and 128 GB of memory. The codes are made available online through `https://github.com/Haoran-S/TSP_CL`.

5.5.2 Randomly Generated Channel

We first demonstrate the performance of our proposed framework using randomly generated channels, for a scenario with $K = 10$ transmitter-receiver pairs. We choose three standard types of random channels used in previous resource allocation literature [4,123] stated as following:

Rayleigh fading: Each channel coefficient h_{ij} is generated according to a standard normal distribution, i.e.,

$$\mathsf{Re}(h_{ij}) \sim \frac{\mathcal{N}(0,1)}{\sqrt{2}},\ \mathsf{Im}(h_{ij}) \sim \frac{\mathcal{N}(0,1)}{\sqrt{2}},\ \forall i,j \in \mathcal{K}. \tag{5.19}$$

Rician fading: Each channel coefficient h_{ij} is generated according to a Gaussian distribution with 0dB K-factor, i.e.,

$$\mathsf{Re}(h_{ij}) \sim \frac{1+\mathcal{N}(0,1)}{2},\ \mathsf{Im}(h_{ij}) \sim \frac{1+\mathcal{N}(0,1)}{2},\ \forall i,j \in \mathcal{K}.$$

Geometry channel: All transmitters and receivers are uniformly randomly distributed in a R×R area. The channel gains $|h_{ij}|^2$ follow the pathloss function

$$|h_{ij}|^2 = \frac{1}{1+d_{ij}^2}|f_{ij}|^2,\ \forall\ i,j,$$

where f_{ij} is the small-scale fading coefficient follows $\mathcal{CN}(0,1)$, d_{ij} is the distance between the ith transmitter and jth receiver.

Then we use these coefficients to generate four different episodes: the Rayleigh fading channel, the Ricean fading channel, and the geometry channel (with nodes distributed in a $10m \times 10m$ and a $50m \times 50m$ area, respectively). We use such drastically changing environments to simulate (perhaps overly harsh) "toy" scenarios. Later we will utilize real data to generate more practical scenarios. For each episode, we generate $20,000$ channel realizations for training and $1,000$ for testing. We also stacked the test data from all episodes to form a mixture test set, i.e., containing $4,000$ channel realizations. During the training stage, a total of $80,000$ channel realizations are available. A batch of $5,000$ realizations is revealed each time, and the memory space contains only $2,000$ samples from the past. That is, $|\mathcal{D}_{1:16}| = 80,000$, $|\mathcal{D}_t| = 5,000$, $|\mathcal{M}_t| = 2,000$, $\forall\ t$.

For the data-driven model, we use the end-to-end learning based fully connected neural network model as implemented in [4]. For each data batch of $5,000$ realizations at time t, we train the model $\mathbf{\Theta}_t$ using the following six different approaches for 20 epochs (with the previous model $\mathbf{\Theta}_{t-1}$ as initialization):

1. Transfer learning ("TL") [125] – update the model using the current data batch (a total of $5,000$ samples);

2. Reservoir sampling based CL ("Reservoir") [138] – update the model using both the current data batch and the memory set (a total of $7,000$ samples), where data samples in the memory set are uniformly randomly sampled from the streaming episodes;

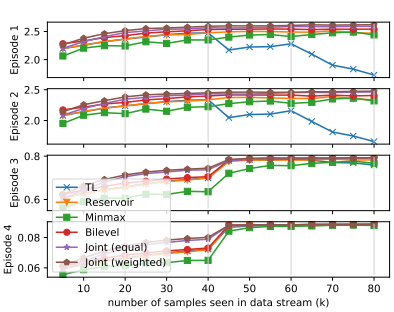
3. Proposed fairness based CL ("Bilevel") in Algorithm 5 – update the model using both the current data batch and the memory set (a total of $7,000$ samples), where data samples in the memory set are selected according to the proposed data-sample fairness criterion (5.8) using Algorithm 5. Unless otherwise specified, as suggested in Section 5.4.2, the training loss $\ell(\cdot)$ is chosen as the MSE loss (5.5), the system performance loss $u(\cdot)$ is chosen as the adaptive weighted negative sum-rate loss (5.6), and the weights is chosen as the sum-rate achievable by the WMMSE method [141].

4. Proposed minimax based CL ("Minimax") – this is the special case of Algorithm 1, as described in remark 5.4.2; In particular, we update the model using both the current data batch and the memory set (a total of $7,000$ samples), where data samples in the memory set are selected according to the proposed minimax criterion (5.7), the training loss $\ell(\cdot)$ and the system performance loss $u(\cdot)$ are chosen as the MSE loss (5.5); The model is trained using the gradient descent ascent (GDA) [169].

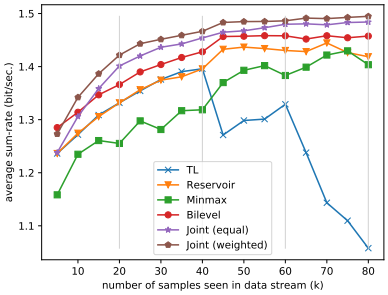
5. Joint training ("Joint (equal)") – update the model using all accumulated data up to current time (up to $80,000$ samples); All data points are treated equally, that is, $\lambda^{(i)}$ in (5.4) are equal for all i, and there is no lower level problem.

6. Joint training ("Joint (weighted)") – update the model using all accumulated data up to current time (up to $80,000$ samples); The proposed fairness based formulation (5.9) is applied but replace the training set $\mathcal{M}_t \cup \mathcal{D}_t$ with all accumulated data.

The simulation results of six different approaches are compared and shown in Fig.



(a) Test performance for each episode



(b) Average test performance for all episodes

Figure 5.2: Testing sum-rate performance on randomly generated channels for (a) each individual episode and (b) average of all episodes. Each sub-figure of (a) represents the testing performance on the data generated from a particular episode (indicated in the y-axis). The grey line indicates the time instances where a new episode starts, which is unknown during training time.

5.2. Specifically, each subplot of Fig. 5.2 (a) shows the performance of the time-varying models trained by different approaches as training data is streaming in, while evaluated at test samples drawing from the episode specified by each subplot. The grey lines indicate the transition points for two consecutive training episodes. The x-axis represents the number of training data that has been seen by the model, while the y-axis represents the sum-rate achieved on the test data. Fig. 5.2 (b) shows the average of all four subplots from Fig. 5.2 (a). Note that the joint training method uses up to $80,000$ in memory spaces and thus violates our memory limitation (i.e., $7,000$ in total), and the transfer learning method adapts the model to new data each time and does not use any additional memory spaces. One can observe that the proposed CL based methods perform well over all tasks, nearly matching the performance of the joint training method, whereas the TL suffers from some significant performance loss as the "outlier" episode comes in (i.e., geometry channel in our case).

5.5.3 Real Measured Channel

To validate our approach under more realistic scenarios, we further consider the outdoor 'O1' ray-tracing scenario generated from the DeepMIMO dataset [177]. The used dataset consists of two streets and one intersection, with the top-view showed in Fig. 5.3. The user grid is located along the horizontal street, with a length of 550m and a height of

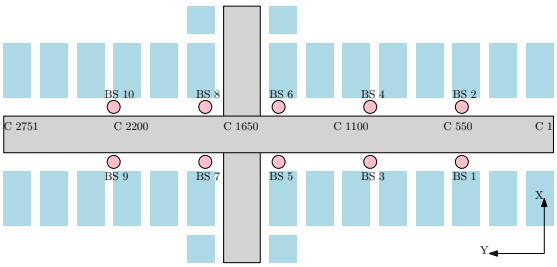
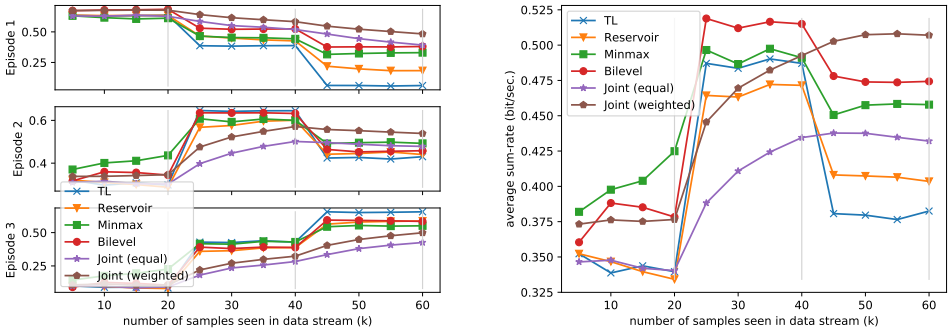


Figure 5.3: Top view of the DeepMIMO ray-tracing scenario, showing the two streets (grey rectangular), the buildings (blue rectangular), and the 10 base stations (red circle).

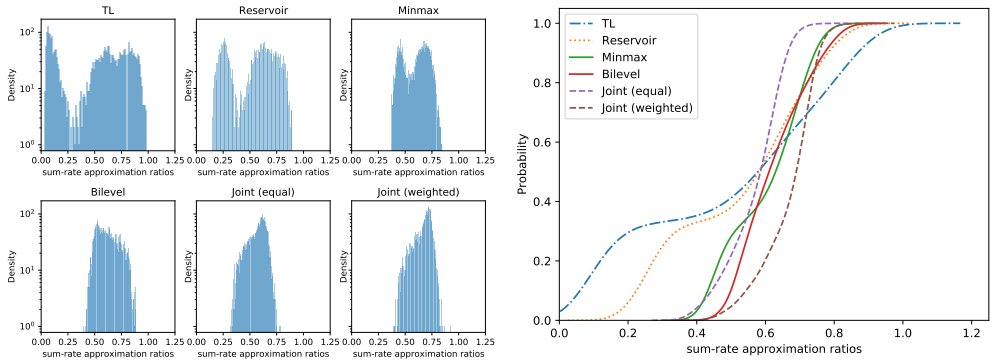


(a) Test performance for each episode (b) Average test performance for all episodes

Figure 5.4: Average sum-rate comparison on real measured channels

36m. This street is divided into a 181×2751 grid, and the users could be located on any grid point. We index the first column from the right as $C1$, and the last column from the left as $C2751$.

An episode is generated by using a particular user distribution. More specifically, the users for episode 1 are all drawn from columns C551 - C1100, and similarly, users from episode 2-3 are from C1101 - C1650, and C1651 - C2200, respectively. For each episode, we generate $20,000$ channel realizations for training and $1,000$ for testing. For each channel realization, we generate the channel based on 10 BSs (i.e., red circles in Fig. 5.3), and randomly pick $K = 10$ user locations from the selected user population. The BS is equipped with single antenna and has the maximum transmit power $p_k = 30$dBm. The noise power is set to -80 dBm.



(a) Probability Density Functions (PDF) (b) Cumulative Distribution Function (CDF)

Figure 5.5: Fairness Comparison. (a) PDF and (b) CDF distribution of the per-sample sum-rate ratio evaluated at the last time stamp ($x = 60,000$) on the test set of all three episodes.

Average sum-rate comparison

We first show the system performance measured by the achieved sum-rate for different approaches in Fig. 5.4. For each subplot of Fig. 5.4 (a), it displays a similar result as each subfigure in Fig. 5.2 (a). It can be observed that, after experiencing all the samples ($x = 60,000$), our proposed fairness based method obtains reasonable sum-rate for all three episodes, while the performance of both TL and reservoir sampling degrades when encountering test data from the old episodes. This can be attributed to the fact that the proposed method can focus on under-performing episodes (i.e. episode 1 and 2) while relaxing on outperforming episodes (i.e. episode 3). If we further average the sum-rate performance on all three episodes from Fig. 5.4 (a), we obtained Fig. 5.4 (b), in which it is clear that our proposed method is able to perform much better than TL and reservoir sampling.

Another interesting observation (from subplot 3 of Fig. 5.4 (a)) is that the proposed method is able to outperform the joint training (which uses the accumulated data) in terms of the average sum-rate, as can be seen in Fig. 5.4 (b) for $40,000 \leq x \leq 60,000$. One possible explanation is that the joint training will treat all samples equally, and thus only 1/3 of training data will contribute to improve the performance of episode 3, resulting a slow adaption to new episodes. Instead, our proposed fairness based method focuses more on data points that generate the highest cost, so it achieves higher average performance.

Fairness comparison

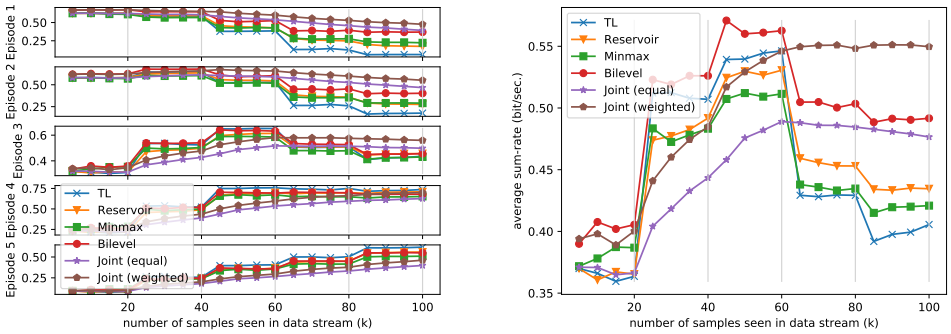
Next, we show that the proposed CL method outperforms other CL-based methods, not only in terms of the average sum-rate, but also in the sample fairness over all tasks. In Fig. 5.5, we show the test data sum-rate ratio distributions for the final models generated by different approaches (i.e., when all the models have seen all $60,000$ data points). Specifically, the sum-rate ratio $R(\pi(\mathbf{\Theta}, \mathbf{x}^{(i)}); \mathbf{x}^{(i)})/\bar{R}(\mathbf{x}^{(i)})$ is computed according to Remark 5.4.2. That is, for a given test sample $\mathbf{x}^{(i)}$, we divide the achievable sum-rate generated from the learning model, by what is achievable by the WMMSE algorithm [141]. It can be observed that our proposed approach contains fewer samples in the low sum-rate region, while TL and reservoir sampling perform worse on those data points. This result suggests that that the proposed approach indeed incorporates the problem structure and advocates fairness across the data samples.

