Distributed Optimization in Learning Algorithms with Parsimony in Communications

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Jineng Ren

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

Advisor: Professor Jarvis D. Haupt

January, 2021

Acknowledgements

First and foremost, my deepest gratitude goes to my advisor Prof. Jarvis D. Haupt. I would like to thank him for sharing his wisdom and providing thoughtful supports and suggestions. His guidance and encouragement have always made me feel warm and confident on the journey of becoming a better researcher. This thesis has benefited greatly from his insightful suggestions and comments. I also want to extend my sincerest appreciation to Prof. Georgios B. Giannakis, Prof. Rui Kuang, and Prof. Andrew Lamperski who are willing to take time from their busy schedules and serve on my doctoral committee. Their commitment to excellence in research and teaching sets good examples for us to follow.

I would like to express my thanks to my great friends and office mates, Yong Li, Yunpeng Hu, Dr. Jianjun Yuan, Kuan Zhang, Meng Ma, Bingcong Li, Khan Nabil, Dr. Leo Li, Dr. Abhinav V. Sambasivan, Dr. Alex Gutierrez, Dr. Mojtaba Elyaderani, Dr. Bo Yang, Prof. Kejun Huang, Prof. Tianyi Chen, Prof. Yanning Shen, Prof. Gang Wang, and many others, for making my study life a wonderful experience. Moreover, I want to thank Prof. Wenchang Sun and Prof. Feng Wei for their mentoring and guidance during my master and undergraduate study. Their ideas, visions, and insights consistently inspire me a lot.

Finally, my family has also been the greatest source of inspiration along my study journey. I want to specially thank my parents and my sisters for their constant love, care, and steady belief in me and thank my fiancee for her true devotion and true support for me to pursue my dream.

Abstract

As we have seen, today machine learning and big data technologies are transforming both our daily life and economies fundamentally. An important factor that fuels the progress of learning algorithms is the abundance of data generated everyday. In many scenarios, including internet of things, intelligent transportation systems, mobile/edge computing, and smart grids, the datasets are often generated and stored locally in different locations. Traditional centralized (concentrated) algorithms, however, are facing challenges in these settings because they usually require much higher computation cost on a single machine, more communications for collecting raw local data, and are more vulnerable to possible failure of the host. Therefore the distributed learning and optimization algorithms, which are essentially exempted from those problems, are becoming promising alternatives that attract growing interest in recent years. Generally speaking, distributed algorithms describe the approaches that solve problems in a collaborative manner over multiple agents (machines, nodes, computation units or cores) based on communications among them. The main theme of this work is the identification of efficient and effective ways to exploit distributed procedures and communication structures in this type of settings and applications.

The first part of this work contains the discussions of a communication-efficient distributed optimization framework for general nonconvex nonsmooth signal processing and machine learning problems under an asynchronous protocol. As we know, an important criterion for preferable distributed algorithms in latency and communication sensitive applications is that they can complete tasks fast with as less communication resources as possible. Thus in this part we present an asynchronous efficient distributed algorithm with reduced waiting time based on the updates utilizing local higher-order information and investigate the theoretical guarantee for the convergence and the simulation performance of this type of algorithms in both strongly convex and nonconvex nonsmooth scenarios.

The second part of this work examines the relationship between communication structures and efficiency in distributed optimization. The strategy of setting proper communication structures or patterns is an inexpensive way to save the communication

cost of distributed algorithms. It is shown here that in the background of multi-agent policy evaluation, certain communication structures can result in significant improvements in the efficiency of distributed algorithms, providing new insight into the setting of communication networks. Specifically, a hierarchical structure that differentiates the roles of each of the agents during the evaluation process is proposed allowing us to freely choose various mixing schemes (and corresponding mixing matrices that are not necessarily symmetric or doubly stochastic), in order to reduce the communication and computation cost, while still maintaining convergence comparable to the homogeneous distributed algorithms. Extensive numerical experiments corroborate that the performance of the proposed approaches indeed improves over other advanced algorithms in terms of total communication efficiency.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

As huge amounts of big data applications emerge in our daily life along with high speed communication techniques, the necessity and feasibility of faster and collaborative computing and learning over networks are becoming more and more prominent. As a result, the study of distributed algorithms, where the nodes (or computation cores) over a network aim to solve inference and optimization problems cooperatively, has attracted significantly more attention in the signal processing and machine learning communities. Finding ways to design and analyze distributed algorithms that can perform effectively and efficiently is becoming a primary focus in various domains including internet of things, cloud and edge computing, as well as federated learning.

For example, in multi-agent reinforcement learning (MARL), agents collaborate to learn the value of a given policy with private local rewards and jointly observed state-action pairs. Since accessing the private data is limited in this setting, the nodes only communicate parameter infromation with their neighbors over a network to achieve a global optimum. Moreover, for smart grids applications, a number of problems such as the optimal power flow management [1] and economic power dispatch [2] can be formulated as distributed optimization problems with constraints on the power system's parameters. Utilizing the intrinsic distributed processing units and network structures is able to help us offload control tasks from the central controller to local controllers and exempt the system from relying heavily on a central controller. What's more, in distributed sensing applications with wireless sensor networks [3], a broad class of estimation problems, which includes source localization, cluster and density estimation, as well as average consensus in distributed particle filters [4–6], can be expressed as

the optimization of a cost function involving data collected by the sensor nodes. The optimization in these problems can be naturally tackled using distributed algorithms. It is also not hard to envision the applications of distributed optimizations in other areas – for example, in the traffic signal control [7], swarm robotics [8], and in the federated learning where the data is generated and stored in edge equipments like mobile phones and vehicles [9].

In practice, the communication cost of distributed algorithms can be very expensive in terms of raw bytes transmitted, latency, or both; moreover, the computation cost of a local processor is usually relatively limited. Therefore an update and information exchange strategy that can converge faster and require less communication and computation cost is always needed for significant improvements in these applications.

The first part of this work examines the effects of computations/updates of the nodes on the convergence and communication efficiency of distributed algorithms in a general nonconvex nonsmooth scenario [10]. As we know, the existing canonical algorithms usually employ update rules that involves first-order gradient information (primal or dual), projection, and proximal operators. This type of algorithms are prevailing in both centralized applications and distributed ones; see, for example, [11–23]. In our own work, we present the first asynchronous communication-efficient algorithm exploiting local higher-order information that is guaranteed to be convergent for nonconvex nonsmooth problems. The proposed algorithm is general in a sense that the convergence is established in both strongly convex and nonconvex nonsmooth settings with no statistical assumption on the data stored in each local machine; moreover, it is also an asynchronous approach that may reduce the waiting time and accelerate the learning processes significantly. This work appears in Chapter 2 in the following. In another effort based on similar form of updates, we considered the utilization of sparsity in the data transmission to reduce the communication cost. Under conditions on the bound of the sparsified vectors' norm, we prove that the estimation error of the proposed algorithm decreases exponentially and matches that of the centralized method. This work is presented and discussed in Section 4.1.

The second part of the dissertation studies the effects of the structure of the communication network, including the connectivity and topology, on the performance and efficiency of distributed algorithms. Though there are abundant articles studying the

characteristics of a variety of distributed algorithms, very limited number of them considered the communication structure of the agents over a given network, and their corresponding effects on the algorithms. Our work here provides significantly new insight into exploiting the communication structures/patterns to achieve better communication efficiency without undermining the performance. Motivated by the problem of estimating the policy values in the multi-agent reinforcement learning, the hierarchical decentralized procedure proposed in this part allows us to freely choose various mixing schemes (and corresponding mixing matrices that differentiates the roles of each of the agents during the interactions), in order to reduce the communication and computation cost, while still maintaining convergence at rates as fast as or even faster than the previous distributed algorithms. This work appears here in Chapter 3. In the background of ADMM based algorithms, by introducing the consensus constraints, [24] proposed the hybrid consensus structure where the roles of the nodes over the network are essentially differentiated as virtual masters (fusion centers) and workers as well. When the objective function is smooth and strongly convex, their procedure with synchronous updates is proven to converge linearly to the minimum of the unconstrained optimization problem. As a follow-on effort on distributed ADMM based algorithms, [25] further expands the application of this type of structures to the asynchronous hybrid algorithm solving broader composition problems with a nonsmooth regularizer. The convergence to KKT points are also established for the proposed algorithm under the condition on the smoothness and convexity of the objective function. This work is included here in Section 4.3. These approaches are promisingly applicable in general settings of distributed algorithms that require information exchange between computation nodes.

In Chapter 4, we briefly present our several other works on sparsity aware communication-efficient algorithms [26], dictionary-based robust PCA via outlier pursuit [27], as well as asynchronous distributed ADMM with multiple masters [25]. Moreover, the combinations of these approaches with the ideas introduced in the previous chapters and some directions for their future improvements and extensions are also discussed.

Chapter 2

Communication-Efficient Asynchronous Distributed Algorithm for Convex and Nonconvex Problems

This chapter proposes and analyzes a communication-efficient distributed optimization framework for general nonconvex nonsmooth signal processing and machine learning problems under an asynchronous protocol. At each iteration, worker machines compute gradients of a known empirical loss function using their own local data, and a master machine solves a related minimization problem to update the current estimate. We prove that for nonconvex nonsmooth problems, the proposed algorithm converges to a stationary point with a sublinear rate over the number of communication rounds, coinciding with the best theoretical rate that can be achieved for this class of problems. Linear convergence to a global minimum is established without any statistical assumptions on the local data for problems characterized by composite loss functions whose smooth part is strongly convex. Extensive numerical experiments verify that the performance of the proposed approach indeed has significant improvement over other state-of-the-art algorithms in terms of total communication efficiency. The results in this chapter come from our article [10], which appeared in the IEEE Transactions on Signal Processing[1].