Gradual scenario change

In the previous experiments, the sets of users from different episodes do not overlap with each other. That is, we were simulating scenarios where the environment is experiencing some *rapid* changes. In this subsection, we further simulate scenarios where the environment changes *slowly*. Towards this end, we generate episodes such that the neighboring ones share some common areas. Specifically, we have five episodes, and users for episode 1 to 5 are drawn from columns C551-C1100, C826-C1375, C1101-C1650, C1376-C1925, and C1651-C2200, respectively. Simulation results are shown in Fig. 5.6. It can be observed that our proposed methods are still effective under this setting.

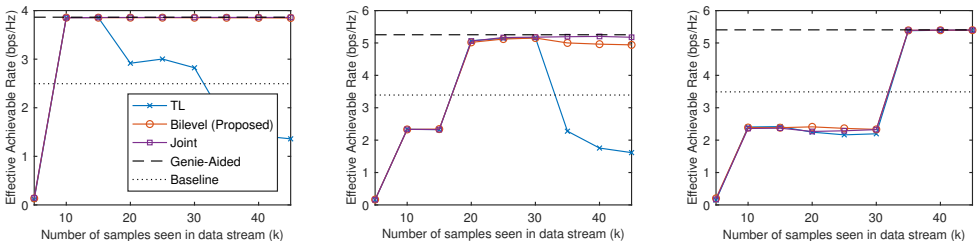
5.5.4 Beamforming Experiments

Next, we further validate our CL based approach, by applying it to a coordinated beamforming problem for the millimeter wave system, where a number of BSs are simultaneously serving one mobile user over the 60 GHz band [176]. Different from the previous sections where only single antenna is adopted, we consider the multi-antenna setup with four BSs (3,4,5,6 in Fig. 5.3), and each BS uses uniform planar array (UPA) consisting of a total of 256 antenna elements (32 columns and 8 rows), and use 30dBm transmit power.



(a) per-episode performance (b) average performance

Figure 5.6: Average sum-rate comparison on real measured channels over slowly changing environment



(a) Test Results on Episode 1 (b) Test Results on Episode 2 (c) Test Results on Episode 3

Figure 5.7: Achievable rate comparison of deep-learning coordinated beamforming strategies, genie-aided solution (perfectly knows the optimal beamforming vectors), and traditional mmWave beamforming techniques.

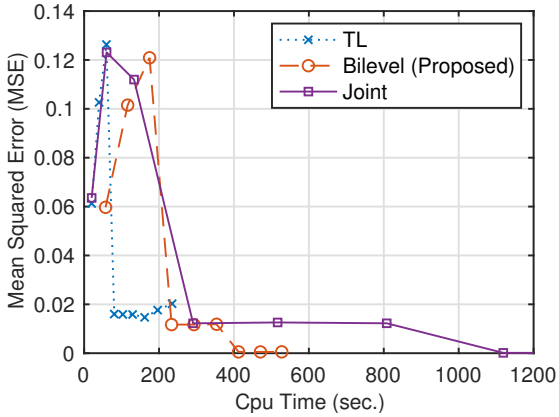


Figure 5.8: Training MSE performance over the mixed dataset of all episodes versus CPU time.

We adopt the problem formulation developed in [4, 176], where the idea is to use the uplink pilot signal received at the terminal BSs with only omni or quasi-omni beam patterns to learn and predict the best RF beamforming vectors. The learning based method we adopt is the fully connected network as suggested in [4, 176]. By leveraging the intuition that the received signal renders an RF defining signature for the user location and its interaction with the surrounding environment, the authors of [176] showed that the DL solution performs almost as well as the genie-aided solution that perfectly knows the optimal beamforming vectors.

In our simulation, we define three episodes, where the user distributions are drawn from columns C551 - C650, C826 - C925, C1101 - C1200, respectively. For each episode, we generate $15,000$ samples for training and $1,000$ for testing. The simulation results over different approaches are compared and reported in Fig. 5.7, where the x-axis represents the number of data samples that has been observed, and the y-axis denotes the effective achievable rate. For our proposed approach (Bilevel), both the training loss $\ell(\cdot)$ and the system performance loss $u(\cdot)$ are chosen as the MSE loss (5.5). It can be observed that the proposed algorithm (Bilevel) almost matches the joint training and the optimal genie-aided performances, with only limited memory usage, and it outperforms the TL approach.

Lastly, we compare the computational cost of all methods during the entire training stage. We record the required training time for all approaches when they experiencing all three episodes, then plot their achieved training loss (i.e., the MSE loss in this case, evaluated on the mixture dataset of all three episodes) versus the consumed cpu time in Fig. 5.8. It can be observed that the joint training and proposed CL approach can achieve zero training loss for all episodes (after 1,100 and 400 seconds, respectively), while the TL approach can never achieve zero training loss for all episodes although it takes less time. The proposed fairness based CL methods strike a good balance between the time complexity and the prediction accuracy.

5.6 Proofs of Lemmas and Theorems

5.6.1 Proof of Theorem 5.4.2

Proof. First, we need to establish the gradient smoothness condition of the compositional function $F^t(\mathbf{\Theta})$ as defined in (5.14). That is, for some $L > 0$, the following holds:

$$\|\nabla F^t(\mathbf{\Theta}) - \nabla F^t(\mathbf{\Theta}')\| \leq L\|\mathbf{\Theta} - \mathbf{\Theta}'\|,$$

where the gradient is computed as

$$\nabla F^t(\mathbf{\Theta}) = \nabla \bar{g}^t(\mathbf{\Theta}) \nabla_1 \bar{f}^t(\bar{g}^t(\mathbf{\Theta}), \mathbf{\Theta}) + \nabla_2 \bar{f}^t(\bar{g}^t(\mathbf{\Theta}), \mathbf{\Theta}).$$

The proof is relegated to Lemma 5.6.2 in the supplemental material 5.6.4 for completeness.

Then, using the smoothness of $\nabla F^t(\mathbf{\Theta}^k)$, we have

$$\begin{aligned}
F^t(\mathbf{\Theta}^{k+1}) \leq\ & F^t(\mathbf{\Theta}^k) + \langle \nabla F^t(\mathbf{\Theta}^k), \mathbf{\Theta}^{k+1} - \mathbf{\Theta}^k \rangle + \frac{L}{2}\|\mathbf{\Theta}^{k+1} - \mathbf{\Theta}^k\|^2 \\
\overset{(5.16a)}{=}\ & F^t(\mathbf{\Theta}^k) - \alpha_k \langle \nabla F^t(\mathbf{\Theta}^k), \nabla g(\mathbf{\Theta}^k; \phi^k) \nabla_1 f(\mathbf{y}^{k+1}, \mathbf{\Theta}; \xi^k) \rangle \\
& - \alpha_k \langle \nabla F^t(\mathbf{\Theta}^k), \nabla_2 f(\mathbf{y}^{k+1}, \mathbf{\Theta}; \xi^k) \rangle + \frac{L}{2}\|\mathbf{\Theta}^{k+1} - \mathbf{\Theta}^k\|^2 \\
=\ & F^t(\mathbf{\Theta}^k) - \alpha_k \|\nabla F^t(\mathbf{\Theta}^k)\|^2 + \frac{L}{2}\|\mathbf{\Theta}^{k+1} - \mathbf{\Theta}^k\|^2 \\
& + \alpha_k \langle \nabla F^t(\mathbf{\Theta}^k), \nabla \bar{g}^t(\mathbf{\Theta}^k) \nabla_1 \bar{f}^t(\bar{g}^t(\mathbf{\Theta}^k), \mathbf{\Theta}) \rangle \\
& - \alpha_k \langle \nabla F^t(\mathbf{\Theta}^k), \nabla g(\mathbf{\Theta}^k; \phi^k) \nabla_1 f(\mathbf{y}^{k+1}, \mathbf{\Theta}; \xi^k) \rangle \\
& + \alpha_k \langle \nabla F^t(\mathbf{\Theta}^k), \nabla_2 \bar{f}^t(\bar{g}^t(\mathbf{\Theta}^k), \mathbf{\Theta}) - \nabla_2 f(\mathbf{y}^{k+1}, \mathbf{\Theta}, \xi^k) \rangle.
\end{aligned}$$

Conditioned on $\mathcal{F}^k$, taking expectation over the sampling process of ϕ^k and ξ^k from

the data set $\mathcal{M}_t \cup \mathcal{D}_t$ on both sides, we have

$$
\begin{aligned}
\mathbb{E}_t\left[F^t(\mathbf{\Theta}^{k+1})|\mathcal{F}^k\right] \overset{(a)}{\leq} & F^t(\mathbf{\Theta}^k) - \alpha_k\|\nabla F^t(\mathbf{\Theta}^k)\|^2 + \frac{L}{2}\mathbb{E}_t[\|\mathbf{\Theta}^{k+1} - \mathbf{\Theta}^k\|^2|\mathcal{F}^k] \\
& + \alpha_k\left\|\nabla F^t(\mathbf{\Theta}^k)\right\| \mathbb{E}_t\left[\|\nabla g(\mathbf{\Theta}^k;\phi^k)\|^2|\mathcal{F}^k\right]^{\frac{1}{2}} \\
& \times \mathbb{E}_t\left[\|\nabla_1 f(g(\mathbf{\Theta}^k);\mathbf{\Theta}^k;\xi^k) - \nabla_1 f(\mathbf{y}^{k+1},\mathbf{\Theta}^k;\xi^k)\|^2|\mathcal{F}^k\right]^{\frac{1}{2}} \\
& + \alpha_k\left\|\nabla F^t(\mathbf{\Theta}^k)\right\| \\
& \times \mathbb{E}_t\left[\|\nabla_2 f(g(\mathbf{\Theta}^k);\mathbf{\Theta}^k,\xi^k) - \nabla_2 f(\mathbf{y}^{k+1};\mathbf{\Theta}^k,\xi^k)\||\mathcal{F}^k\right] \\
\overset{(b)}{\leq} & F^t(\mathbf{\Theta}^k) - \alpha_k\|\nabla F^t(\mathbf{\Theta}^k)\|^2 + LC_g^2C_{f_1}^2\alpha_k^2 + LC_{f_2}^2\alpha_k^2 \\
& + \left(\alpha_k C_g L_{f_{11}} + \alpha_k L_{f_{21}}\right)\|\nabla F^t(\mathbf{\Theta}^k)\| \\
& \times \mathbb{E}_t\left[\|g(\mathbf{\Theta}^k) - \mathbf{y}^{k+1}\|^2|\mathcal{F}^k\right]^{\frac{1}{2}} \\
\overset{(c)}{\leq} & F^t(\mathbf{\Theta}^k) - \alpha_k\|\nabla F^t(\mathbf{\Theta}^k)\|^2 + LC_g^2C_{f_1}^2\alpha_k^2 + LC_{f_2}^2\alpha_k^2 \\
& + \frac{\alpha_k^2C_g^2L_{f_{11}}^2 + \alpha_k^2L_{f_{21}}^2}{2\beta_k}\|\nabla F^t(\mathbf{\Theta}^k)\|^2 \\
& + \beta_k\mathbb{E}_t\left[\|g(\mathbf{\Theta}^k) - \mathbf{y}^{k+1}\|^2|\mathcal{F}^k\right] \\
\leq & F^t(\mathbf{\Theta}^k) - \alpha_k\left(1 - \frac{\alpha_kC_g^2L_{f_{11}}^2 + \alpha_kL_{f_{21}}^2}{2\beta_k}\right)\|\nabla F^t(\mathbf{\Theta}^k)\|^2 \\
& + \beta_k\mathbb{E}_t\left[\|g(\mathbf{\Theta}^k) - \mathbf{y}^{k+1}\|^2|\mathcal{F}^k\right] + LC_g^2C_{f_1}^2\alpha_k^2 + LC_{f_2}^2\alpha_k^2,
\end{aligned}
$$

where in (a) we use the Cauchy-Schwartz inequality; in (b) we use the update rule (5.15), the boundedness of $\|\nabla g\|, \|\nabla_1 f\|$ and $\|\nabla_2 f\|$ and the Lipschitz continuous gradient of f from Lemma 5.6.2 in the supplemental material 5.6.4; and in (c) we use the Young's inequality.