[1] ©[2020] IEEE. Reprinted, with permission, from [J. Ren and J. Haupt, A Provably Communication-Efficient Asynchronous Distributed Inference Method for Convex and Nonconvex Problems, IEEE Transactions on Signal Processing, May 2020]

2.1 Introduction

Due to rapid developments in information and computing technology, modern applications often involve vast amounts of data, rendering local processing (e.g., in a single machine, or on a single processing core) computationally challenging or even prohibitive. To deal with this problem, distributed and parallel implementations are natural methods that can fully leverage multi-core computing and storage technologies. However, one drawback of distributed algorithms is that the communication cost can be very expensive in terms of raw bytes transmitted, latency, or both, as machines (i.e., computation nodes) need to frequently transmit and receive information among one another. Therefore, algorithms that require less communication are preferred in this case.

In this section we study a general communication-efficient distributed algorithm which can be applied to a broad class of nonconvex nonsmooth inference problems. Assume that we have available some N data samples. We consider a general problem appearing frequently in signal processing and machine learning applications; we aim to solve

$$\underset{\mathbf{x}\in\mathbb{R}^p}{\operatorname{argmin}}\, \mathbf{L}(\mathbf{x}) := \frac{1}{N}\sum_{k=1}^{N} l_k(\mathbf{x}) + h(\mathbf{x}), \tag{2.1}$$

where each $l_k(\mathbf{x})$ is a loss function associated with the k-th data sample, and is assumed smooth but possibly nonconvex with Lipschitz continuous gradient and $h(\mathbf{x})$ is a convex (proper and lower semi-continuous) function that is possibly nonsmooth. We let $\mathbf{x}^*$ denote a solution to the above problem. Problem (2.1) covers many important machine learning and signal processing problems such as localization with wireless acoustic sensor networks (WASNs) [28], support vector machines (SVM) [29], the independent principal component analysis (ICA) reconstruction problem [30], and the sparse principal component analysis (PCA) problem [31].

For our distributed approach, we consider a network of m total machines having a star topology, where one node designated as the "Master" node (node 1, without loss of generality) is located at the center of the star, and the remaining $m-1$ nodes (with indices $2, 3, \ldots, m$) are the "Worker" nodes (see Figure 2.1). Without loss of generality, assume that the number of data samples is evenly divisible by m, i.e., $N = nm$ for

some integer n, and each machine stores n unique data samples. Then (2.1) can be reformulated to the following problem:

$$\underset{\mathbf{x}\in\mathbb{R}^p}{\operatorname{argmin}} \mathbf{L}(\mathbf{x}) := \frac{1}{mn}\sum_{j=1}^{m}\sum_{i=1}^{n} l_{ji}(\mathbf{x}) + h(\mathbf{x}), \tag{2.2}$$

where $l_{ji}(\mathbf{x})$ is the loss function corresponding to the i-th sample of the j-th machine.

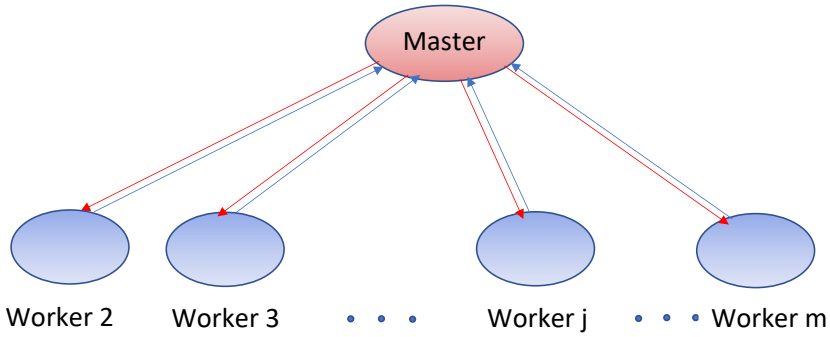


Figure 2.1: An m-node network with a star topology

2.1.1 Main Results

We propose an Efficient Distributed Algorithm for Nonconvex Nonsmooth Inference (EDANNI), and show that, for general problems of the form of (2.2), EDANNI converges to a stationary point if the algorithm parameters are chosen appropriately according to the maximum network delay. Our results differ significantly from existing distributed algorithms in [3, 11, 12] which are all developed for convex problems. Therefore, the analysis and algorithm proposed here are applicable not only to standard convex learning problems but also to important nonconvex problems. Note that [18] and [19] also considered asynchronous distributed algorithms with a master/worker structure for nonconvex problems, but a specific convergence rate is not given. In contrast, our algorithm is proved to converge to a stationary point sublinearly in nonconvex settings. To the best of our knowledge, ours is the first communication-efficient algorithm exploiting local higher-order information that is guaranteed to be convergent for nonconvex nonsmooth problems. Moreover, linear convergence is proved in the strongly convex setting with no statistical assumption on the data stored in each local machine, which

is another improvement on existing works. The synchronization inherent in previous works, including [3, 11, 12, 32, 33], slows down those methods since the master needs to wait for the slowest worker at each iteration; here, we propose an asynchronous approach that may accelerate the inference tasks significantly with proper delay bound (as we will demonstrate in the simulation results).

Contribution. Our contribution is threefold: (i) We propose an asynchronous distributed algorithm for nonsmooth problems and establish its sublinear convergence to a stationary point in nonconvex cases and linear convergence to a global minimum in strongly convex cases, coinciding with the best theoretical rates that can be achieved for this class of problems. (ii) Asynchrony with bounded delay is exploited in the proposed method to reduce the idle time. Moreover, the convergence rate is guaranteed in the case when subproblems are not solved exactly, facilitating its use in practical applications. (iii) In the experiments, we show the preferable communication efficiency of EDANNI in both the convex application (LASSO) and the nonconvex application (Sparse PCA) over other state-of-the-art algorithms.

2.1.2 Related Work

There is a large body of work on distributed optimization for modern data-intensive applications with varied accessibility; see, for example, [3, 11–20]. (Parts of the results presented here also appeared in our conference paper [20] without detailed theoretical analysis.) Early works including [11,13] mainly considered the convergence of parallelizing stochastic gradient descent schemes which stem from the idea of the seminal text by Bertsekas and Tsitsiklis [16]. Niu et al. [14] proposed a lock-free implementation of distributed SGD (Stochastic Gradient Descent) called Hogwild! and provided its rates of convergence for sparse learning problems. That was followed up by many variants like [34,35]. Recently, there are works like [36–39] that further examined the Byzantine-robust methods for distributed learning based on the SGD. For solving machine learning and deep learning problems, works including [17,18,40] studied distributed optimizations under a parameter server framework. Chang et al. [19] studied asynchronous distributed optimizations based on the alternating direction method of multipliers (ADMM). By formulating the optimization problem as a consensus problem, ADMM can be used to solve the consensus problem in a fully parallel fashion over networks with a star topology.

One drawback of such approaches is that they can be computationally intensive, since each worker machine is required to solve a high dimensional nonsmooth subproblem. As we will show, these methods also converge more slowly (in terms of communication rounds) as compared to the proposed approach (see Section 2.4).

Comparison with block coordinate descent methods: The proposed algorithm may look similar to some concentrated (non-distributed) optimization approaches, including block coordinate descent methods like [21–23], which also consider some types of nonconvex and nonsmooth problems. However, these algorithms work in quite different settings from the one considered in this section. Without an additional assumption that the objective function is separable with respect to the variable coordinates, it is hard to apply them to the distributed setting because, to update a coordinate of the variable, these schemes would require that each node has access to the information (that may be partial or compressed [41]) of the whole dataset in order to know the related objective function.

Comparison with fully decentralized methods: There are also some papers on distributed optimizations that consider a different architecture where the roles of all nodes are the same in the sense that each of them needs to update their local estimation of the variable and the gradient information and communicate with their neighbors, see for example, [25, 42–46]. Similar structures are employed in the algorithms for graph signal processing [47–49]. Here we call algorithms with such a structure fully decentralized algorithms. One advantage of fully decentralized algorithms is that they are applicable to networks with more general topologies. Apart from the ADMM based methods discussed above, the approaches proposed by Tian et al. [50], Wu et al. [51] recently are general asynchronous fully decentralized algorithms for nonsmooth or nonconvex problems. The approach in [50] works in the nonconvex setting and can achieve linear convergence in the strongly convex settings, but it requires that the objective functions are smooth. Wu et al. [51] propose a method that can handle nonsmooth problems but its convergence is only guaranteed for convex problems. For nonsmooth problems, the works [42, 52–57] discuss synchronous fully decentralized algorithms where all nodes need to be synchronized at each iteration. However, specific convergence rates are not given in [55–57]; others only establish the sublinear convergence to stationary points in the nonconvex or weakly convex setting while the convergence rate under strongly

convex assumptions is not considered.

In contrast, the proposed algorithm is a distributed algorithm with the master/worker structure that is widely used in works including [14,17–19,32,33,58]. For example, based on such a structure, [18] and [19] study the distributed proximal gradient based approach and the distributed ADMM based algorithm, respectively, for general composite learning problems; [58] proposes an inexact Newton based algorithm for minimizing self-concordant empirical loss functions. Moreover, a seminal lock-free stochastic gradient approach using such a structure is proposed by Recht et al. [14] to solve sparse learning problems. Algorithms with such a structure are different from the fully decentralized algorithms because the workers in those algorithms only compute gradient information and don't update the local copies of the variable. The master collects information from the workers, updates the variable, and broadcasts it to the workers. Though algorithms with the master/worker structure require a specific network with a star topology, they often demonstrate better convergence than fully decentralized ones in practice [59–61], with some exceptions in special cases [62]. Moreover, like distributed ADMM, disributed fully decentralized algorithms can be computationally intensive since each node needs to solve a nonsmooth subproblem to update their local variables. Finally, to verify the preferable performance of the proposed algorithms over other types of fully decentralized algorithms in communication efficiency, our experiments also compare with well-known fully decentralized algorithms, PG-EXTRA [42] and Asyn Primal-Dual [51], that are applicable to nonsmooth problems. To the best of our knowledge, by now there is no fully decentralized method that allows asynchrony and also has specific theoretical guarantees on the convergence rate in both nonconvex and strongly convex nonsmooth settings as the proposed algorithm does.