Define the Lyapunov function

$$
\mathcal{V}^k = F^t(\mathbf{\Theta}^k) + \|g(\mathbf{\Theta}^{k-1}) - \mathbf{y}^k\|^2.
$$

It follows that

$$
\begin{aligned}
&\mathbb{E}_t[\mathcal{V}^{k+1}|\mathcal{F}^k] \qquad (5.20)\\
&\leq \mathcal{V}^k - \alpha_k\left(1-\frac{\alpha_k C_g^2 L_{f_{11}}^2+\alpha_k L_{f_{21}}^2}{2\beta_k}\right)\|\nabla F^t(\mathbf{\Theta}^k)\|^2\\
&\quad + LC_g^2C_{f_1}^2\alpha_k^2 + LC_{f_2}^2\alpha_k^2\\
&\quad + (1+\beta_k)\mathbb{E}_t\left[\|g(\mathbf{\Theta}^k)-\mathbf{y}^{k+1}\|^2|\mathcal{F}^k\right] - \|g(\mathbf{\Theta}^{k-1})-\mathbf{y}^k\|^2\\
&\overset{(a)}{\leq} \mathcal{V}^k - \alpha_k\left(1-\frac{\alpha_k C_g^2 L_{f_{11}}^2+\alpha_k L_{f_{21}}^2}{2\beta_k}\right)\|\nabla F^t(\mathbf{\Theta}^k)\|^2\\
&\quad + LC_g^2C_{f_1}^2\alpha_k^2 + LC_{f_2}^2\alpha_k^2 + 2(1+\beta_k)\beta_k^2V_g^2\\
&\quad + \left((1+\beta_k)(1-\beta_k)^2-1\right)\|g(\mathbf{\Theta}^{k-1})-\mathbf{y}^k\|^2\\
&\quad + 8(1+\beta_k)(1-\beta_k)^2C_g^2\left(C_g^2C_{f_1}^2+C_{f_2}^2\right)\alpha_k^2\\
&\overset{(b)}{\leq} \mathcal{V}^k - \alpha_k\left(1-\frac{\alpha_k C_g^2 L_{f_{11}}^2+\alpha_k L_{f_{21}}^2}{2\beta_k}\right)\|\nabla F^t(\mathbf{\Theta}^k)\|^2\\
&\quad + LC_g^2C_{f_1}^2\alpha_k^2 + LC_{f_2}^2\alpha_k^2 + 2(1+\beta_k)\beta_k^2V_g^2\\
&\quad + 8C_g^2\left(C_g^2C_{f_1}^2+C_{f_2}^2\right)\alpha_k^2,
\end{aligned}
$$

where (a) follows from Lemma 5.4.2, and (b) uses that $(1+\beta_k)(1-\beta_k)^2 = (1-\beta_k^2)(1-\beta_k) \leq 1$. The corresponding constant V_g is defined in Lemma 5.4.1 and $L, C_g, C_{f_1}, C_{f_2}, L_{f_{11}}, L_{f_{21}}$ are defined in Lemma 5.6.2 in the supplemental material 5.6.4.

Select (with $\beta_k \in (0,1)$)

$$\alpha_k = \frac{\beta_k}{C_g^2L_{f_{11}}^2+L_{f_{21}}^2} := \frac{\beta_k}{L_0} \qquad (5.21)$$

so that $1-\frac{\alpha_k C_g^2 L_{f_{11}}^2+\alpha_k L_{f_{21}}^2}{2\beta_k} = \frac{1}{2}$, and define

$$
\begin{aligned}
\tilde{C} :&= LC_g^2C_{f_1}^2 + LC_{f_2}^2 + 4\left(C_g^2L_{f_{11}}^2+L_{f_{21}}^2\right)^2V_g^2\\
&\quad + 8C_g^2\left(C_g^2C_{f_1}^2+C_{f_2}^2\right). \qquad (5.22)
\end{aligned}
$$

Further taking expectation over $\mathcal{F}^k$ on both sides of (5.20), then it follows that

$$\mathbb{E}_t[\mathcal{V}^{k+1}] \leq \mathbb{E}_t[\mathcal{V}^k] - \frac{\alpha_k}{2}\mathbb{E}_t[\|\nabla F^t(\boldsymbol{\Theta}^k)\|^2] + \tilde{C}\alpha_k^2. \tag{5.23}$$

Telescoping over k and rearranging terms, we have

$$\frac{\sum_{k=0}^K \alpha_k \mathbb{E}_t[\|\nabla F^t(\boldsymbol{\Theta}^k)\|^2]}{\sum_{k=0}^K \alpha_k} \leq \frac{2\mathcal{V}^0 + 2\tilde{C}\sum_{k=0}^K \alpha_k^2}{\sum_{k=0}^K \alpha_k}.$$

Choosing the stepsize as $\alpha_k = \frac{1}{\sqrt{K}}$ leads to

$$\frac{\sum_{k=0}^{K-1} \mathbb{E}_t[\|\nabla F^t(\boldsymbol{\Theta}^k)\|^2]}{K} \leq \frac{2\mathcal{V}^0 + 2\tilde{C}}{\sqrt{K}}.$$

By initializing of $\mathbf{y}^0 = g(\boldsymbol{\Theta}^{-1})$, we have

$$\mathcal{V}^0 = F^t(\boldsymbol{\Theta}^0) + \|g(\boldsymbol{\Theta}^{-1}) - \mathbf{y}^0\|^2 = F^t(\boldsymbol{\Theta}^0).$$

The proof is complete.

5.6.2 Verify Assumptions

In this section, we show that for power allocation problem (5.1), Assumption 5 can be satisfied.

Claim 5.6.1. *Consider the power allocation problem* (5.1) *with K transmitter and receiver pairs, pick $\ell(\cdot)$ and $u(\cdot)$ as suggested in* (5.5) *and* (5.6)*. Let $\mathcal{H}$ indicate a set of bounded channel coefficients with dimension $K \times K$. Suppose that the neural network $\pi(\boldsymbol{\Theta}; \mathbf{h}) \in [0, \mathbf{p}_{\max}]$ has bounded Jacobian $J^\pi \in \mathbb{R}^{K\times d}$ and Hessian $H^\pi \in \mathbb{R}^{K\times d\times d}$ for all $\boldsymbol{\Theta} \in \mathbb{R}^d$, $\mathbf{h} \in \mathcal{H}$ and $\mathbf{p} \in [0, \mathbf{p}_{\max}]$ with $\mathbf{p}_{\max} \in \mathbb{R}_+^K$, where $J^\pi_{i,j} = \frac{\partial \pi_i(\boldsymbol{\Theta};\mathbf{h})}{\partial \boldsymbol{\Theta}_j}$, $H^\pi_{k,i,j} = \frac{\partial^2 \pi_k(\boldsymbol{\Theta};\mathbf{h})}{\partial \boldsymbol{\Theta}_i \partial \boldsymbol{\Theta}_j}$. Then Assumption 5 holds true, that is, there exist some positive*

constants $C_{\ell_0}, C_{\ell_1}, C_{\ell_2}, C_{u_0}, C_{u_1}, C_{u_2}$*, such that the following holds:*

$$\begin{aligned}
\|\ell(\boldsymbol{\Theta};\mathbf{x},\mathbf{p})\| \le C_{\ell_0}, \quad \|u(\boldsymbol{\Theta};\mathbf{x},\mathbf{p})\| \le C_{u_0},\ \forall\ \mathbf{h}\in\mathcal{H}, \forall\ \boldsymbol{\Theta}\\
\|\nabla_{\boldsymbol{\Theta}}\ell(\boldsymbol{\Theta};\mathbf{x},\mathbf{p})\| \le C_{\ell_1}, \quad \|\nabla_{\boldsymbol{\Theta}}u(\boldsymbol{\Theta};\mathbf{x},\mathbf{p})\| \le C_{u_1},\ \forall\ \mathbf{h}\in\mathcal{H}, \forall\ \boldsymbol{\Theta}\\
\left\|\nabla^2_{\boldsymbol{\Theta}}\ell(\boldsymbol{\Theta};\mathbf{x},\mathbf{p})\right\| \le C_{\ell_2}, \quad \left\|\nabla^2_{\boldsymbol{\Theta}}u(\boldsymbol{\Theta};\mathbf{x},\mathbf{p})\right\| \le C_{u_2},\ \forall\ \mathbf{h}\in\mathcal{H}, \forall\ \boldsymbol{\Theta}.
\end{aligned}$$

Proof. First, based on our specific problem (5.1), we know that the allocated power $\mathbf{p}\in[0,\mathbf{p}_{\max}]$, and output of the neural network $\pi(\boldsymbol{\Theta};\mathbf{x})\in[0,\mathbf{p}_{\max}]$ are bounded. Then we have:

$$\begin{aligned}
\ell(\boldsymbol{\Theta};\mathbf{x},\mathbf{p}) &= \|\mathbf{p}-\pi(\boldsymbol{\Theta},\mathbf{x})\|^2 \in \mathbb{R},\\
\nabla_{\boldsymbol{\Theta}}\ell(\boldsymbol{\Theta};\mathbf{x},\mathbf{p}) &= 2\left(J^{\pi(\boldsymbol{\Theta};\mathbf{x})}\right)^T(\pi(\boldsymbol{\Theta},\mathbf{x})-\mathbf{p}) \in \mathbb{R}^{d\times 1},\\
\nabla^2_{\boldsymbol{\Theta}}\ell(\boldsymbol{\Theta};\mathbf{x},\mathbf{p}) &= 2\sum_k\left[H^{\pi(\boldsymbol{\Theta};\mathbf{x})}_{k,:,:}(\pi_k(\boldsymbol{\Theta},\mathbf{x})-\mathbf{p}_k)\right] + 2(J^{\pi(\boldsymbol{\Theta};\mathbf{x})})^T J^{\pi(\boldsymbol{\Theta};\mathbf{x})} \in \mathbb{R}^{d\times d}.
\end{aligned}$$

Since by assumption, we know Jacobian J^π and Hessian H^π are bounded, then, $\ell(\boldsymbol{\Theta};\mathbf{x},\mathbf{p})$, $\nabla_{\boldsymbol{\Theta}}\ell(\boldsymbol{\Theta};\mathbf{x},\mathbf{p})$ and $\nabla^2_{\boldsymbol{\Theta}}\ell(\boldsymbol{\Theta};\mathbf{x},\mathbf{p})$ are all bounded.

Similarly, we can also show that $u(\boldsymbol{\Theta};\mathbf{x},\mathbf{p})$, $\nabla_{\boldsymbol{\Theta}}u(\boldsymbol{\Theta};\mathbf{x},\mathbf{p})$, and $\nabla^2_{\boldsymbol{\Theta}}u(\boldsymbol{\Theta};\mathbf{x},\mathbf{p})$ are bounded. Towards this end, we first compute the elements in $\nabla_\pi R(\pi(\boldsymbol{\Theta},\mathbf{x});\mathbf{x})$ and $\nabla^2_\pi R(\pi(\boldsymbol{\Theta},\mathbf{x});\mathbf{x})$:

$$\begin{aligned}
R(\pi(\boldsymbol{\Theta},\mathbf{x});\mathbf{x}) &= \sum_{k=1}^K \log\left(1+\frac{|\mathbf{x}_{kk}|^2\pi_k}{\sum_{i\neq k}|\mathbf{x}_{ki}|^2\pi_i+\sigma_k^2}\right),\\
\nabla_\pi R(\pi(\boldsymbol{\Theta},\mathbf{x});\mathbf{x}) &= \left[\nabla_{\pi_k}R(\pi(\boldsymbol{\Theta},\mathbf{x});\mathbf{x})\right]_{k=1:K},\\
\nabla_{\pi_k}R(\pi(\boldsymbol{\Theta},\mathbf{x});\mathbf{x}) &= \frac{|\mathbf{x}_{kk}|^2}{\sum_{i=1}^K|\mathbf{x}_{ki}|^2\pi_i+\sigma_k^2}\\
&\quad+\sum_{j\neq k}\frac{-|\mathbf{x}_{jj}|^2\pi_j|\mathbf{x}_{jk}|^2}{\left(\sum_{i\neq j}|\mathbf{x}_{ji}|^2\pi_i+\sigma_j^2\right)\left(\sum_{i=1}^K|\mathbf{x}_{ji}|^2\pi_i+\sigma_j^2\right)},
\end{aligned}$$

$$
\begin{aligned}
\nabla^2_{\pi} R(\pi(\mathbf{\Theta}, \mathbf{x}); \mathbf{x}) &= \left[\nabla^2_{\pi_{kt}} R(\pi(\mathbf{\Theta}, \mathbf{x}); \mathbf{x})\right]_{k=1:K, t=1:K}, \\
\nabla^2_{\pi_{kt}} R(\pi(\mathbf{\Theta}, \mathbf{x}); \mathbf{x})|_{t \neq k} &= \frac{-|\mathbf{x}_{kk}|^2 |\mathbf{x}_{kt}|^2}{\left(\sum_{i=1}^{K} |\mathbf{x}_{ki}|^2 \pi_i + \sigma_k^2\right)^2} \\
&\quad + \sum_{j \neq k,t} \frac{|\mathbf{x}_{jj}|^2 \pi_j |\mathbf{x}_{jk}|^2 |\mathbf{x}_{jt}|^2 \left(\sum_{i \neq j} |\mathbf{x}_{ji}|^2 \pi_i + \sum_{i=1}^{K} |\mathbf{x}_{ji}|^2 \pi_i + 2\sigma_j^2\right)}{\left(\sum_{i \neq j} |\mathbf{x}_{ji}|^2 \pi_i + \sigma_j^2\right)^2 \left(\sum_{i=1}^{K} |\mathbf{x}_{ji}|^2 \pi_i + \sigma_j^2\right)^2} \\
&\quad - \frac{|\mathbf{x}_{tt}|^2 |\mathbf{x}_{tk}|^2 \left(\sum_{i \neq t} |\mathbf{x}_{ti}|^2 \pi_i + \sigma_t^2\right)}{\left(\sum_{i \neq t} |\mathbf{x}_{ti}|^2 \pi_i + \sigma_t^2\right) \left(\sum_{i=1}^{K} |\mathbf{x}_{ti}|^2 \pi_i + \sigma_t^2\right)^2}, \\
\nabla^2_{\pi_{kt}} R(\pi(\mathbf{\Theta}, \mathbf{x}); \mathbf{x})|_{t = k} &= \frac{-|\mathbf{x}_{kk}|^2 |\mathbf{x}_{kt}|^2}{\left(\sum_{i=1}^{K} |\mathbf{x}_{ki}|^2 \pi_i + \sigma_k^2\right)^2} \\
&\quad + \sum_{j \neq k} \frac{|\mathbf{x}_{jj}|^2 \pi_j |\mathbf{x}_{jk}|^2 |\mathbf{x}_{jt}|^2 \left(\sum_{i \neq j} |\mathbf{x}_{ji}|^2 \pi_i + \sum_{i=1}^{K} |\mathbf{x}_{ji}|^2 \pi_i + 2\sigma_j^2\right)}{\left(\sum_{i \neq j} |\mathbf{x}_{ji}|^2 \pi_i + \sigma_j^2\right)^2 \left(\sum_{i=1}^{K} |\mathbf{x}_{ji}|^2 \pi_i + \sigma_j^2\right)^2}.
\end{aligned}
$$

Because all elements of $\mathbf{h}_{ij}$, π_i, and σ_k are bounded, it is clear that both $\nabla_{\pi} R(\pi(\mathbf{\Theta}, \mathbf{x}); \mathbf{x})$ and $\nabla^2_{\pi} R(\pi(\mathbf{\Theta}, \mathbf{x}); \mathbf{x})$ are bounded.