A growing interest in distributed algorithms also appears in the statistics community [63–65]. Most of those algorithms depend on the partition of data, so their work usually involves statistical assumptions that handle the correlation between the data in local machines. A popular approach in early works is averaging estimators generated locally by different machines [63, 66, 67]. Ma et al. [68] and Jaggi et al. [69] studied distributed optimization based on stochastic dual coordinate descent, however, their communication complexity is not better than that of the previous approaches. Shamir et al. [70] and Zhang and Xiao [58] proposed truly communication-efficient distributed optimization

algorithms which leveraged local second-order information, though these approaches are only guaranteed to work for convex and smooth objectives. In a similar spirit, Wang et al. [32] and Jordan et al. [33] developed communication-efficient algorithms for sparse learning with ℓ_1 regularization. However, each of these works needs an assumption about strong convexity of the loss functions, which may limit their approaches to only a small set of real-world applications. Here we describe an algorithm with similar flavor, but with more general applicability, and establish its convergence rate in both strongly convex and nonconvex nonsmooth settings. Moreover, unlike [32, 33, 58, 70] where the convergence analyses rely on certain statistical assumptions on the data stored in machines, our convergence analysis is deterministic and characterizes the worst-case convergence conditions.

The rest of this chapter is organized as follows. Section 2.2 explains the proposed algorithm in detail and introduces the notion of bounded delay in asynchronous settings. The first part of Section 2.3 provides the analysis and theoretical guarantees for the convergence of the proposed approach in both nonconvex settings and strongly convex settings. The rest of Section 2.3 analyzes the convergence of the proposed approach when the subproblems are not solved exactly. In Section 2.4, we first compare the convergence of the proposed approach with other algorithms in both convex settings (LASSO) and nonconvex settings (Sparse PCA). Next, the speedup and idle time savings of asynchrony are also demonstrated for different settings of the bounded delay. Finally, Section 2.5 provides conclusions on the proposed method, summarizes the main theoretical guarantees and experimental results, and also discusses some future directions and potential improvements.

Notations. For a vector $\mathbf{v} = (v_1, \cdots, v_s)^\top \in \mathbb{R}^s$ and $q > 0$ we write $\|\mathbf{v}\|_q = (\sum_{i=1}^s |v_i|^q)^{1/q}$; for $q \geq 1$ this is a norm. Usually $\|\mathbf{v}\|_2$ is briefly written as $\|\mathbf{v}\|$. The set of natural numbers is denoted by $\mathbb{N}$. For an integer $m \in \mathbb{N}$, we write $[m]$ as shorthand for the set $\{1, \ldots, m\}$.

2.2 Algorithm

In this section, we describe our approach to computing a minimizer $\mathbf{x}^*$ of (2.2). Recall that we have m machines. Let us denote $t \geq 0$ as the iteration number, then $\mathcal{A}_t \subseteq$

$[m] := \{1, 2, \cdots, m\}$ is defined as the subset of $[m]$ that contains the indices of worker machines from which the master receives updated gradient information during iteration t; worker i is said to be "arrived" if $i \in \mathcal{A}_t$ (see also [19]). At iteration t, the master machine solves a subproblem to obtain an updated estimate, and communicates this to the worker machines in the subset $\mathcal{A}_t$. After receiving the updated estimate, the worker machines will compute the corresponding gradients of local empirical loss functions. These gradients are then communicated back to the master machine, and the process continues.

Formally, let

$$\mathbf{L}_j(\mathbf{x}) = \frac{1}{n} \sum_{i \in [n]} l_{ji}(\mathbf{x}), \quad j \in [m]$$

be the empirical loss at each machine. Let t_j be the latest time (in terms of the iteration count) when the worker j is arrived up to and including iteration t. At the t-th iteration, the master (machine 1) solves the following subproblem to update $\mathbf{x}^{t+1}$

$$\begin{aligned} \mathbf{x}^{t+1} = & \operatorname{argmin}_{\mathbf{x}} \mathbf{L}_1(\mathbf{x}) + h(\mathbf{x}) + \tfrac{\rho}{2} \|\mathbf{x} - \mathbf{x}^t\|^2 \\ & + \left\langle \tfrac{1}{m} \textstyle\sum_{j \in [m]} \nabla \mathbf{L}_j(\mathbf{x}^{t_j}) - \nabla \mathbf{L}_1(\mathbf{x}^{t_1}), \mathbf{x} - \mathbf{x}^t \right\rangle. \end{aligned} \tag{2.3}$$

The motivation of solving (2.3) is discussed in detail in Section 2.3. Intuitively, the objective function in (2.3) gives a better approximation to the original objective function than the ones that linearize all the local functions, and thus its solution is expected to be a closer approximation of the minimum of the original problem. After this udpate, the new variable estimation $\mathbf{x}^{t+1}$ is communicated to the worker machines in $\mathcal{A}_t$. Since machine 1 is assumed to be the master machine, iteration t_1 is actually t.

Now one question is: which partial sets of worker machines (with indices in $\mathcal{A}_t$) from which the master receives updated gradient information during iteration t are sufficient to ensure convergence of a distributed approach? Firstly, let $\tau \geq 0$ be a maximum tolerable delay, that is, the maximum number of iterations for which every worker machine can be inactive. The set $\mathcal{A}_t$ should satisfy:

Assumption 2.2.1 (Bounded delay)**.** *For all $i \in [m]$ and iteration $t \geq 0$, it holds that $i \in \mathcal{A}_t \cup \mathcal{A}_{t-1} \cup \cdots \cup \mathcal{A}_{\max\{t-\tau,0\}}$.*

To satisfy Assumption 2.2.1, $\mathcal{A}_t$ should contain at least the indices of the worker machines that have been inactive for longer than τ iterations. That is, the master needs to wait until those workers finish their current computation and have been arrived. Note that by the definition of t_j, it holds that

$$t - \tau \leq t_j \leq t, \quad \forall j \in [m].$$

Assumption 2.2.1 requires that every worker j is arrived at least once within the period $[t-\tau, t]$. In other words, the gradient information $\nabla \mathbf{L}_j(\mathbf{x}^{t_j})$ used by the master must be at most τ iterations old. To guarantee the bounded delay, at every iteration the master needs to wait for the workers who have not been active for τ iterations, if such workers exist. Note that, when $\tau = 0$, one has $j \in \mathcal{A}_t$ for all $j \in [m]$, which reduces to the synchronous case where the master always waits for all the workers at every iteration.

The proposed approach is presented in Algorithm 1, which specifies respectively the steps for the workers and the master. Algorithm 1 has three prominent differences compared with its synchronous alternatives. First, only the workers j in $\mathcal{A}_t$ update the gradient $\nabla \mathbf{L}_j(\mathbf{x}^t)$ and transmit it to the master machine. For the workers j in $\mathcal{A}_t^c$ (the complementary set of $\mathcal{A}_t$ in $[m]$), the master uses their latest gradient information before t, i.e., $\nabla \mathbf{L}_j(\mathbf{x}^{t_j})$. Second, the variables d_j's are defined to count the delays of the workers since their last updates. Here, d_j is set to zero if worker j is arrived at the current iteration; otherwise, d_j is increased by one. Therefore, to ensure Assumption 2.2.1 holds at each iteration, the master should wait if there exists at least one worker whose $d_j > \tau - 1$. Third, after solving subproblem (2.3), the master transmits the up-to-date variable $\mathbf{x}^{t+1}$ only to the arrived workers.

2.3 Theoretical Analysis

Solving subproblem (2.3) is inspired by the approaches of Wang et al. [32], Jordan et al. [33], and Shamir et al. [70], and is designed to take advantage of both global first-order information (gradients) and local higher-order information (e.g., second-order derivatives). Indeed, when $\rho = 0$, $h(\mathbf{x}) = 0$, and $\mathbf{L}_1$ is quadratic with an invertible

Algorithm 1: Efficient Distributed Algorithm for Nonconvex-Nonsmooth Inference (EDANNI)

Input: Loss functions $\{l_{ji}(\cdot,\cdot)\}_{i\in[n],j\in[m]}$, parameter ρ, initial point $\mathbf{x}^0$. Set $d_1 = \cdots = d_m = 0$ and $\mathcal{A}_0 = [m]$;

for $t = 0, 1, \ldots$ **do**

Master:

if $t = 0$, **then** send $\mathbf{x}^0$ to all worker machines.

Else receive gradients $\nabla \mathbf{L}_j(\mathbf{x}^t)$ from workers j whose indices form a set $\mathcal{A}_t$ such that $d_j \leq \tau - 1, \quad \forall j \in \mathcal{A}_t^c$.

Update

$$d_j = \begin{cases} 0 & \forall j \in \mathcal{A}_t \\ d_j + 1 & \forall j \in \mathcal{A}_t^c \end{cases}.$$

Solve subproblem (2.3) with the ρ, the coefficient of the proximal term, that will specified later in Assumption III.2 for nonconvex scenarios or in Theorem III.2 for strongly convex scenarios to obtain $\mathbf{x}^{t+1}$. Broadcast $\mathbf{x}^{t+1}$ to the worker machines whose indices are in $\mathcal{A}_t$.

end

Worker machines:

for $j = 2, 3, \ldots, m$ **do**

if receive $\mathbf{x}^{t+1}$ from the master **then**

Calculate gradient $\nabla \mathbf{L}_j(\mathbf{x}^{t+1})$ and transmit it to the master.

end

end for

end for

Hessian, the first-order optimality condition of (2.3) implies that:

$$\mathbf{x}^{t+1} = \mathbf{x}^t - \nabla^2 \mathbf{L}_1(\mathbf{x}^t)^{-1} \left(\tfrac{1}{m} \textstyle\sum_{j=1}^m \nabla \mathbf{L}_j(\mathbf{x}^{t_j}) \right),$$

which is similar to a Newton updating step. The more general case has a *proximal Newton* flavor; see, e.g., [71] and the references therein. However, our method is different from their methods in the asynchronous indices, the proximal term $\frac{\rho}{2}\|\mathbf{x} - \mathbf{x}^t\|^2$ as well as the first order term in (2.3). Based on this form, deterministic convergence can be established without any statistical assumptions on the local datasets that are required in [32,33,70] in strongly convex settings. Moreover, the convergence to stationary points in nonconvex settings that is not guaranteed in these works can also be established here.

Intuitively, if we have a first-order approximation

$$\mathbf{L}_1(\mathbf{x}) \approx \mathbf{L}_1(\mathbf{x}^t) + \left\langle \nabla \mathbf{L}_1(\mathbf{x}^{t_1}), \mathbf{x} - \mathbf{x}^t \right\rangle, \tag{2.4}$$

then (2.3) reduces to

$$\begin{aligned} \mathbf{x}^{t+1} = \underset{\mathbf{x}}{\operatorname{argmin}} & \left\langle \tfrac{1}{m} \textstyle\sum_{j\in[m]} \nabla \mathbf{L}_j(\mathbf{x}^{t_j}), \mathbf{x} - \mathbf{x}^t \right\rangle \\ & + h(\mathbf{x}) + \frac{\rho}{2}\|\mathbf{x} - \mathbf{x}^t\|^2, \end{aligned} \tag{2.5}$$

which is a first-order proximal gradient updating step.

We consider the convergence of the proposed approach under the asynchronous protocol where the master is free to make updates with gradients from only a partial set of worker machines. We start with introducing important conditions that are used commonly in previous work (e.g., [19, 72]).

Assumption 2.3.1. *The function $\mathbf{L}_j(\mathbf{x})$ is differentiable and has Lipschitz continuous gradient for all $j \in [m]$, i.e., there exists some constant $L > 0$ such that*

$$\|\nabla \mathbf{L}_j(\mathbf{x}) - \nabla \mathbf{L}_j(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|.$$

Assumption 2.3.2. *For all t, the parameter ρ in (2.3) is chosen large enough such that:*

I. $\gamma(\rho) > 3L + 2L\delta\tau$ and $\rho > \frac{2L\tau}{\delta}$, for some constant $\delta > 0$, where $\gamma(\rho)$ represents the convex modulus of the function $h(\mathbf{x}) + \frac{\rho}{2}\|\mathbf{x} - \mathbf{x}^t\|^2$.

II. There exists a constant $\underline{L}$ such that

$$\mathbf{L}(\mathbf{x}) > \underline{L} > -\infty \quad \forall \mathbf{x} \in \mathbb{R}^p.$$

Here δ is a positive constant. The first part of Assumption 2.3.2 is satisfied if there exists at least one such constant satisfying the inequalities in Part I. Moreover, the following concept is needed in the first part of Theorem 2.3.1.

Definition 2.3.1. *We say a function $\mathcal{F}(\mathbf{x})$ is coercive if*

$$\lim_{\|\mathbf{x}\|\to\infty} \mathcal{F}(\mathbf{x}) = +\infty.$$

Define

$$\widetilde{\nabla}_{\mathbf{x}}\mathbf{L}\left(\mathbf{x}^t\right) = \mathbf{x}^t - \mathbf{Prox}_h\Big(\mathbf{x}^t - \tfrac{1}{m}\sum_{j\in[m]} \nabla\mathbf{L}_j(\mathbf{x}^t)\Big), \tag{2.6}$$

where $\mathbf{Prox}_h$ is a proximal operator defined by $\mathbf{Prox}_h[\mathbf{z}] := \operatorname{argmin}_{\mathbf{x}} h(\mathbf{x}) + \frac{1}{2}\|\mathbf{x}-\mathbf{z}\|^2$. Usually $\widetilde{\nabla}_{\mathbf{x}}\mathbf{L}\left(\mathbf{x}^t\right)$ is called the proximal gradient of $\mathbf{L}$; $\mathbf{x}$ is a stationary point when $\widetilde{\nabla}_{\mathbf{x}}\mathbf{L}\left(\mathbf{x}\right) = 0$.

Based on the assumptions, now we can present the first main theorem.

Theorem 2.3.1. *Suppose Assumption 2.2.1, 2.3.1, and 2.3.2 are satisfied. Then we have the following claims for the sequence generated by Algorithm 1 (EDANNI).*

- **(Boundedness of Sequence).** *The gap between $\mathbf{x}^t$ and $\mathbf{x}^{t+1}$ converges to 0, i.e.,*

  $$\lim_{t\to\infty} \mathbf{x}^{t+1} - \mathbf{x}^t = 0.$$

  If $\mathbf{L}(\mathbf{x})$ is coercive, then the sequence $\{\mathbf{x}^t\}$ generated by Algorithm 1 is bounded.

- **(Convergence to Stationary Points).** *Every limit point of the iterates $\{\mathbf{x}^t\}$ generated by Algorithm 1 is a stationary point of problem (2.2). Furthermore, $\left\|\widetilde{\nabla}_{\mathbf{x}}\mathbf{L}(\mathbf{x}^t)\right\| \to 0$, as $t \to \infty$.*

- **(Sublinear Convergence Rate).** *Given $\epsilon > 0$, let us define T to be the first time for the optimality gap to reach below ϵ, i.e.,*

  $$T := \operatorname*{argmin}_t \left\{ \left\|\widetilde{\nabla}_{\mathbf{x}}\mathbf{L}(\mathbf{x}^t)\right\|^2 < \epsilon \right\}.$$

  Then there exists a constant $\nu > 0$ such that

  $$T \le \frac{\nu}{\epsilon} + 1, \tag{2.7}$$

where ν equals to a positive constant times

$$\left(2(2+\rho)^2+8L^2\tau\right)/\min\left\{\frac{\gamma(\rho)}{2}-\frac{3L}{2}-L\delta\tau,\frac{\rho}{2}-\frac{L\tau}{\delta}\right\}$$

for some $\delta>0$. Therefore, the optimality measure $\left\|\widetilde{\nabla}_{\mathbf{x}}\mathbf{L}(\mathbf{x}^t)\right\|$ converges to 0 in a sublinear manner.

Remark 2.3.1. *The theorem suggests that the iterates $\{\mathbf{x}^t\}$ may or may not be bounded without the coerciveness property of $\mathbf{L}(\mathbf{x})$. However, it guarantees that the optimality measure $\left\|\widetilde{\nabla}_{\mathbf{x}}\mathbf{L}(\mathbf{x}^t)\right\|$ converges to 0 sublinearly. We remark that [18] also analyzed the convergence of a proximal gradient method based communication-efficient algorithm for nonconvex problems, but they did not give a specific convergence rate. Note that such sublinear complexity bound (2.7) is tight when applying first-order methods for nonconvex unconstrained problems (see [73, 74]).*

Remark 2.3.2. *Note that when $h(\mathbf{x})=\|\mathbf{x}\|_1$ (ℓ_1 norm of $\mathbf{x}$), the convex modulus of $h(\mathbf{x})$ is 0 (i.e., $h(\mathbf{x})$ is weakly convex) since $h(\mathbf{x})$ is a convex and piece-wise linear function. It can also be proved that the convex modulus of $h(\mathbf{x})=\|\mathbf{x}\|_s$ is 0, for $s>1$; when $h(\mathbf{x})=\|\mathbf{x}\|_s^2$ ($s\in(1,2]$), its convex modulus is positive, i.e., $h(\mathbf{x})$ is strongly convex (see [75]). In this section, $h(\mathbf{x})$ is assumed to be convex (possibly nonsmooth), thus the convex modulus of $h(\mathbf{x})$ is at least 0. Therefore the convex modulus $\gamma(\rho)$ of function $\frac{\rho}{2}\|\mathbf{x}-\mathbf{x}^t\|^2+h(\mathbf{x})$ is lower bounded by ρ. Moreover, observe that as long as $[\frac{2L\tau}{\rho},\frac{\gamma(\rho)-3L}{2L\tau})$ is not empty, there always exists a constant δ such that Assumption 2.3.2(I) holds. Thus to understand the constant δ intuitively, a trivial analogy is that: "$\max(a,b)<v$" is equivalent to "there exists an intermediate constant δ such that $a<\delta$ and $b<\delta<v$". Similar to the one in the analogy, here the δ also plays a role of an intermediate constant. This form of conditions can be found in previous papers like [74] (where the constants d and c in Theorem 3 play a similar role as δ does here).*

Now let us first define

$$\mathbf{F}(\mathbf{x},\mathbf{x}^t):=\tfrac{1}{m}\textstyle\sum_{j\in[m]}\mathbf{L}_j(\mathbf{x})+\frac{\rho}{2}\left\|\mathbf{x}-\mathbf{x}^t\right\|^2+h(\mathbf{x}).$$

This function serves as the Lyapunov function that will be used to prove the convergence

of the proposed algorithm. The gap between $\mathbf{x}^t$ and $\mathbf{x}^{t+1}$ is denoted by $\Delta^{(t)} := \mathbf{x}^{t+1} - \mathbf{x}^t$, for $t \in \mathbb{N}$. The proof of Theorem 2.3.1 relies on Lemmata 2.3.1, 2.3.2, and 2.3.3 below.

First, Lemma 2.3.1 bounds the decent of the middle two terms of subproblem (2.3).