Next, we derive the bounds for $u(\mathbf{\Theta}; \mathbf{x}, \mathbf{p})$, $\nabla_{\mathbf{\Theta}} u(\mathbf{\Theta}; \mathbf{x}, \mathbf{p})$, and $\nabla^2_{\mathbf{\Theta}} u(\mathbf{\Theta}; \mathbf{x}, \mathbf{p})$,

$$
\begin{aligned}
u(\mathbf{\Theta}; \mathbf{x}, \mathbf{p}) &= R(\pi(\mathbf{\Theta}, \mathbf{x}); \mathbf{x}) = \sum_{k=1}^{K} \alpha_k \log\left(1 + \frac{|\mathbf{x}_{kk}|^2 \pi_k}{\sum_{j \neq k} |\mathbf{x}_{kj}|^2 \pi_j + \sigma_k^2}\right), \\
\nabla_{\mathbf{\Theta}} u(\mathbf{\Theta}; \mathbf{x}, \mathbf{p}) &= (J^{\pi(\mathbf{\Theta}; \mathbf{x})})^T \cdot \nabla_{\pi} R(\pi(\mathbf{\Theta}, \mathbf{x}); \mathbf{x}), \\
\nabla^2_{\mathbf{\Theta}} u(\mathbf{\Theta}; \mathbf{x}, \mathbf{p}) &= \sum_{k} \left[(H^{\pi(\mathbf{\Theta}; \mathbf{x})}_{k,:,:})^T \cdot \nabla_{\pi_k} R(\pi(\mathbf{\Theta}, \mathbf{x}); \mathbf{x})\right] \\
&\quad + (J^{\pi(\mathbf{\Theta}; \mathbf{x})})^T \cdot \nabla^2_{\pi} R(\pi(\mathbf{\Theta}, \mathbf{x}); \mathbf{x}) \cdot J^{\pi(\mathbf{\Theta}; \mathbf{x})},
\end{aligned}
$$

given all of J^{π}, H^{π}, $\nabla_{\pi} R(\pi(\mathbf{\Theta}, \mathbf{x}); \mathbf{x})$ and $\nabla^2_{\pi} R(\pi(\mathbf{\Theta}, \mathbf{x}); \mathbf{x})$ are bounded, the proof is complete. Finally, we note that the assumption of boundedness of Jacobian J^{π} and Hessian H^{π} are reasonable; see for example [172, Theorem 3.2], where the Lipschitz continuous and Lipschitz continuous gradient constants of neural networks are explicitly characterized.

5.6.3 Proof of Lemma 5.4.2

Claim 5.6.2 (Tracking Error Contraction [175, Lemma 1])**.** *Consider $\mathcal{F}^k$ as the collection of random variables, i.e., $\mathcal{F}^k := \{\phi^0, \ldots, \phi^{k-1}, \xi^0, \ldots, \xi^{k-1}\}$. Suppose Assumption 6 and 5 hold, and $\mathbf{y}^{k+1}$ is generated by running iteration* (5.16b) *conditioned $\mathcal{F}^k$. The mean square error of $\mathbf{y}^{k+1}$ satisfies*

$$\begin{aligned}&\mathbb{E}_t\left[\|\bar{g}^t(\boldsymbol{\Theta}^k) - \mathbf{y}^{k+1}\|^2 \mid \mathcal{F}^k\right]\\ \leq&(1-\beta_k)^2\|\bar{g}^t(\boldsymbol{\Theta}^{k-1}) - \mathbf{y}^k\|^2 + 4(1-\beta_k)^2 C_g^2\|\boldsymbol{\Theta}^k - \boldsymbol{\Theta}^{k-1}\|^2 + 2\beta_k^2 V_g^2,\end{aligned}$$

where C_g and V_g are defined in Lemma 5.4.1.

Proof. From the update (5.16b), we have that

$$\begin{aligned}&\mathbf{y}^{k+1} - \bar{g}^t(\boldsymbol{\Theta}^k)\\ =&(1-\beta_k)(\mathbf{y}^k - \bar{g}^t(\boldsymbol{\Theta}^{k-1})) + (1-\beta_k)(\bar{g}^t(\boldsymbol{\Theta}^{k-1}) - \bar{g}^t(\boldsymbol{\Theta}^k)) + \beta_k(g(\boldsymbol{\Theta}^k;\phi^k) - \bar{g}^t(\boldsymbol{\Theta}^k))\\ &+(1-\beta_k)(g(\boldsymbol{\Theta}^k;\phi^k) - g(\boldsymbol{\Theta}^{k-1};\phi^k))\\ =&(1-\beta_k)(\mathbf{y}^k - \bar{g}^t(\boldsymbol{\Theta}^{k-1})) + (1-\beta_k)T_1 + \beta_k T_2 + (1-\beta_k)T_3, \qquad (5.24)\end{aligned}$$

where we define the three terms as

$$T_1 := \bar{g}^t(\boldsymbol{\Theta}^{k-1}) - \bar{g}^t(\boldsymbol{\Theta}^k) \quad T_2 := g(\boldsymbol{\Theta}^k;\phi^k) - \bar{g}^t(\boldsymbol{\Theta}^k) \quad T_3 := g(\boldsymbol{\Theta}^k;\phi^k) - g(\boldsymbol{\Theta}^{k-1};\phi^k).$$

Conditioned on $\mathcal{F}^k$, taking expectation over the sampling process of ϕ^k from the data set $\mathcal{M}_t \cup \mathcal{D}_t$, we have

$$\mathbb{E}_t\left[(1-\beta_k)T_1 + \beta_k T_2 + (1-\beta_k)T_3|\mathcal{F}^k\right] = \mathbf{0} \quad \text{and} \quad \mathbb{E}_t\left[T_2|\mathcal{F}^k\right] = \mathbf{0}. \qquad (5.25)$$

Therefore, taking a conditional expectation on the norm square of both sides of (5.24), we have

$$
\begin{aligned}
&\mathbb{E}_t[\|\mathbf{y}^{k+1} - \bar{g}^t(\mathbf{\Theta}^k)\|^2|\mathcal{F}^k] \\
\overset{(5.24)}{=}&\mathbb{E}_t[\|(1-\beta_k)(\mathbf{y}^k - \bar{g}^t(\mathbf{\Theta}^{k-1}))\|^2|\mathcal{F}^k] + \mathbb{E}_t\left[\|(1-\beta_k)T_1 + \beta_k T_2 + (1-\beta_k)T_3\|^2|\mathcal{F}^k\right] \\
&+ 2\mathbb{E}_t\left[\left\langle (1-\beta_k)(\mathbf{y}^k - \bar{g}^t(\mathbf{\Theta}^{k-1})), (1-\beta_k)T_1 + \beta_k T_2 + (1-\beta_k)T_3\right\rangle |\mathcal{F}^k\right] \\
\overset{(5.25)}{=}&(1-\beta_k)^2\|\mathbf{y}^k - \bar{g}^t(\mathbf{\Theta}^{k-1})\|^2 + \mathbb{E}_t\left[\|(1-\beta_k)T_1 + \beta_k T_2 + (1-\beta_k)T_3\|^2|\mathcal{F}^k\right] \\
\overset{\text{(i)}}{\leq}&(1-\beta_k)^2\|\mathbf{y}^k - \bar{g}^t(\mathbf{\Theta}^{k-1})\|^2 + 2\mathbb{E}_t\left[\|(1-\beta_k)T_1 + \beta_k T_2\|^2|\mathcal{F}^k\right] + 2(1-\beta_k)^2\mathbb{E}_t\left[\|T_3\|^2|\mathcal{F}^k\right] \\
=&(1-\beta_k)^2\|\mathbf{y}^k - \bar{g}^t(\mathbf{\Theta}^{k-1})\|^2 + 2(1-\beta_k)^2\mathbb{E}_t[\|T_1\|^2 \mid \mathcal{F}^k] + 2\beta_k^2\mathbb{E}_t[\|T_2\|^2 \mid \mathcal{F}^k] \\
&+ 4\beta_k(1-\beta_k)\left\langle T_1, \mathbb{E}_t[T_2 \mid \mathcal{F}^k]\right\rangle + 2(1-\beta_k)^2\mathbb{E}_t\left[\|T_3\|^2|\mathcal{F}^k\right] \\
\overset{\text{(ii)}}{\leq}&(1-\beta_k)^2\|\mathbf{y}^k - \bar{g}^t(\mathbf{\Theta}^{k-1})\|^2 + 2(1-\beta_k)^2\mathbb{E}_t\left[\|\bar{g}^t(\mathbf{\Theta}^k) - \bar{g}^t(\mathbf{\Theta}^{k-1})\|^2|\mathcal{F}^k\right] \\
&+ 2(1-\beta_k)^2\mathbb{E}_t\left[\|g(\mathbf{\Theta}^k;\phi^k) - g(\mathbf{\Theta}^{k-1};\phi^k)\|^2|\mathcal{F}^k\right] + 2\beta_k^2 V_g^2 \\
\overset{\text{(iii)}}{\leq}&(1-\beta_k)^2\|\mathbf{y}^k - \bar{g}^t(\mathbf{\Theta}^{k-1})\|^2 + 4(1-\beta_k)^2 C_g^2\|\mathbf{\Theta}^k - \mathbf{\Theta}^{k-1}\|^2 + 2\beta_k^2 V_g^2,
\end{aligned}
$$

where in (i) we use the Cauchy–Schwartz inequality, in (ii) we use the bounded variance property from Lemma 5.4.1 and the unbiasedness (5.25), and in (iii) we use the property that the $\bar{g}^t(\mathbf{\Theta})$ and $g(\mathbf{\Theta};\phi)$ are Lipschitz continuous from (5.18). The proof is then complete.

5.6.4 Additional Lemmas

Lemma 5.6.1. *Suppose Assumption 5 holds, then function $F^t(\cdot)$ has Lipshictz continuous gradient with some universal constant $\bar{L}$, where*

$$\bar{L} := \bar{C}_{f_1}\bar{L}_g + \bar{C}_g^2\bar{L}_{f_{11}} + \bar{C}_g\bar{L}_{f_{12}} + \bar{C}_g\bar{L}_{f_{21}} + \bar{L}_{f_{22}},$$

and

$$\bar{L}_{f_{11}} := 2C_{l_0}e^{4C_{u_0}}, \quad \bar{L}_{f_{12}} = \bar{L}_{f_{21}} := C_{u_1}C_{l_0}e^{3C_{u_0}} + C_{l_1}e^{3C_{u_0}},$$
$$\bar{L}_{f_{22}} := \left(C_{u_1}^2 C_{\ell_0} + C_{u_2}C_{\ell_0} + 2C_{u_1}C_{\ell_1} + C_{\ell_2}\right)e^{2C_{u_0}}, \quad \bar{L}_g := C_{u_1}^2 e^{C_{u_0}} + C_{u_2}e^{C_{u_0}},$$
$$\bar{C}_{f_1} := C_{l_0}e^{3C_{u_0}}, \quad \bar{C}_{f_2} := C_{u_1}C_{l_0}e^{2C_{u_0}} + C_{l_1}e^{2C_{u_0}}, \quad \bar{C}_g := C_{u_1}e^{C_{u_0}}.$$

Proof. To begin with, we show that $\bar{f}^t$ and $\bar{g}^t$ are bounded. Given Assumption 5 and $\bar{f}^t, \bar{g}^t$ are defined as (5.11), we have

$$\bar{g}^t(\boldsymbol{\Theta}) := \frac{1}{|\mathcal{D}|}\cdot\sum_{i\in\mathcal{D}} e^{u(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})} \in \left[e^{-C_{u_0}}, e^{C_{u_0}}\right],$$
$$\bar{f}^t(\bar{g}^t(\boldsymbol{\Theta}');\boldsymbol{\Theta}) := \frac{\sum_{i\in\mathcal{D}} e^{u(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})}\cdot\ell(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})}{|\mathcal{D}|\cdot\bar{g}^t(\boldsymbol{\Theta}')} \in \left[-C_{l_0}e^{2C_{u_0}}, C_{l_0}e^{2C_{u_0}}\right],$$

where $\mathcal{D} := \mathcal{M}_t \cup \mathcal{D}_t$, and C_{ℓ_0}, C_{u_0} are defined in xxx. Note that we abused the notation a bit by omitting the subscript pf t when defining $\mathcal{D}$.