Lemma 2.3.1. *Suppose Assumption 2.3.1 and Assumption 2.3.2 (I) are satisfied. Then the following is true for iterates $\{\mathbf{x}^t\}$ generated by Algorithm 1 (EDANNI)*

$$\begin{aligned}
&\frac{\rho}{2}\|\mathbf{x}^{t+1} - \mathbf{x}^t\|^2 + h(\mathbf{x}^{t+1}) - \frac{\rho}{2}\|\mathbf{x}^t - \mathbf{x}^{t-1}\|^2 - h(\mathbf{x}^t) \\
&\leq -\Big\langle \nabla \mathbf{L}_1(\mathbf{x}^{t+1}) + \frac{1}{m}\sum_{j=1}^{m} \nabla \mathbf{L}_j(\mathbf{x}^{t_j}) - \nabla \mathbf{L}_1(\mathbf{x}^{t_1}), \Delta^{(t)} \Big\rangle \\
&- \frac{\gamma(\rho)}{2}\big\|\Delta^{(t)}\big\|^2 - \frac{\rho}{2}\big\|\Delta^{(t-1)}\big\|^2. \qquad (2.8)
\end{aligned}$$

Based on Lemma 2.3.1, Lemma 2.3.2 gives the bound of the descent of the Lyapunov function.

Lemma 2.3.2. *Under the assumptions of Theorem 2.3.1, for any $\delta > 0$ we have*

$$\begin{aligned}
&\mathbf{F}(\mathbf{x}^{t+1}, \mathbf{x}^t) - \mathbf{F}(\mathbf{x}^t, \mathbf{x}^{t-1}) \\
&\leq \Big(\frac{3L}{2} - \frac{\gamma(\rho)}{2} + L\delta\tau\Big)\big\|\Delta^{(t)}\big\|^2 - \frac{\rho}{2}\big\|\Delta^{(t-1)}\big\|^2 \\
&\qquad + \frac{L}{\delta}\sum_{k=1}^{\tau}\big\|\Delta^{(t-k)}\big\|^2. \qquad (2.9)
\end{aligned}$$

Now that we have the descent of the Lyapunov function, to prove the convergence, we only need the boundedness of the function $\mathbf{F}$. This is proved by the following lemma.

Lemma 2.3.3. *Suppose the assumptions of Theorem 2.3.1 are satisfied. Then for $\mathbf{x}^t$ generated by (EDANNI), there exists some constants $\underline{\mathbf{F}}$ and $\bar{\mathbf{F}}$ such that*

$$+\infty > \bar{\mathbf{F}} > \mathbf{F}(\mathbf{x}^{t+1}, \mathbf{x}^t) > \underline{\mathbf{F}} > -\infty, \quad \forall t \geq 0.$$

The proofs of these lemmata are in the appendix. Now with these lemmata, the conclusions in Theorem 2.3.1 can be proved in Appendix 5.1.1.

Besides the convergence in the nonconvex setting, in the following theorem we show that the proposed algorithm converges linearly if $\mathbf{L}_j$ is strongly convex. Quite interestingly, comparing with the results of [32, 33], here the linear convergence is established without any statistical assumptions on the data stored in each local machine.

The linear convergence (in Theorem 2.3.2) relies on the strong convexity condition in the following assumption. Note that it is not required in Theorem 2.3.1.

Assumption 2.3.3. *For all $j \in [m]$, the function $\mathbf{L}_j$ is strongly convex with positive modulus σ^2, which means that*

$$\mathbf{L}_j(\mathbf{x}) \geq \mathbf{L}_j(\mathbf{y}) + \langle \nabla \mathbf{L}_j(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle + \frac{\sigma^2}{2}\|\mathbf{x} - \mathbf{y}\|^2,$$

for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^p$, $j \in [m]$.

Now we can give the second theorem.

Theorem 2.3.2. *Suppose Assumption 2.2.1, 2.3.1, and 2.3.3 are satisfied. If ρ is sufficiently large such that*

$$\frac{\delta_1 L + \frac{\rho}{2}(1+\delta_1)}{\frac{\rho}{2}(1+\delta_1) + \delta_1} + \frac{3L}{2} - \frac{\rho}{2} + L\delta\tau < 0 \tag{2.10}$$

and

$$\begin{aligned} &\frac{\delta_1 L + \frac{\rho}{2}(1+\delta_1)}{\frac{\rho}{2}(1+\delta_1) + \delta_1} + \frac{3L}{2} - \frac{\rho}{2} + L\delta\tau \\ &- \frac{\rho\eta}{2} + \Big(\frac{L}{\delta} + \frac{\frac{\delta_1}{2}L^2\tau}{\frac{\rho}{2}(1+\delta_1) + \delta_1}\Big)\frac{\eta^\tau - 1}{\eta - 1} < 0, \end{aligned} \tag{2.11}$$

for some $\delta > 0$ and $\delta_1 > (2L + \rho + 1)/\sigma^2$, then it holds for the sequence generated by (EDANNI) that

$$\begin{aligned} 0 &\leq \mathbf{F}(\mathbf{x}^{t+1}, \mathbf{x}^t) - \mathbf{F}(\mathbf{x}^*, \mathbf{x}^*) \\ &\leq \frac{1}{\eta^t}(\mathbf{F}(\mathbf{x}^1, \mathbf{x}^0) - \mathbf{F}(\mathbf{x}^*, \mathbf{x}^*)), \end{aligned} \tag{2.12}$$

where $\eta := 1 + \frac{1}{\frac{\rho}{2}(1+\delta_1)+\delta_1}$.

Remark 2.3.3. *Note the above conditions can be satisfied when ρ is sufficiently larger than the order of $L\tau$ and the exponential of τ and δ_1 is larger than the order of ρ/σ^2. Theorem 2.3.2 asserts that with the strong convexity of $\mathbf{L}_j$'s, the augmented optimality gap in* (2.12) *decreases linearly to zero under these conditions. Similar to the discussion*

in Remark 2.3.2, the conditions (2.10) *and* (2.11) *can also be converted to the equivalent conditions that do not contain δ, which are, however, of a quite complicated form that is hard to analyze. Therefore the intermediate constant δ is also used here to maintain the conditions of a relatively simpler form. Moreover, Assumption 2.3.3 can be replaced by only requiring: (a) each $\mathbf{L}_j$ is convex and (b) $\frac{1}{m}\sum_{j\in[m]}\mathbf{L}_j$ is strongly convex with modulus σ^2. These two conditions are weaker than Assumption 2.3.3 since they don't assume all $\mathbf{L}_j$'s are strongly convex.*

To prove Theorem 2.3.2, we need the following lemma to bound the optimality gap of Lyapunov function $\mathbf{F}$ by the progress of the iterates $\{\mathbf{x}^t\}$.

Lemma 2.3.4. *Suppose Assumption 2.2.1, 2.3.1, and 2.3.3 hold and we have $\delta_1 > (2L+\rho+1)/\sigma^2$ for some $\delta_1 > 0$, then it follows that*

$$\begin{aligned}
&\frac{1}{\frac{\rho}{2}(1+\delta_1)+\delta_1}\left(\mathbf{F}(\mathbf{x}^{t+1},\mathbf{x}^t)-\mathbf{F}(\mathbf{x}^*,\mathbf{x}^*)\right)\\
&\leq \frac{\delta_1 L+\frac{\rho}{2}(1+\delta_1)}{\frac{\rho}{2}(1+\delta_1)+\delta_1}\|\mathbf{x}^t-\mathbf{x}^{t+1}\|^2\\
&+\frac{1}{\frac{\rho}{2}(1+\delta_1)+\delta_1}\frac{\delta_1}{2m}L^2\sum_{j\in[m]}\|\mathbf{x}^{t_j}-\mathbf{x}^t\|^2. \qquad (2.13)
\end{aligned}$$

The proof of Lemma 2.3.4 is in Appendix 5.1.3. Based on it, we can prove Theorem 2.3.2 in Appendix 5.1.2.

2.3.1 Inexactly Solving the Subproblems

In this section we discuss the case where subproblem (2.3) is not solved exactly. The motivation is that in some practical applications, it may not be easy to exactly minimize the objective function. The following analysis shows that the convergence still holds true when there are small errors in solving the subproblems, thus implying the robustness of the proposed algorithm. Specifically, we assume subproblem (2.3) is solved with some error at iteration t; that is, there is an error $\boldsymbol{\epsilon}^t$ such that

$$\begin{aligned}
\boldsymbol{\epsilon}^t \in \nabla\mathbf{L}_1(\mathbf{x}^{t+1})+\frac{1}{m}\sum_{j\in[m]}\nabla\mathbf{L}_j(\mathbf{x}^{t_j})-\nabla\mathbf{L}_1(\mathbf{x}^{t_1})\\
+\partial h(\mathbf{x}^{t+1})+\rho\left(\mathbf{x}^{t+1}-\mathbf{x}^t\right), \qquad (2.14)
\end{aligned}$$

which is equivalent to

$$\mathbf{x}^{t+1} = \mathbf{Prox}_h\Big[\mathbf{x}^{t+1} - \Big(\nabla \mathbf{L}_1(\mathbf{x}^{t+1}) + \frac{1}{m}\sum_{j\in[m]} \nabla \mathbf{L}_j(\mathbf{x}^{t_j}) - \nabla \mathbf{L}_1(\mathbf{x}^{t_1}) + \rho\left(\mathbf{x}^{t+1} - \mathbf{x}^t\right) - \boldsymbol{\epsilon}^t\Big)\Big]. \tag{2.15}$$

Different from exactly solving the subproblem, there is now an extra term $\boldsymbol{\epsilon}^t$ in (2.15) as compared to (5.3). Note that in this section $\mathbf{x}^{t+1}$ denotes the *inexact* solution of subproblem (2.3).

First, we introduce the following assumption that gives the bound of the error term.