Next, we show that the gradients of $\bar{f}^t$ and $\bar{g}^t$ are bounded. We obtain their gradients as following

$$\nabla\bar{g}^t(\boldsymbol{\Theta}) = \frac{1}{|\mathcal{D}|}\cdot\sum_{i\in\mathcal{D}}\left(e^{u(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})}\cdot\nabla u(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})\right),$$
$$\nabla_1\bar{f}^t(z;\boldsymbol{\Theta}) = -\frac{\sum_{i\in\mathcal{D}} e^{u(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})}\cdot\ell(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})}{|\mathcal{D}|\cdot z^2},$$
$$\nabla_2\bar{f}^t(z;\boldsymbol{\Theta}) = \frac{\sum_{i\in\mathcal{D}}\left(e^{u(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})}\cdot\nabla u(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})\cdot\ell(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})\right)}{|\mathcal{D}|\cdot z} + \frac{\sum_{i\in\mathcal{D}}\left(e^{u(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})}\cdot\nabla\ell(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})\right)}{|\mathcal{D}|\cdot z}.$$

Combining the boundedness property from Assumption 5, we conclude that for any $\boldsymbol{\Theta}$ and $\boldsymbol{\Theta}'$,

$$\begin{aligned}\left\|\nabla\bar{g}^t(\boldsymbol{\Theta})\right\| &\le C_{u_1}e^{C_{u_0}} := \bar{C}_g,\\ \left\|\nabla_1\bar{f}^t(\bar{g}^t(\boldsymbol{\Theta}');\boldsymbol{\Theta})\right\| &\le C_{l_0}e^{3C_{u_0}} := \bar{C}_{f_1},\\ \left\|\nabla_2\bar{f}^t(\bar{g}^t(\boldsymbol{\Theta}');\boldsymbol{\Theta})\right\| &\le C_{u_1}C_{l_0}e^{2C_{u_0}} + C_{l_1}e^{2C_{u_0}} := \bar{C}_{f_2}.\end{aligned} \tag{5.27}$$

Next, we show that functions $\nabla \bar{f}^t$ and $\nabla \bar{g}^t$ are L_f- and L_g-smooth by bounding the Hessian of $\|\bar{f}^t\|$ and $\|\bar{g}^t\|$, where we have

$$\begin{aligned}
\nabla^2_{11}\bar{f}^t(\mathbf{z};\boldsymbol{\Theta}) &= \frac{2\cdot\sum_{i\in\mathcal{D}} e^{u(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})}\cdot \ell(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})}{|\mathcal{D}|\cdot\mathbf{z}^3},\\
\nabla^2_{12}\bar{f}^t(\mathbf{z};\boldsymbol{\Theta}) &= -\frac{\sum_{i\in\mathcal{D}} e^{u(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})}\cdot \nabla u(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})\cdot \ell(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})}{|\mathcal{D}|\cdot\mathbf{z}^2}\\
&\quad - \frac{\sum_{i\in\mathcal{D}} e^{u(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})}\cdot \nabla \ell(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})}{|\mathcal{D}|\cdot\mathbf{z}^2},\\
\nabla^2_{21}\bar{f}^t(\mathbf{z};\boldsymbol{\Theta}) &= -\frac{\sum_{i\in\mathcal{D}} \left(e^{u(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})}\cdot \nabla u(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})\cdot \ell(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})\right)}{|\mathcal{D}|\cdot\mathbf{z}^2}\\
&\quad - \frac{\sum_{i\in\mathcal{D}} \left(e^{u(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})}\cdot \nabla \ell(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})\right)}{|\mathcal{D}|\cdot\mathbf{z}^2},
\end{aligned}$$

$$\begin{aligned}
\nabla^2_{22}\bar{f}^t(\mathbf{z};\boldsymbol{\Theta}) &= \frac{\sum_{i\in\mathcal{D}} \left(e^{u(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})}\cdot \left\|\nabla u(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})\right\|^2\cdot \ell(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})\right)}{|\mathcal{D}|\cdot\mathbf{z}}\\
&\quad + \frac{\sum_{i\in\mathcal{D}} \left(e^{u(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})}\cdot \nabla^2 u(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})\cdot \ell(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})\right)}{|\mathcal{D}|\cdot\mathbf{z}}\\
&\quad + \frac{2\cdot\sum_{i\in\mathcal{D}} \left(e^{u(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})}\cdot \nabla u(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})\cdot \nabla \ell(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})\right)}{|\mathcal{D}|\cdot\mathbf{z}}\\
&\quad + \frac{\sum_{i\in\mathcal{D}} \left(e^{u(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})}\cdot \nabla^2 \ell(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})\right)}{|\mathcal{D}|\cdot\mathbf{z}},\\
\nabla^2\bar{g}^t(\boldsymbol{\Theta}) &= \frac{1}{|\mathcal{D}|}\cdot\sum_{i\in\mathcal{D}}\left(e^{u(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})}\cdot\left\|\nabla u(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})\right\|^2 + e^{u(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})}\cdot\nabla^2 u(\boldsymbol{\Theta};\mathbf{x}^{(i)},\mathbf{p}^{(i)})\right).
\end{aligned}$$

Further considering that the Assumption 5, we can conclude that for any $\boldsymbol{\Theta}$ and $\boldsymbol{\Theta}'$

$$\begin{aligned}
\left\|\nabla_{11}^2 \bar{f}^t(\bar{g}^t(\boldsymbol{\Theta}');\boldsymbol{\Theta})\right\| &\leq 2C_{l_0}e^{4C_{u_0}} := \bar{L}_{f_{11}},\\
\left\|\nabla_{12}^2 \bar{f}^t(\bar{g}^t(\boldsymbol{\Theta}');\boldsymbol{\Theta})\right\| &\leq C_{u_1}C_{l_0}e^{3C_{u_0}} + C_{l_1}e^{3C_{u_0}} := \bar{L}_{f_{12}},\\
\left\|\nabla_{21}^2 \bar{f}^t(\bar{g}^t(\boldsymbol{\Theta}');\boldsymbol{\Theta})\right\| &\leq C_{u_1}C_{l_0}e^{3C_{u_0}} + C_{l_1}e^{3C_{u_0}} := \bar{L}_{f_{21}},\\
\left\|\nabla_{22}^2 \bar{f}^t(\bar{g}^t(\boldsymbol{\Theta}');\boldsymbol{\Theta})\right\| &\leq \left(C_{u_1}^2C_{\ell_0} + C_{u_2}C_{\ell_0} + 2C_{u_1}C_{\ell_1} + C_{\ell_2}\right)e^{2C_{u_0}} := \bar{L}_{f_{22}},\\
\left\|\nabla^2 \bar{g}^t(\boldsymbol{\Theta})\right\| &\leq C_{u_1}^2e^{C_{u_0}} + C_{u_2}e^{C_{u_0}} := \bar{L}_g.
\end{aligned}$$

Or in other words, when $z = \bar{g}^t(\cdot)$, the following holds true:

$$\begin{aligned}
\|\nabla_1 \bar{f}^t(\mathbf{z},\cdot) - \nabla_1 \bar{f}^t(\mathbf{z}',\cdot)\| &\leq \bar{L}_{f_{11}}\|\mathbf{z}-\mathbf{z}'\|,\\
\|\nabla_1 \bar{f}^t(\mathbf{z},\boldsymbol{\Theta}) - \nabla_1 \bar{f}^t(\mathbf{z},\boldsymbol{\Theta}')\| &\leq \bar{L}_{f_{12}}\|\boldsymbol{\Theta}-\boldsymbol{\Theta}'\|,\\
\|\nabla_2 \bar{f}^t(\mathbf{z},\cdot) - \nabla_2 \bar{f}^t(\mathbf{z}',\cdot)\| &\leq \bar{L}_{f_{21}}\|\mathbf{z}-\mathbf{z}'\|,\\
\|\nabla_2 \bar{f}^t(\mathbf{z},\boldsymbol{\Theta}) - \nabla_2 \bar{f}^t(\mathbf{z},\boldsymbol{\Theta}')\| &\leq \bar{L}_{f_{22}}\|\boldsymbol{\Theta}-\boldsymbol{\Theta}'\|,\\
\|\nabla \bar{g}^t(\boldsymbol{\Theta}) - \nabla \bar{g}^t(\boldsymbol{\Theta}')\| &\leq \bar{L}_g\|\boldsymbol{\Theta}-\boldsymbol{\Theta}'\|.
\end{aligned} \tag{5.28}$$

Then, we are ready to establish the smoothness condition of the gradient of the compositional function $F^t(\boldsymbol{\Theta}) = \bar{f}^t(\bar{g}^t(\boldsymbol{\Theta}),\boldsymbol{\Theta})$, we have

$$\nabla F^t(\boldsymbol{\Theta}) = \nabla \bar{g}^t(\boldsymbol{\Theta})\nabla_1 \bar{f}^t(\bar{g}^t(\boldsymbol{\Theta}),\boldsymbol{\Theta}) + \nabla_2 \bar{f}^t(\bar{g}^t(\boldsymbol{\Theta}),\boldsymbol{\Theta}),$$

and

$$\begin{aligned}
\|\nabla F^t(\boldsymbol{\Theta}) - \nabla F^t(\boldsymbol{\Theta}')\| \leq& \|\nabla \bar{g}^t(\boldsymbol{\Theta})\nabla_1 \bar{f}^t(\bar{g}^t(\boldsymbol{\Theta}),\boldsymbol{\Theta}) - \nabla \bar{g}^t(\boldsymbol{\Theta}')\nabla_1 \bar{f}^t(\bar{g}^t(\boldsymbol{\Theta}),\boldsymbol{\Theta})\|\\
&+ \|\nabla \bar{g}^t(\boldsymbol{\Theta}')\nabla_1 \bar{f}^t(\bar{g}^t(\boldsymbol{\Theta}),\boldsymbol{\Theta}) - \nabla \bar{g}^t(\boldsymbol{\Theta}')\nabla_1 \bar{f}^t(\bar{g}^t(\boldsymbol{\Theta}'),\boldsymbol{\Theta}')\|\\
&+ \|\nabla_2 \bar{f}^t(\bar{g}^t(\boldsymbol{\Theta}),\boldsymbol{\Theta}) - \nabla_2 \bar{f}^t(\bar{g}^t(\boldsymbol{\Theta}'),\boldsymbol{\Theta}')\|\\
\leq& \bar{C}_{f_1}\bar{L}_g\|\boldsymbol{\Theta}-\boldsymbol{\Theta}'\| + \bar{C}_g\bar{L}_{f_{11}}\|\bar{g}^t(\boldsymbol{\Theta}) - \bar{g}^t(\boldsymbol{\Theta}')\| + \bar{C}_g\bar{L}_{f_{12}}\|\boldsymbol{\Theta}-\boldsymbol{\Theta}'\|\\
&+ \bar{L}_{f_{21}}\|\bar{g}^t(\boldsymbol{\Theta}) - \bar{g}^t(\boldsymbol{\Theta}')\| + \bar{L}_{f_{22}}\|\boldsymbol{\Theta}-\boldsymbol{\Theta}'\|\\
\leq& \bar{L}\|\boldsymbol{\Theta}-\boldsymbol{\Theta}'\|,
\end{aligned}$$

where in the second inequality we use the boundedness of $\|\nabla \bar{g}^t\|$, $\|\nabla \bar{f}^t\|$ and the Lipschitz continuous gradient of $\bar{g}^t$ and $\bar{f}^t$, see (5.27) and (5.28), and in the third inequality we

use the Lipschitz continuity of $\bar{g}^t$ and $\bar{f}^t$ (implied by the boundedness of $\|\nabla \bar{g}^t\|, \|\nabla \bar{f}^t\|$), and $\bar{L}$ is defined as

$$\bar{L} := \bar{C}_{f_1}\bar{L}_g + \bar{C}_g^2\bar{L}_{f_{11}} + \bar{C}_g\bar{L}_{f_{12}} + \bar{C}_g\bar{L}_{f_{21}} + \bar{L}_{f_{22}}.$$

The proof is complete.