Assumption 2.3.4. *There exists a constant $c_1 > 0$ such that the error term in* (2.14) *satisfies*

$$\|\boldsymbol{\epsilon}^t\|^2 < c_1\big\|\Delta^{(t-1)}\big\|^2, \quad \textit{for} \;\; t > 0. \tag{2.16}$$

This assumption requires that the error in solving the subproblem is bounded by a constant times the progress of $\mathbf{x}^t$ at the previous iteration. We can first compute the norm of the right hand side of (2.14) and compare the square of the norm with the right hand side of Assumption 2.3.4. If there is one subgradient of h satisfying this inequality (which can be checked easily), then the estimation of the variable in the current iteration satisfies this assumption. Note that when $\Delta^{(t-1)} := \mathbf{x}^t - \mathbf{x}^{t-1} = 0$, it holds that $\mathbf{x}^t$ is a stationary point in the nonconvex scenario and $\mathbf{x}^t = \mathbf{x}^*$ in the strongly convex scenario. Similar to the case when the subproblem is solved exactly, we have the following corollary.

Corollary 2.3.1. *Let Assumptions 2.2.1, 2.3.1, 2.3.2(II), and 2.3.4 hold, and suppose ρ satisfies $\gamma(\rho) > 3L + 2L\delta\tau + 1$ and $\rho > 2L\tau/\delta + c_1$, for some constant $\delta > 0$. Then all conclusions in Theorem 2.3.1 hold true for the sequence generated by (EDANNI) with subproblems being solved inexactly, as quantified by* (2.16)*. Specifically,*

- **(Boundedness of Sequence).** *The gap between $\mathbf{x}^t$ and $\mathbf{x}^{t+1}$ converges to* 0*, i.e.,*

$$\lim_{t\to\infty} \mathbf{x}^{t+1} - \mathbf{x}^t = 0.$$

If $\mathbf{L}(\mathbf{x})$ is coercive, then the sequence $\{\mathbf{x}^t\}$ generated by Algorithm 1 is bounded.

- **(Convergence to Stationary Points).** *Every limit point of the iterates* $\{\mathbf{x}^t\}$ *generated by Algorithm 1 is a stationary point of problem* (2.2). *Furthermore,* $\left\|\widetilde{\nabla}_{\mathbf{x}}\mathbf{L}(\mathbf{x}^t)\right\| \to 0$, *as* $t \to \infty$.

- **(Sublinear Convergence Rate).** *Given* $\epsilon > 0$, *let us define* T *to be the first time for the optimality gap to reach below* ϵ. *Specifically, we define*

$$T := \operatorname*{argmin}_t \left\{ \left\|\widetilde{\nabla}_{\mathbf{x}}\mathbf{L}(\mathbf{x}^t)\right\|^2 < \epsilon \right\}.$$

Then there exists a constant $\nu > 0$ *such that*

$$T \leq \frac{\nu}{\epsilon} + 1, \tag{2.17}$$

where ν *equals to a positive constant times* $\frac{3\left((2+\rho)^2+4L^2\tau+c_1\right)}{\min\left\{\frac{\gamma(\rho)}{2}-\frac{3L}{2}-L\delta\tau-\frac{1}{2}, \frac{\rho}{2}-\frac{L\tau}{\delta}-\frac{c_1}{2}\right\}}$ *for some* $\delta > 0$. *Therefore, the optimality measure* $\left\|\widetilde{\nabla}_{\mathbf{x}}\mathbf{L}(\mathbf{x}^t)\right\|$ *converges to* 0 *in a sublinear manner.*

The idea of the proof of Corollary 2.3.1 is similar to that of the exact case. The proofs of corollaries are in the Appendix.

Results corresponding to Theorem 2.3.2 also holds in the scenario of solving the subproblems inexactly. In summary we have the following corollary.

Corollary 2.3.2. *Suppose Assumption 2.2.1, 2.3.1, 2.3.3, and 2.3.4 are satisfied. If* ρ *is sufficiently large such that*

$$\frac{\delta_1 L + \frac{\rho}{2}(1+\delta_1) + \frac{1}{2}}{\frac{\rho}{2}(1+\delta_1) + \delta_1} + \frac{3L}{2} - \frac{\rho - 1}{2} + L\delta\tau < 0$$

and

$$\begin{aligned}&\frac{\delta_1 L + \frac{\rho}{2}(1+\delta_1) + \frac{1}{2}}{\frac{\rho}{2}(1+\delta_1) + \delta_1} + \frac{3L}{2} - \frac{\rho - 1}{2} + L\delta\tau - \frac{(\rho - c_1)\eta}{2} \\ &+ \left(\frac{L}{\delta} + \frac{\frac{\delta_1}{2}L^2\tau}{\frac{\rho}{2}(1+\delta_1) + \delta_1}\right)\frac{\eta^\tau - 1}{\eta - 1} + \frac{c_1\eta}{\frac{\rho}{2}(1+\delta_1) + \delta_1} < 0,\end{aligned}$$

for some $\delta > 0$ *and* $\delta_1 > (2L + \rho + 1)/\sigma^2$, *then for the sequence generated by (EDANNI)*

with subproblems being solved inexactly we have

$$\begin{aligned} 0 &\leq \mathbf{F}(\mathbf{x}^{t+1}, \mathbf{x}^t) - \mathbf{F}(\mathbf{x}^*, \mathbf{x}^*) \\ &\leq \frac{1}{\eta^t}\Big(\mathbf{F}(\mathbf{x}^1, \mathbf{x}^0) - \mathbf{F}(\mathbf{x}^*, \mathbf{x}^*)\Big), \end{aligned}$$

where $\eta := 1 + \frac{1}{\frac{p}{2}(1+\delta_1)+\delta_1}$.

2.4 Experiments

Now we test our algorithm[2] on both a convex application (LASSO) and a nonconvex application (Sparse PCA). In both settings, we compare with various advanced algorithms:

(1) Efficient Distributed Learning with the Parameter Server (Parameter Server): the advanced proximal gradient descent method with the parameter server proposed in [18].

(2) Asynchronous Distributed ADMM (AD-ADMM): the ADMM based asynchronous algorithm proposed in [19].

(3) Efficient Distributed Algorithm for Nonconvex-Nonsmooth Inference (EDANNI): the proposed approach presented here.

We first compare their communication cost, that is, the total number of transmissions between the master and the workers. Then the effects of the asynchrony on the working time and the idle time are examined.

2.4.1 LASSO

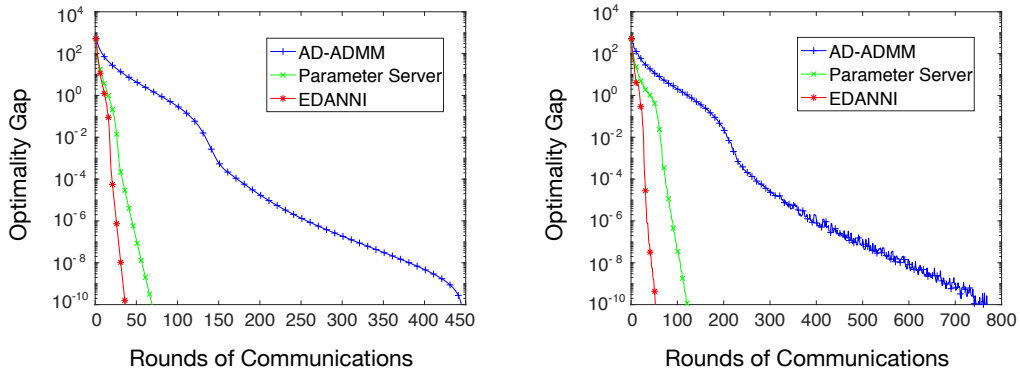
In this example, to demonstrate the convergence performance of the above algorithms in terms of communication rounds on convex problems, we consider the LASSO problem:

$$\underset{\mathbf{w}\in\mathbb{R}^p}{\operatorname{argmin}} \frac{1}{2mn} \sum_{j\in[m]} \sum_{i\in[n]} \left\|\mathbf{x}_{ji}^T\mathbf{w} - y_{ji}\right\|^2 + \theta\|\mathbf{w}\|_1, \tag{2.18}$$

[2]The code is available at the repository: `https://github.com/jackiejen/EDANNI_communication_efficient_distributed_algorithm`

Table 2.1: Comparison of the communication cost for LASSO

Type	Parameters (m,n,p,s,θ)	Method	Communication
Synchronous	(10,10000,500,5,0.01)	**AD-ADMM**	20.3
		PS	1.8
		EDANNI	**1**
	(20,500,500,5,0.01)	**AD-ADMM**	34.1
		PS	1.3
		EDANNI	**1**
	(20,500,1000,10,0.01)	**AD-ADMM**	8.3
		PS	1.4
		EDANNI	**1**
Asynchronous ($\tau = 10$)	(10,10000,500,5,0.01)	**AD-ADMM**	21.1
		PS	1.6
		EDANNI	**1**
	(20,500,500,5,0.01)	**AD-ADMM**	35.1
		PS	1.3
		EDANNI	**1**
	(20,500,1000,10,0.01)	**AD-ADMM**	6.7
		PS	1.5
		EDANNI	**1**

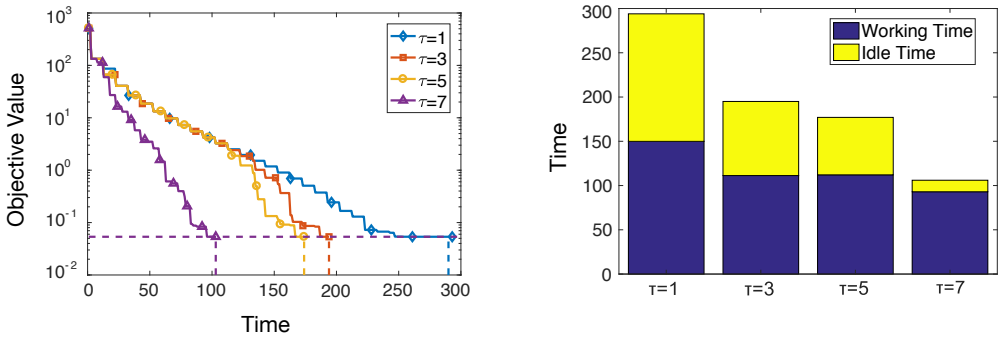


(a) Synchronous, **(12.3, 2.0, 1)** (b) Asynchronous ($\tau = 3$), **(14.6, 2.3, 1)**

Figure 2.2: Comparison of candidate algorithms in LASSO when $m = 10$, $n = 10000$, $p = 1000$, $s = 20$, $\theta = 0.01$, $\rho = 12$, and $\mathbf{X} \sim \mathcal{N}(0, \Sigma)$.

where $\theta > 0$ is the coefficient of the regularizer. Here one communication round denotes one round of back-and-forth information exchange between the master and the workers, which is actually equivalent to one iteration in Algorithm 1. Note that from now on we switch the notation to let the unknown quantity be denoted by $\mathbf{w}$ instead of $\mathbf{x}$.