Then, follows the same reasoning, we will have following results when stochastic sampling is used,

Lemma 5.6.2. *Suppose Assumption 5 and 6 hold, $f(\cdot)$ and $g(\cdot)$ are defined as (5.13), and the first input of $f(\mathbf{z};\boldsymbol{\Theta};\xi)$ is bounded away from zero as $\|\mathbf{z}\| \geq C_z$, then the following holds:*

(1) The stochastic gradients of f and g are bounded in expectation, that is, there exist positive constants C_g, C_{f_1}, C_{f_2}, such that the following relations hold

$$\begin{aligned}
\mathbb{E}_t\left[\|\nabla g(\boldsymbol{\Theta};\phi)\|\right] &\leq C_g,\\
\mathbb{E}_t\left[\|\nabla_1 f(\mathbf{z};\boldsymbol{\Theta};\xi)\|\right] &\leq C_{f_1},\\
\mathbb{E}_t\left[\|\nabla_2 f(\mathbf{z};\boldsymbol{\Theta};\xi)\|\right] &\leq C_{f_2},
\end{aligned}$$

where the constants are defined as:

$$C_{f_1} := C_{l_0}e^{C_{u_0}}/C_z^2, \quad C_{f_2} := C_{u_1}C_{l_0}e^{C_{u_0}}/C_z + C_{l_1}e^{C_{u_0}}/C_z, \quad C_g := C_{u_1}e^{C_{u_0}}.$$

(2) Functions ∇f and ∇g are L_f- and L_g-smooth, that is, for any $\boldsymbol{\Theta}, \boldsymbol{\Theta}' \in \mathbb{R}^d$, and $\mathbf{z}, \mathbf{z}'$ satisfying $\|\mathbf{z}\| \leq C_z$ and $\|\mathbf{z}'\| \leq C_z$, we have:

$$\begin{aligned}
\|\nabla_1 f(\mathbf{z},\boldsymbol{\Theta};\xi) - \nabla_1 f(\mathbf{z}',\boldsymbol{\Theta};\xi)\| &\leq L_{f_{11}}\|\mathbf{z}-\mathbf{z}'\|,\\
\|\nabla_1 f(\mathbf{z},\boldsymbol{\Theta};\xi) - \nabla_1 f(\mathbf{z},\boldsymbol{\Theta}';\xi)\| &\leq L_{f_{12}}\|\boldsymbol{\Theta}-\boldsymbol{\Theta}'\|,\\
\|\nabla_2 f(\mathbf{z},\boldsymbol{\Theta};\xi) - \nabla_2 f(\mathbf{z}',\boldsymbol{\Theta};\xi)\| &\leq L_{f_{21}}\|\mathbf{z}-\mathbf{z}'\|,\\
\|\nabla_2 f(\mathbf{z},\boldsymbol{\Theta};\xi) - \nabla_2 f(\mathbf{z},\boldsymbol{\Theta}';\xi)\| &\leq L_{f_{22}}\|\boldsymbol{\Theta}-\boldsymbol{\Theta}'\|,\\
\|\nabla g(\boldsymbol{\Theta};\phi) - \nabla g(\boldsymbol{\Theta}';\phi)\| &\leq L_g\|\boldsymbol{\Theta}-\boldsymbol{\Theta}'\|,
\end{aligned} \tag{5.29}$$

where

$$L_{f_{11}} := 2C_{l_0}e^{C_{u_0}}/C_z^3, \quad L_{f_{12}} = L_{f_{21}} := \left(C_{u_1}C_{l_0}e^{C_{u_0}} + C_{l_1}e^{C_{u_0}}\right)/C_z^2,$$
$$L_{f_{22}} := \left(C_{u_1}^2 C_{\ell_0} + C_{u_2}C_{\ell_0} + 2C_{u_1}C_{\ell_1} + C_{\ell_2}\right)e^{C_{u_0}}/C_z, \quad L_g := C_{u_1}^2 e^{C_{u_0}} + C_{u_2}e^{C_{u_0}}.$$

Proof. The derivation of this result is similar to those presented in Lemma 5.6.1, except the change of $i \in \mathcal{D}$ with $i \in \xi$ or $i \in \phi$ in derivations, and the usage of $z = \mathbf{y}$ instead of $z = \bar{g}^t(\cdot)$.

Chapter 6

Summary and Future Directions

In the last chapter, we provide a summary of the main results discussed in this thesis, and also point out a few promising directions for future research.

6.1 Thesis Summary

This thesis presented a set of contributions at the intersection of optimization theory, machine learning advances, and applications in modern wireless systems. The focus was on building fundamental connections between methodologies from optimization, machine learning, and networking communities, and developing interdisciplinary approaches for the modern large-scale non-convex problem.

In the first part of the thesis, which contains Chapters 2–4, the aim is to develop theoretical guarantees for distributed non-convex optimization that arising in the modern big data era.

Chapters 2 and 3 represent the first work that investigates the performance of optimal first-order non-convex algorithms for distributed information processing and optimization problems. We first set our scope by defining the problem, network, and algorithm classes $(\mathcal{P}, \mathcal{N}, \mathcal{A})$ that are under consideration. We then provide a lower complexity bound that characterizes the worst-case performance for any first-order distributed algorithm in class $\mathcal{A}$, and finally propose and analyze two algorithms that are capable of (nearly) achieving the lower bound in various settings. The various bounds

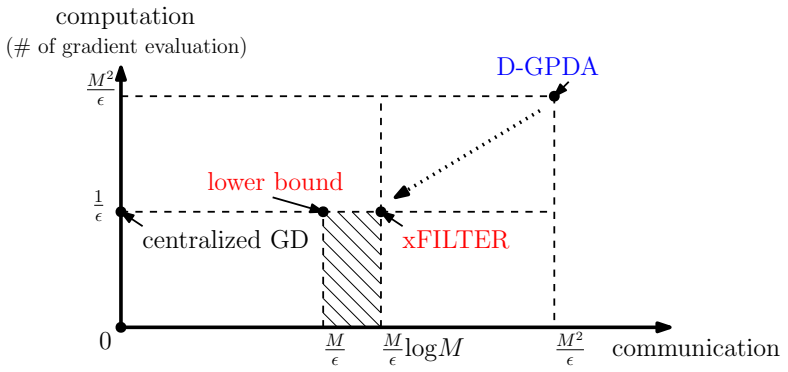
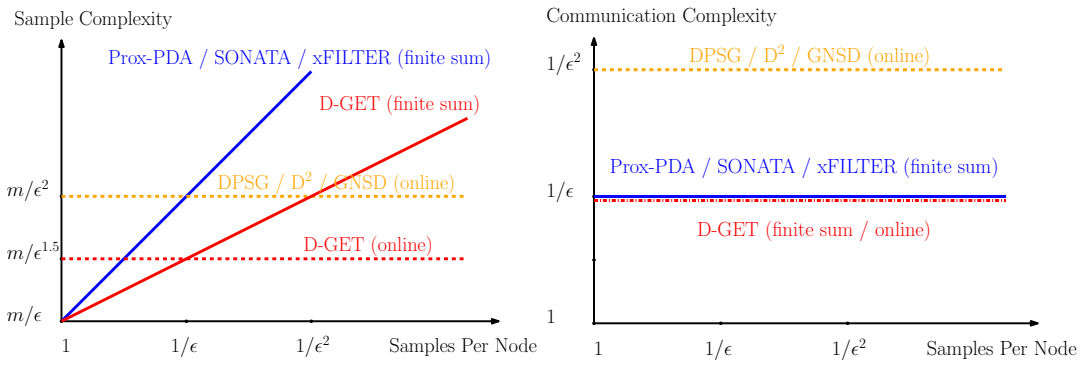


Figure 6.1: Graphical comparison of various bounds analyzed in this work, illustrated over a path graph with M nodes.

discussed in the work is illustrated in Fig. 6.1 through a M-node path graph as an example. To the best of our knowledge, the proposed algorithms in Chapter 3 are the first and the only available distributed non-convex algorithms in class $\mathcal{A}$ that can optimally reduce *both* the size of the gradient and the consensus error for $(\mathcal{P}, \mathcal{N})$, and achieving the (near) optimal rate performance for problem/network classes $(\mathcal{P}, \mathcal{N})$.

In Chapter 4, we proposed a joint gradient estimation and tracking approach (D-GET) for fully decentralized non-convex optimization problems. By utilizing modern variance reduction and gradient tracking techniques, the proposed method improves the sample and/or communication complexities compared with existing methods. In particular, for decentralized finite sum problems, the proposed approach requires only $\mathcal{O}(mn^{1/2}\epsilon^{-1})$ sample complexity and $\mathcal{O}(\epsilon^{-1})$ communication complexity to reach the ϵ stationary solution. For online problem, our approach achieves an $\mathcal{O}(m\epsilon^{-3/2})$ sample and an $\mathcal{O}(\epsilon^{-1})$ communication complexity, which significantly improves upon the best existing bounds of $\mathcal{O}(m\epsilon^{-2})$ and $\mathcal{O}(\epsilon^{-2})$ as derived in [77]. The main results can be summarized as in Fig. 6.2.

The second part of the thesis, which includes Chapters 5, introduced a class of learning-based approaches for modern wireless systems, utilizing the theory and algorithms proposed in part I of the thesis. The key message is: *DNNs have great potential as computationally cheap surrogates of expensive optimization algorithms for quasi-optimal and real-time wireless resource allocation.* In particular, we design a new "learning to continuously optimize" framework for optimizing wireless resources in dynamic environments, where parameters such as CSIs keep changing. By introducing continual



(a) Sample Complexity (b) Communication Complexity

Figure 6.2: Comparison of the sample and communication complexities for a number of decentralized methods. Existing deterministic methods enjoy lower sample complexity at smaller sample sizes, but such complexity scales linearly when the number of samples increases. Stochastic methods generally suffer from high communication complexity. The proposed D-GET bridges the gap between existing deterministic and stochastic methods, and achieves the optimal sample and communication complexities. Note that online methods can also be applied for finite sum problems, thus the actual sample complexity of D-GET is the minimum rate of both cases.

learning (CL) into the modeling process, our framework is able to seamlessly and efficiently adapt to the episodically dynamic environment, without knowing the episode boundary, and most importantly, maintain high performance over all the previously encountered scenarios. The proposed approach is validated through two popular wireless resource allocation problems (one for power control and one for beamforming), and uses both synthetic and ray-tracing based data sets. Simulation results show that our framework is consistently better than naive transfer learning method, and it achieves better performance than classical CL based approaches. Our empirical results make us believe that our approaches can be extended to many other related problems.

6.2 Future Research Directions

Moving forward, I will continue my research on developing machine learning and optimization theories and tools for modern applications. My work represents a preliminary step towards understanding the capability of distributed non-convex learning and deep learning for wireless problems. There are many interesting questions to be addressed in the future, and some of them are listed below:

- Is it possible to merge the inner Chebyshev iteration with the outer dual update to design a *single-loop* algorithm and to extend the proposed algorithms to problems

with nonsmooth regularizers and constraints?

- Is it possible to design second order methods to further speed up the convergence?
- Is it possible to design global information free algorithms that only require local structures to initialize the parameters?
- Is it possible to design theoretical results for the proposed stage-wise bilevel optimization problem (5.8) or even the global bilevel optimization problem (5.4)?
- Is it possible to quantify the generalization performance of the proposed fairness framework?
- How to further reduce the computational complexity of DNNs?

In the end, we live in a highly smart and connected world, and the exponentially increasing number of smart devices has posed a great challenge for modern distributed learning tasks. My long-term research objective is to formally establish links between theoretical limits and design principles of distributed optimization and learning, and ultimately contribute to the vision of a highly smart and connected world.

References

[1] H. Sun and M. Hong, "Distributed non-convex first-order optimization and information processing: Lower complexity bounds and rate optimal algorithms," *IEEE Transactions on Signal processing*, vol. 67, no. 22, pp. 5912–5928, 2019.

[2] H. Sun, S. Lu, and M. Hong, "Improving the sample and communication complexity for decentralized non-convex optimization: Joint gradient estimation and tracking," in *International Conference on Machine Learning*, pp. 9217–9228, PMLR, 2020.

[3] H. Sun and M. Hong, "Distributed non-convex first-order optimization and information processing: Lower complexity bounds and rate optimal algorithms," in *proceedings of the 52nd Asilomar Conference on Signals, Systems, and Computers*, pp. 38–42, IEEE, 2018.

[4] H. Sun, X. Chen, Q. Shi, M. Hong, X. Fu, and N. D. Sidiropoulos, "Learning to optimize: Training deep neural networks for interference management," *IEEE Transactions on Signal Processing*, vol. 66, no. 20, pp. 5438–5453, 2018.

[5] H. Sun, Z. Zhao, X. Fu, and M. Hong, "Limited feedback double directional massive mimo channel estimation: From low-rank modeling to deep learning," in *proceedings of the IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 1–5, IEEE, 2018.

[6] H. Sun, A. O. Kaya, M. Macdonald, H. Viswanathan, and M. Hong, "Deep learning based preamble detection and toa estimation," in *proceedings of the IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, 2019.

[7] H. Sun, W. Pu, M. Zhu, X. Fu, T.-H. Chang, and M. Hong, "Learning to continuously optimize wireless resource in episodically dynamic environment," in *proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4945–4949, IEEE, 2021.

[8] P. Bianchi and J. Jakubowicz, "Convergence of a multi-agent projected stochastic gradient algorithm for non-convex optimization," *IEEE Transactions on Automatic Control*, vol. 58, no. 2, pp. 391–405, 2013.

[9] M. Zhu and S. Martínez, "An approximate dual subgradient algorithm for distributed non-convex constrained optimization," *IEEE Transactions on Automatic Control*, vol. 58, pp. 1534–1539, June 2013.

[10] M. Hong, Z.-Q. Luo, and M. Razaviyayn, "Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems," *SIAM Journal On Optimization*, vol. 26, no. 1, pp. 337–364, 2016.

[11] P. Di Lorenzo and G. Scutari, "Next: In-network nonconvex optimization," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 2, pp. 120–136, 2016.

[12] D. Hajinezhad and M. Hong, "Perturbed proximal primal dual algorithm for nonconvex nonsmooth optimization," *Mathematical Programming*, vol. 176, pp. 207–245, July 2019.

[13] M. Hong, D. Hajinezhad, and M.-M. Zhao, "Prox-PDA: The proximal primal-dual algorithm for fast distributed nonconvex optimization and learning over networks," in *the Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

[14] T.-H. C. H.-T. Wai and A. Scaglione, "A consensus-based decentralized algorithm for non-convex optimization with application to dictionary learning," in *the Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, 2015.

[15] D. Hajinezhad, M. Hong, and A. Garcia, "Zone: Zeroth order nonconvex multi-agent optimization over networks," *IEEE Transactions on Automatic Control*, 2019.

[16] A. Daneshmand, G. Scutari, and F. Facchinei, "Distributed dictionary learning," in *Proceedings of the Asilomar Conference on Signals, Systems, and Computers*, Nov. 6–9, 2016.

[17] A. Daneshmand, Y. Sun, G. Scutari, and F. Facchinei, "Distributed dictionary learning over networks," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, March 5-9 2017.

[18] J. Zeng and W. Yin, "On nonconvex decentralized gradient descent," *IEEE Transactions on Signal Processing*, vol. 66, pp. 2834–2848, June 2018.

[19] Z. Jiang, A. Balu, C. Hegde, and S. Sarkar, "Collaborative deep learning in fixed topology networks," in *Advances in Neural Information Processing Systems*, 2017.

[20] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," in *Advances in Neural Information Processing Systems*, 2017.

[21] S. Vlaski and A. H. Sayed, "Distributed learning in non-convex environments–part i: Agreement at a linear rate," *arXiv preprint arXiv:1907.01848*, 2019.

[22] S. Vlaski and A. H. Sayed, "Distributed learning in non-convex environments–part ii: Polynomial escape from saddle-points," *arXiv preprint arXiv:1907.01849*, 2019.

[23] B. Swenson, S. Kar, H. V. Poor, and J. Moura, "Annealing for distributed global optimization," *arXiv preprint arXiv:1903.07258*, 2019.

[24] Y. Nesterov, "Smooth minimization of nonsmooth functions," *Mathematical Programming*, vol. 103, pp. 127–152, 2005.

[25] Y. Nesterov, "A method of solving a convex programming problem with convergence rate $o(1/k^2)$," *Soviet Mathematics Doklady*, vol. 27, pp. 372–376, 1983.

[26] A. Nemirovsky and D. Yudin, "Problem complexity and method efficiency in optimization," in *Interscience Series in Discrete Mathematics*, Wiley, 1983.

[27] Y. Nesterov, *Introductory lectures on convex optimization: A basic course*. Springer, 2004.

[28] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imgaging Science*, vol. 2, no. 1, pp. 183 – 202, 2009.

[29] Y. Ouyang, Y. Chen, G. Lan, and J. E. Pasiliao, "An accelerated linearized alternating direction method of multipliers," *SIAM Journal on Imaging Sciences*, vol. 8, no. 1, pp. 644–681, 2015.

[30] P. Tseng, "On accelerated proximal gradient methods for convex-concave optimization," 2008. preprint.

[31] D. Jakovetic, J. Xavier, and J. M. F. Moura, "Fast distributed gradient methods," *IEEE Transactions on Automatic Control*, vol. 59, pp. 1131–1146, May 2014.

[32] K. Scaman, F. Bach, S. Bubeck, Y. Lee, and L. Massoulié, "Optimal algorithms for smooth and strongly convex distributed optimization in networks," *arXiv preprint arXiv:1702.08704*, 2017.

[33] C. Uribe, S. Lee, A. Gasnikov, and A. Nedić, "Optimal algorithms for distributed optimization," *arXiv preprint arXiv:1712.00232*, 2017.

[34] K. Scaman, F. Bach, S. Bubeck, L. Massoulié, and Y. T. Lee, "Optimal algorithms for non-smooth distributed optimization in networks," in *Advances in Neural Information Processing Systems*, pp. 2740–2749, 2018.

[35] C. Cartis, N. Gould, and P. Toint, "On the complexity of steepest descent, newton's and regularized newton's methods for nonconvex unconstrained optimization problems," *SIAM journal on optimization*, vol. 20, no. 6, pp. 2833–2852, 2010.

[36] Y. Carmon, J. C. Duchi, O. Hinder, and A. Sidford, "Lower bounds for finding stationary points i," *Mathematical Programming*, Jun 2019.

[37] Y. Tian, Y. Sun, B. Du, and G. Scutari, "Asy-sonata: Achieving geometric convergence for distributed asynchronous optimization," *arXiv preprint arXiv:1803.10359*, 2018.

[38] A. Daneshmand, Y. Sun, and G. Scutari, "Convergence rate of distributed convex and nonconvex optimization methods with gradient tracking," 2018. Purdue University, Tech. Rep.

[39] X. Fu, K. Huang, N. Sidiropolous, A. M.-S. So, and M. Hong, "Scalable and optimal generalized canonical correlation analysis via alternating optimization," 2016. submitted to NIPS 2016.

[40] F. R. K. Chung, *Spectral Graph Theory*. The American Mathematical Society, 1997.

[41] S. Butler, *Algebraic aspects of the normalized Laplacian*, pp. 295–315. Cham: Springer International Publishing, 2016.

[42] 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, pp. 592–606, March 2012.

[43] P. A. Forero, A. Cano, and G. B. Giannakis, "Distributed clustering using wireless sensor networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, pp. 707–724, Aug 2011.

[44] A. Nedić and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[45] A. Nedic and A. Olshevsky, "Distributed optimization over time-varying directed graphs," *IEEE Transactions on Automatic Control*, vol. 60, no. 3, pp. 601–615, 2015.

[46] 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, 2014.

[47] D. Jakovetić, J. M. Moura, and J. Xavier, "Linear convergence rate of a class of distributed augmented lagrangian algorithms," *IEEE Transactions on Automatic Control*, vol. 60, no. 4, pp. 922–936, 2015.

[48] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[49] I. Schizas, G. Mateos, and G. Giannakis, "Distributed LMS for consensus-based in-network adaptive processing,," *IEEE Transactions on Signal Processing*, vol. 57, no. 6, pp. 2365 – 2382, 2009.

[50] M. Hong, J. D. Lee, and M. Razaviyayn, "Gradient primal-dual algorithm converges to second-order stationary solutions for nonconvex distributed optimization," in *the Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018.

[51] A. Daneshmand, G. Scutari, and V. Kungurtsev, "Second-order guarantees of gradient algorithms over networks," in *Proceedings of the 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2018.

[52] B. Swenson, R. Murray, H. V. Poor, and S. Kar, "Distributed gradient descent: Nonconvergence to saddle points and the stable-manifold theorem," *arXiv preprint arXiv:1908.02747*, 2019.

[53] C. Duenner, A. Lucchi, M. Gargiani, A. Bian, T. Hofmann, and M. Jaggi, "A distributed second-order algorithm you can trust," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

[54] C.-H. Fang, S. B. Kylasa, F. Roosta-Khorasani, M. W. Mahoney, and A. Grama, "Distributed second-order convex optimization," *arXiv preprint arXiv:1807.07132*, 2018.

[55] H. Uzawa, "Iterative methods in concave programming," in *Studies in Linear and Nonlinear Programming*, p. 154–165, Stanford University Press, 1958.

[56] A. Nedić and A. Ozdaglar, "Subgradient methods for saddle-point problems," *Journal of optimization theory and applications*, vol. 142, no. 1, pp. 205–228, 2009.

[57] R. T. Rockafellar, "Augmented lagrangians and applications of the proximal point algorithm in convex programming," *Mathematics of operations research*, vol. 1, no. 2, pp. 97–116, 1976.

[58] S. J. Wright, "Implementing proximal point methods for linear programming," *Journal of optimization Theory and Applications*, vol. 65, no. 3, pp. 531–554, 1990.

[59] D. Tian, H. Mansour, A. Knyazev, and A. Vetro, "Chebyshev and conjugate gradient filters for graph image denoising," in *Proceedings of the IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, 2014.

[60] A. Gadde, S. K. Narang, and A. Ortega, "Bilateral filter: Graph spectral interpretation and extensions," in *Proceedings of the IEEE International Conference on Image Processing*.

[61] V. S. Ryaben'kii and S. V. Tsynkov, *A Theoretical Introduction to Numerical Analysis*. CRC Press, 2007.

[62] A. A. Samarskij and E. S. Nikolaev, *Numerical Methods for Grid Equations Volume II Iterative Methods*. Springer, 1989.

[63] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2508–2530, 2006.

[64] A. Antoniadis, I. Gijbels, and M. Nikolova, "Penalized likelihood regression for generalized linear models with non-quadratic penalties," *Annals of the Institute of Statistical Mathematics*, vol. 63, no. 3, pp. 585–615, 2011.

[65] T. Tatarenko and B. Touri, "Non-convex distributed optimization," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3744–3757, 2017.

[66] A. Nedić, A. Olshevsky, A. Ozdaglar, and J. N. Tsitsiklis, "On distributed averaging algorithms and quantization effects," *IEEE Transactions on Automatic Control*, vol. 54, no. 11, pp. 2506–2517, 2009.

[67] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[68] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 5330–5340, 2017.

[69] J. Chen and A. H. Sayed, "Diffusion adaptation strategies for distributed optimization and learning over networks," *IEEE Transactions on Signal Processing*, vol. 60, no. 8, pp. 4289–4305, 2012.

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

[71] K. Ali and W. Van Stam, "TiVo: making show recommendations using a distributed collaborative filtering architecture," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pp. 394–401, 2004.

[72] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, 2016.

[73] J. Chen, Z. J. Towfic, and A. H. Sayed, "Dictionary learning over distributed models," *IEEE Transactions on Signal Processing*, vol. 63, no. 4, pp. 1001–1016, 2014.

[74] M. Hong, D. Hajinezhad, and M.-M. Zhao, "Prox-PDA: The proximal primal-dual algorithm for fast distributed nonconvex optimization and learning over networks," in *Proceedings of International Conference on Machine Learning (ICML)*, pp. 1529–1538, 2017.

[75] P. Di Lorenzo and G. Scutari, "Next: In-network nonconvex optimization," 2016.

[76] Y. Sun, A. Daneshmand, and G. Scutari, "Convergence rate of distributed optimization algorithms based on gradient tracking," *arXiv preprint arXiv:1905.02637*, 2019.

[77] H. Tang, X. Lian, M. Yan, C. Zhang, and J. Liu, "D^2: Decentralized training over decentralized data," in *Proceedings of International Conference on Machine Learning (ICML)*, pp. 4855–4863, 2018.

[78] M. Assran, N. Loizou, N. Ballas, and M. Rabbat, "Stochastic gradient push for distributed deep learning," in *Proceedings of International Conference on Machine Learning (ICML)*, pp. 344–353, 2019.

[79] S. Lu, X. Zhang, H. Sun, and M. Hong, "GNSD: a gradient-tracking based nonconvex stochastic algorithm for decentralized optimization," in *Proceedings of IEEE Data Science Workshop (DSW)*, pp. 315–321, June 2019.

[80] C. Fang, C. J. Li, Z. Lin, and T. Zhang, "SPIDER: Near-optimal non-convex optimization via stochastic path-integrated differential estimator," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 689–699, 2018.

[81] L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takáč, "SARAH: A novel method for machine learning problems using stochastic recursive gradient," in *Proceedings of International Conference on Machine Learning (ICML)*, pp. 2613–2621, 2017.

[82] S. Pu and A. Nedić, "Distributed stochastic gradient tracking methods," *arXiv preprint arXiv:1805.11454*, 2018.

[83] S. Lu and C. W. Wu, "Decentralized stochastic non-convex optimization over weakly connected time-varying digraphs," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 5770–5774, 2020.

[84] Y. Arjevani, Y. Carmon, J. C. Duchi, D. J. Foster, N. Srebro, and B. Woodworth, "Lower bounds for non-convex stochastic optimization," *arXiv preprint arXiv:1912.02365*, 2019.

[85] J. Zeng and W. Yin, "On nonconvex decentralized gradient descent," *IEEE Transactions on Signal Processing*, vol. 66, no. 11, pp. 2834–2848, 2018.

[86] D. P. Bertsekas, *Parallel and distributed computation: numerical methods*, vol. 23. Prentice hall Englewood Cliffs, NJ, 1989.

[87] A. Nedić, A. Olshevsky, and M. G. Rabbat, "Network topology and communication-computation tradeoffs in decentralized optimization," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 953–976, 2018.

[88] M. Hong, Z.-Q. Luo, and M. Razaviyayn, "Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems," *SIAM Journal on Optimization*, vol. 26, no. 1, pp. 337–364, 2016.

[89] A. Daneshmand, G. Scutari, and F. Facchinei, "Distributed dictionary learning," in *Proceedings of the 50th Asilomar Conference on Signals, Systems and Computers*, pp. 1001–1005, 2016.

[90] P. Bianchi, G. Fort, and W. Hachem, "Performance of a distributed stochastic approximation algorithm," *IEEE Transactions on Information Theory*, vol. 59, no. 11, pp. 7405–7418, 2013.

[91] Z. Jiang, A. Balu, C. Hegde, and S. Sarkar, "Collaborative deep learning in fixed topology networks," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 5904–5914, 2017.

[92] Y. Nesterov, "Introductory lectures on convex programming volume i: Basic course," *Lecture notes*, vol. 3, no. 4, p. 5, 1998.

[93] S. Ghadimi and G. Lan, "Stochastic first-and zeroth-order methods for nonconvex stochastic programming," *SIAM Journal on Optimization*, vol. 23, no. 4, pp. 2341–2368, 2013.

[94] A. Defazio, F. Bach, and S. Lacoste-Julien, "SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 1646–1654, 2014.

[95] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 315–323, 2013.