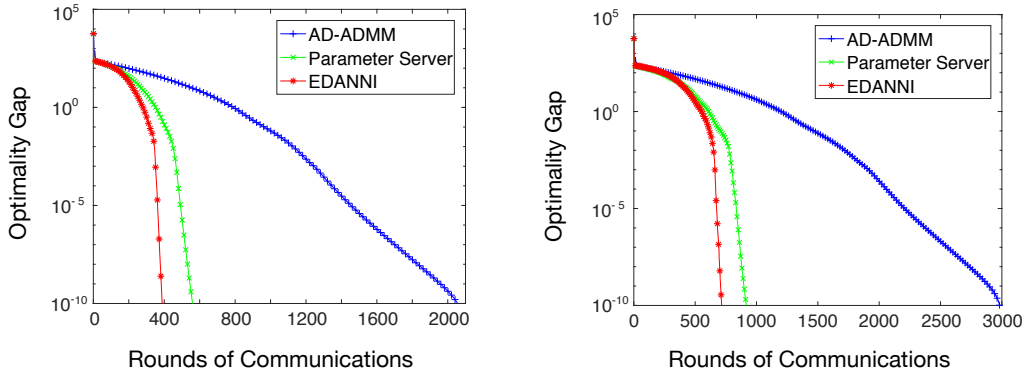
The data $\{\mathbf{x}_{ji}\}_{i\in[n],j\in[m]}$ is independently sampled from a multivariate Gaussian



(a) Convergence for different τ's

(b) Time table for different τ's

Figure 2.3: Comparison of EDANNI in LASSO with different settings of the delay bound τ when $\rho = 12$.



(a) Synchronous, **(6.0, 1.7, 1)**

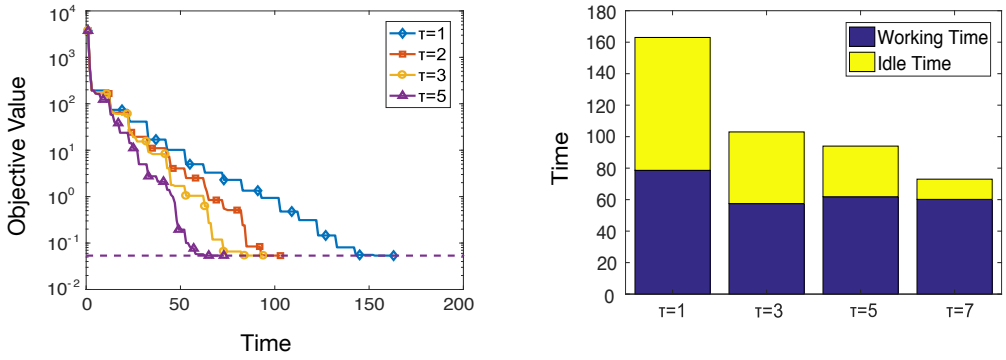
(b) Asynchronous ($\tau = 2$), **(4.2, 1.4, 1)**

Figure 2.4: Comparison of candidate algorithms in LASSO when $m = 10$, $n = 1000$, $p = 12000$, $s = 1000$, $\theta = 0.01$, $\rho = 14$, and $\mathbf{X} \sim \mathcal{N}(0, \Sigma)$.

distribution with zero mean and covariance matrix Σ. For $r \in [p], t \in [p]$, the rt-th entry of covariance matrix is set to be: $|\Sigma_{rt}| = 0.5^{|r-t|}$. The corresponding y_{ji} is constructed by

$$y_{ji} = \mathbf{x}_{ji}^T \mathbf{w}^* + \epsilon_{ji}, \quad \forall j \in [m], i \in [n],$$

where noise ϵ_{ji} is a zero mean Gaussian random variable with variance 0.01. The true



(a) Convergence for different τ's

(b) Time table for different τ's

Figure 2.5: Comparison of EDANNI in LASSO with different settings of the delay bound τ when $\rho = 14$.

parameter $\mathbf{w}^*$ is s-sparse where all the entries are zero except that the first s entries are i.i.d random variables from a uniform distribution in [0,1].

In Figure 2.2, the algorithms are compared in the setting where $m = 10$, $n = 10000$, $p = 1000$, $s = 20$, and $\theta = 0.01$. The EDANNI is implemented with Matlab Parallel Computing Toolbox over an HP Linux Distributed Cluster (Mesabi HPC in University of Minnesota)[3]. Specifically, there are 12 CPU cores on each HP server, each of them can be assigned as one worker/master node. Here we set $\rho = 12$ and the assumptions in Theorem 2.3.2 are satisfied with $L = 1.10$, $\sigma^2 = 0.35$, $\delta = 1$. The y-axis denotes the optimality gap of the objective function value. Both the synchronous scenario and the asynchronous scenario are considered. Here the tolerable delay τ is set to be 3 and the master will update the variables once the workers j with $d_j > \tau - 1$ have been arrived. To simulate the asynchronous case, at each iteration half of the workers are assumed to be arrived with probability 0.2 and the other workers are assumed be arrived with probability 0.5.

One can observe from Figure 2.2 that the proposed algorithm indeed converges faster than AD-ADMM and Parameter Server in terms of communication rounds, in both synchronous scenarios and asynchronous scenarios, but the gap between EDANNI and Parameter Server is smaller than that between EDANNI and AD-ADMM. The

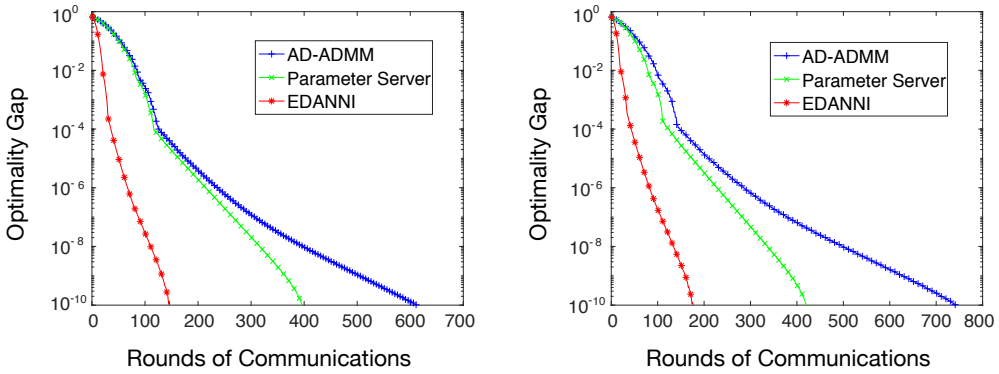
[3] `https://www.msi.umn.edu/content/mesabi`

triple of numbers in each figure caption indicates the communication cost needed for AD-ADMM, Parameter Server, and EDANNI to attain the minimum objective value with error less than 10^{-6}. For simplicity, we scale the communication complexity of EDANNI to 1. The results for other settings of m, n, p, s are summarized in Table 2.1. It is shown in these results that EDANNI is the most communication-efficient among the three algorithms.

Figure 2.3 shows the performance of the proposed approach when we choose different maximum tolerable delay τ. It can be observed from Figure 2.3 (a) that the convergence rate varies much with different values of τ. EDANNI converges faster when τ is larger, and converges relatively slower when $\tau = 0$ (i.e., the synchronous case). The results in Figure 2.3 (b) shows that the ratio of the computing time over the idle time increases when the delay bound τ becomes larger, therefore speeding up the convergence. Here the asynchronous implementation is simulated by randomly setting the computation speed for each node from a uniform $[1, 10]$ distribution.

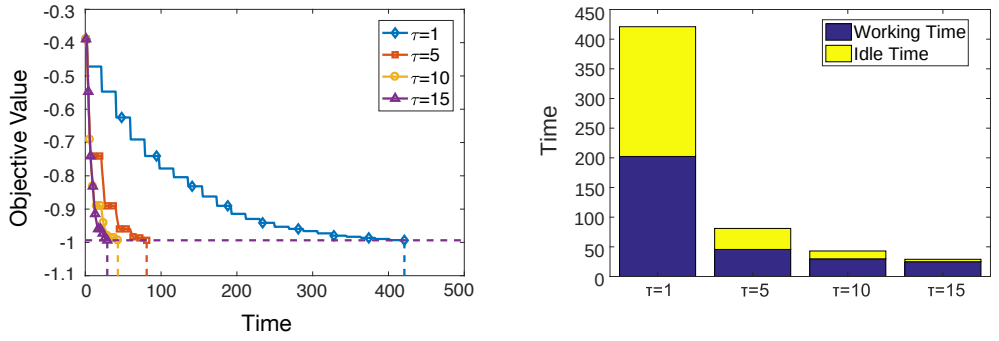
Note that the coefficient matrices in the LASSO problems considered above are undercomplete so that they are strongly convex and Theorem 2.3.2 can be applied. In Figure 2.4 and Figure 2.5, we also show the performance of the competing algorithms on weakly convex LASSO problems with high dimensional variables (m = 10, n=1000, p=12000). We comment that the best performance of the proposed algorithm on weakly convex cases hasn't been studied yet. It is an interesting direction for future study. By now the convergence of the proposed algorithm on these weakly convex applications is guaranteed by Theorem 2.3.1. The conditions in Assumption 2.3.2 are satisfied with $L = 1.92$, $\rho = 14$, $\delta = 0.90$. It can be observed in Figure 2.4 that EDANNI again converges faster than other algorithms. To attain an accuracy of 10^{-10} in the synchronous case, it needs 388 rounds of communication, while Parameter Server needs 559 and AD-ADMM needs 2051. Similarly, for the asynchronous case, Figure 2.4 (b) shows that the proposed algorithm needs significantly less communication rounds to achieve a given accuracy. Further, Figure 2.5 compares the convergence and time table of the proposed algorithm for different choices of the delay bound τ. It shows that when the τ's are chosen to be moderate, the convergence time increases as τ decreases. Sometimes larger τ may also result in longer working time as shown in the cases when $\tau = 3$ and $\tau = 5$ in Figure 2.5 (b). However, since the idle time is distinctly longer when τ is smaller,

we can observe that the total convergence time when $\tau = 5$ is still less than that when $\tau = 3$.