[96] S. J. Reddi, A. Hefny, S. Sra, B. Poczos, and A. Smola, "Stochastic variance reduction for nonconvex optimization," in *Proceedings of International Conference on Machine Learning (ICML)*, pp. 314–323, 2016.

[97] Z. Allen-Zhu and E. Hazan, "Variance reduction for faster non-convex optimization," in *Proceedings of International Conference on Machine Learning (ICML)*, pp. 699–707, 2016.

[98] L. Lei, C. Ju, J. Chen, and M. I. Jordan, "Non-convex finite-sum optimization via SCSG methods," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 2348–2358, 2017.

[99] L. M. Nguyen, M. van Dijk, D. T. Phan, P. H. Nguyen, T.-W. Weng, and J. R. Kalagnanam, "Optimal finite-sum smooth non-convex optimization with sarah," *arXiv preprint arXiv:1901.07648*, 2019.

[100] Z. Wang, K. Ji, Y. Zhou, Y. Liang, and V. Tarokh, "Spiderboost and momentum: Faster variance reduction algorithms," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 2403–2413, 2019.

[101] D. Zhou, P. Xu, and Q. Gu, "Stochastic nested variance reduced gradient descent for nonconvex optimization," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 3921–3932, 2018.

[102] 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.

[103] 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.

[104] Z. Shen, A. Mokhtari, T. Zhou, P. Zhao, and H. Qian, "Towards more efficient stochastic decentralized learning: Faster convergence and sparse communication," *arXiv preprint arXiv:1805.09969*, 2018.

[105] K. Yuan, B. Ying, J. Liu, and A. H. Sayed, "Variance-reduced stochastic learning by networked agents under random reshuffling," *IEEE Transactions on Signal Processing*, vol. 67, no. 2, pp. 351–366, 2018.

[106] H. Hendrikx, F. Bach, and L. Massoulie, "An accelerated decentralized stochastic proximal algorithm for finite sums," *arXiv preprint arXiv:1905.11394*, 2019.

[107] Z. Wang and H. Li, "Edge-based stochastic gradient algorithm for distributed optimization," *IEEE Transactions on Network Science and Engineering*, 2019.

[108] R. Xin, U. A. Khan, and S. Kar, "Variance-reduced decentralized stochastic optimization with gradient tracking," *arXiv preprint arXiv:1909.11774*, 2019.

[109] B. Li, S. Cen, Y. Chen, and Y. Chi, "Communication-efficient distributed optimization in networks with gradient tracking," *arXiv preprint arXiv:1909.05844*, 2019.

[110] S. Cen, H. Zhang, Y. Chi, W. Chen, and T.-Y. Liu, "Convergence of distributed stochastic variance reduced methods without sampling extra data," *arXiv preprint arXiv:1905.12648*, 2019.

[111] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems and Control Letters*, vol. 53, no. 1, pp. 65–78, 2004.

[112] S. Boyd, P. Diaconis, and L. Xiao, "Fastest mixing markov chain on a graph," *SIAM Review*, vol. 46, no. 4, pp. 667–689, 2004.

[113] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Computational Intelligence and Neuroscience*, 2018.

[114] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *IEEE Computational Intelligence Magazine*, vol. 13, no. 3, pp. 55–75, 2018.

[115] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in *Proceedings of the 21th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pp. 1235–1244, 2015.

[116] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, vol. 1. MIT press Cambridge, 2016.

[117] H. Ye, G. Y. Li, and B.-H. Juang, "Power of deep learning for channel estimation and signal detection in OFDM systems," *IEEE Wireless Communications Letters*, vol. 7, no. 1, pp. 114–117, 2017.

[118] E. Nachmani, E. Marciano, L. Lugosch, W. J. Gross, D. Burshtein, and Y. Be'ery, "Deep learning methods for improved decoding of linear codes," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 119–131, 2018.

[119] C.-K. Wen, W.-T. Shih, and S. Jin, "Deep learning for massive MIMO CSI feedback," *IEEE Wireless Communications Letters*, vol. 7, no. 5, pp. 748–751, 2018.

[120] S. Dörner, S. Cammerer, J. Hoydis, and S. Ten Brink, "Deep learning based communication over the air," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 132–143, 2017.

[121] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 563–575, 2017.

[122] W. Lee, M. Kim, and D.-H. Cho, "Deep power control: Transmit power control scheme based on convolutional neural network," *IEEE Communications Letters*, vol. 22, no. 6, pp. 1276–1279, 2018.

[123] F. Liang, C. Shen, W. Yu, and F. Wu, "Towards optimal power control via ensembling deep neural networks," *IEEE Transactions on Communications*, vol. 68, no. 3, pp. 1760–1776, 2019.

[124] M. Eisen and A. R. Ribeiro, "Optimal wireless resource allocation with random edge graph neural networks," *IEEE Transactions on Signal Processing*, 2020.

[125] Y. Shen, Y. Shi, J. Zhang, and K. B. Letaief, "LORM: Learning to optimize for resource management in wireless networks with few training samples," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 665–679, 2019.

[126] H. Huang, Y. Peng, J. Yang, W. Xia, and G. Gui, "Fast beamforming design via deep learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 1, pp. 1065–1069, 2019.

[127] W. Cui, K. Shen, and W. Yu, "Spatial deep learning for wireless scheduling," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1248–1261, 2019.

[128] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, vol. 113, pp. 54–71, 2019.

[129] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of Learning and Motivation*, vol. 24, pp. 109–165, Elsevier, 1989.

[130] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[131] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017.

[132] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, "Lifelong learning with dynamically expandable networks," in *International Conference on Learning Representations*, 2018.

[133] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," *Proceedings of Machine Learning Research*, vol. 70, p. 3987, 2017.

[134] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *arXiv:1606.04671*, 2016.

[135] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," in *Advances in Neural Information Processing Systems*, pp. 6467–6476, 2017.

[136] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," in *Advances in Neural Information Processing Systems*, pp. 2990–2999, 2017.

[137] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.

[138] D. Isele and A. Cosgun, "Selective experience replay for lifelong learning," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pp. 3302–3309, 2018.

[139] R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio, "Gradient based sample selection for online continual learning," in *Advances in Neural Information Processing Systems*, pp. 11816–11825, 2019.

[140] D. Rolnick, A. Ahuja, J. Schwarz, T. Lillicrap, and G. Wayne, "Experience replay for continual learning," in *Advances in Neural Information Processing Systems*, pp. 350–360, 2019.

[141] Q. Shi, M. Razaviyayn, Z.-Q. Luo, and C. He, "An iteratively weighted MMSE approach to distributed sum-utility maximization for a MIMO interfering broadcast channel," *IEEE Transactions on Signal Processing*, vol. 59, no. 9, pp. 4331–4340, 2011.

[142] Y. Shen, Y. Shi, J. Zhang, and K. B. Letaief, "A graph neural network approach for scalable wireless power control," in *proceedings of the 2019 IEEE Globecom Workshops (GC Wkshps)*.

[143] Y. S. Nasir and D. Guo, "Multi-agent deep reinforcement learning for dynamic power allocation in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2239–2250, 2019.

[144] A. Balatsoukas-Stimming and C. Studer, "Deep unfolding for communications systems: A survey and some new directions," in *proceedings of the IEEE International Workshop on Signal Processing Systems (SiPS)*, pp. 266–271, 2019.

[145] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pp. 399–406, 2010.

[146] J. R. Hershey, J. L. Roux, and F. Weninger, "Deep unfolding: Model-based inspiration of novel deep architectures," *arXiv preprint arXiv:1409.2574*, 2014.

[147] P. Sprechmann, R. Litman, T. B. Yakar, A. M. Bronstein, and G. Sapiro, "Supervised sparse analysis and synthesis operators," in *Advances in Neural Information Processing Systems*, pp. 908–916, 2013.

[148] N. Samuel, T. Diskin, and A. Wiesel, "Deep MIMO detection," *arXiv preprint arXiv:1706.01151*, 2017.

[149] N. Samuel, T. Diskin, and A. Wiesel, "Learning to detect," *IEEE Transactions on Signal Processing*, vol. 67, no. 10, pp. 2554–2564, 2019.

[150] H. He, C.-K. Wen, S. Jin, and G. Y. Li, "A model-driven deep learning network for mimo detection," in *proceedings of the IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 584–588, 2018.

[151] S. Cammerer, T. Gruber, J. Hoydis, and S. Brink, "Scaling deep learning-based decoding of polar codes via partitioning," *arXiv preprint arXiv:1702.06901*, 2017.

[152] M. Eisen, C. Zhang, L. F. Chamon, D. D. Lee, and A. Ribeiro, "Learning optimal resource allocations in wireless systems," *IEEE Transactions on Signal Processing*, vol. 67, no. 10, pp. 2775–2790, 2019.

[153] M. Borgerding, P. Schniter, and S. Rangan, "Amp-inspired deep networks for sparse linear inverse problems," *IEEE Transactions on Signal Processing*, vol. 65, no. 16, pp. 4293–4308, 2017.

[154] Q. Hu, Y. Cai, Q. Shi, K. Xu, G. Yu, and Z. Ding, "Iterative algorithm induced deep-unfolding neural networks: Precoding design for multiuser mimo systems," *IEEE Transactions on Wireless Communications*, 2020.

[155] M. B. Ring, *Continual learning in reinforcement environments*. PhD thesis, University of Texas at Austin Austin, Texas 78712, 1994.

[156] A. Robins, "Catastrophic forgetting, rehearsal and pseudorehearsal," *Connection Science*, vol. 7, no. 2, pp. 123–146, 1995.

[157] S. Shalev-Shwartz *et al.*, "Online learning and online convex optimization," 2011.

[158] R. Aljundi, E. Belilovsky, T. Tuytelaars, L. Charlin, M. Caccia, M. Lin, and L. Page-Caccia, "Online continual learning with maximal interfered retrieval," in *Advances in Neural Information Processing Systems*, pp. 11849–11860, 2019.

[159] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[160] Y. Yuan, G. Zheng, K.-K. Wong, B. Ottersten, and Z.-Q. Luo, "Transfer learning and meta learning based fast downlink beamforming adaptation," *arXiv preprint arXiv:2011.00903*, 2020.

[161] A. Zappone, M. Di Renzo, M. Debbah, T. T. Lam, and X. Qian, "Model-aided wireless artificial intelligence: Embedding expert knowledge in deep neural networks for wireless system optimization," *IEEE Vehicular Technology Magazine*, vol. 14, no. 3, pp. 60–69, 2019.

[162] Z.-Q. Luo and S. Zhang, "Dynamic spectrum management: Complexity and duality," *IEEE Journal of Selected Topics in Signal Processing*, vol. 2, no. 1, pp. 57–73, 2008.

[163] T. L. Hayes, N. D. Cahill, and C. Kanan, "Memory efficient experience replay for streaming learning," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 9769–9776, IEEE, 2019.

[164] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," *IEEE/ACM Transactions on Networking,*, vol. 8, no. 5, pp. 556 –567, 2000.

[165] M. Hong and Z.-Q. Luo, "Signal processing and optimal resource allocation for the interference channel," in *Academic Press Library in Signal Processing*, Academic Press, 2013.

[166] B. Song, H. Sun, W. Pu, S. Liu, and M. Hong, "To supervise or not to supervise: How to effectively learn wireless interference management models?," in *proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2021.

[167] M. Bengtsson and B. Ottersten, "Optimal downlink beamforming using semidefinite optimization," in *Proceedings of the 37th Annual Allerton Conference*, 1999.

[168] M. Razaviyayn, M. Hong, and Z.-Q. Luo, "Linear transceiver design for a MIMO interfering broadcast channel achieving max-min fairness," *Signal Processing*, vol. 93, no. 12, pp. 3327–3340, 2013.

[169] T. Lin, C. Jin, and M. Jordan, "On gradient descent ascent for nonconvex-concave minimax problems," in *International Conference on Machine Learning*, pp. 6083–6093, PMLR, 2020.

[170] M. Razaviyayn, T. Huang, S. Lu, M. Nouiehed, M. Sanjabi, and M. Hong, "Nonconvex min-max optimization: Applications, challenges, and recent theoretical advances," *IEEE Signal Processing Magazine*, vol. 37, no. 5, pp. 55–66, 2020.

[171] M. Hong, H.-T. Wai, Z. Wang, and Z. Yang, "A two-timescale framework for bilevel optimization: Complexity analysis and application to actor-critic," *arXiv preprint arXiv:2007.05170*, 2020.

[172] C. Herrera, F. Krach, and J. Teichmann, "Estimating full lipschitz constants of deep neural networks," *arXiv preprint arXiv:2004.13135*, 2020.

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

[174] M. Wang, E. X. Fang, and H. Liu, "Stochastic compositional gradient descent: algorithms for minimizing compositions of expected-value functions," *Mathematical Programming*, vol. 161, no. 1-2, pp. 419–449, 2017.

[175] T. Chen, Y. Sun, and W. Yin, "Solving stochastic compositional optimization is nearly as easy as solving stochastic optimization," *arXiv preprint arXiv:2008.10847*, 2020.

[176] A. Alkhateeb, S. Alex, P. Varkey, Y. Li, Q. Qu, and D. Tujkovic, "Deep learning coordinated beamforming for highly-mobile millimeter wave systems," *IEEE Access*, vol. 6, pp. 37328–37348, 2018.

[177] A. Alkhateeb, "DeepMIMO: A generic deep learning dataset for millimeter wave and massive MIMO applications," in *proceedings of the Information Theory and Applications Workshop (ITA)*, (San Diego, CA), 2019.