(a) Synchronous, **(4.2, 2.7, 1)** (b) Asynchronous ($\tau = 8$), **(4.3, 2.4, 1)**

Figure 2.6: Comparison of candidate algorithms in sparse PCA when $m = 16$, $n = 8000$, $p = 50$, $q = 100$, $s = 120$, $\theta = 0.1$.

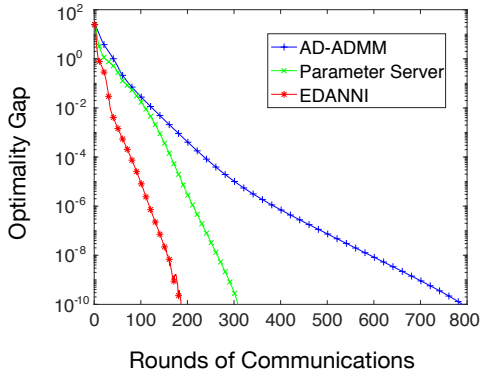


(a) Convergence for different τ's (b) Time table for different τ's

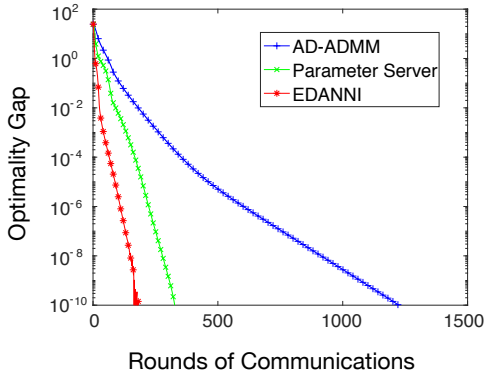
Figure 2.7: Comparison of EDANNI in sparse PCA with different settings of the delay bound τ when $\rho = 20$.

2.4.2 Sparse PCA

Section 2.4.1 shows the results for convex nonsmooth problems. To verify the convergence for nonconvex nonsmooth problems, we consider the following sparse PCA problem [31]:

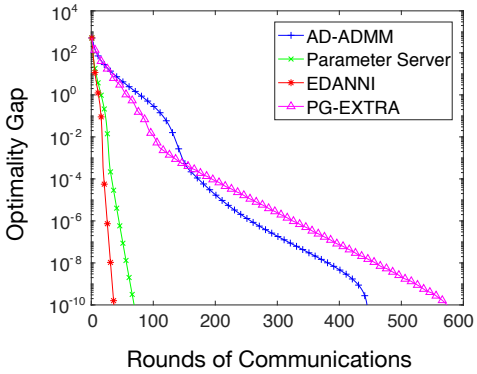


(a) Synchronous, **(5.2, 1.8, 1)**

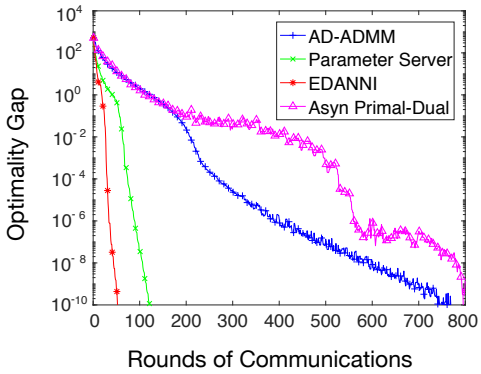


(b) Asynchronous ($\tau = 3$), **(6.5, 1.9, 1)**

Figure 2.8: Comparison of candidate algorithms in sparse PCA when $m = 8$, $n = 1000$, $p = 10000$, $q = 50$, $s = 2000$, $\theta = 0.01$.



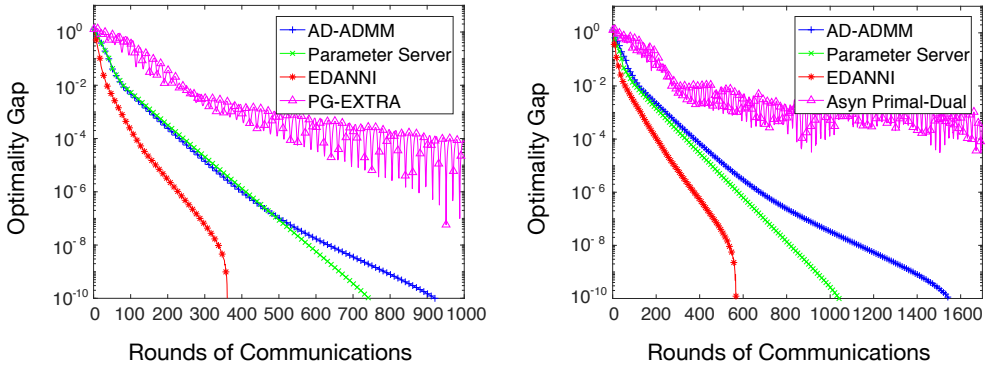
(a) Synchronous, (12.3, 2.0, **1**, 15.8)



(b) Asynchronous ($\tau = 3$), (14.6, 2.3, **1**, 15.6)

Figure 2.9: Comparison of candidate algorithms in LASSO when $m = 10$, $n = 10000$, $p = 1000$, $s = 20$, $\rho = 12$, and $\theta = 0.01$.

$$\begin{aligned} \underset{\mathbf{w}\in\mathbb{R}^p}{\operatorname{argmin}} & -\frac{1}{mn}\sum_{j\in[m]}\sum_{i\in[n]}\mathbf{w}^T B_{ji}B_{ji}^T\mathbf{w} + \theta\|\mathbf{w}\|_1, \\ \text{s.t.} \quad & \|\mathbf{w}\| \leq 1 \end{aligned} \tag{2.19}$$



(a) Synchronous, (2.6, 2.1, **1**, >10) (b) Asynchronous ($\tau = 4$), (2.7, 1.9, **1**, >10)

Figure 2.10: Comparison of candidate algorithms in sparse PCA when $m = 8$, $n = 20000$, $p = 10$, $q = 100$, $s = 30$, and $\theta = 0.1$.

where $B_{ji} \in \mathbb{R}^{p\times q}$ is a sparse matrix, $\forall j \in [m],\ \ i \in [n]$, and the regularization coefficient $\theta > 0$. Note that this is not a convex problem. The function $h(\mathbf{x})$ in (2.2) corresponds to $\theta\|\mathbf{w}\|_1 + \mathbf{I}_{\|\mathbf{w}\|\leq 1}(\mathbf{w})$ in this problem. In the first example, we set $m = 16$, $n = 8000$, $p = 50$, $q = 100$, and $\theta = 0.1$. Each matrix $B_{ji} \in \mathbb{R}^{50\times 100}$ is a sparse normally distributed random matrix with $s = 120$ non-zero entries. The parameter ρ in (2.3) is set to be 2 times the eigenvalue of $\sum_{i\in[n],j\in[m]} B_{ji}B_{ji}^\top$ with the largest magnitude. It can be checked that the conditions in Assumption 2.3.2 are satisfied with $L = 1.23$, $\rho = 39$, $\delta = 0.91$ in the asynchronous case. The algorithms are compared in both the synchronous and the asynchronous scenarios. One can see from Figure 2.6 that the proposed approach converges faster than AD-ADMM and Parameter Server with much less communication cost. The results for other settings of m, n, p, q, s in Table 2.2 also verify such a conclusion.

The performance of the proposed approach with different maximum tolerable delay τ is summarized in Figure 2.7. Here the distributed implementation is simulated in the same way as in the LASSO case. In Figure 2.7 we set $m = 6$, $n = 20$, $p = 100$, $q = 1000$. One can observe from Figure 2.7 (a) that in this example the convergence rate in terms of time is indeed affected by values of τ. The running time when $\tau = 15$ is much less than that when τ is small. Similar to the LASSO case, Figure 2.7 (b) shows that the ratio of the computing time in the overall running time increases closely to 1 when the

delay bound τ becomes 15, therefore speeding up the convergence. Such results can also be observed for other choices of parameters m, n, p, q, s, θ.

In Figure 2.8 we demonstrate the convergence results of the competing algorithms for sparse PCA problems with high dimensional variables ($p = 10000$, $s = 2000$). In this case, the matrices $B_{ji}B_{ji}^\top$ can be semi-positive definite. The conditions in Assumption 2.3.2 are satisfied with $L = 1.43$, $\rho = 12$, $\delta = 0.80$ in this setting. It is shown in Figure 2.8 that the proposed algorithm also achieves better communication efficiency than other competing algorithms. Moreover, Figure 14 in the supplement of [10] compares the convergence and time table of the proposed algorithm for different choices of the delay bound τ when solving sparse PCA problems. It can be observed that the total convergence time increases when τ decreases due to the significantly longer idle time as shown in Figure 14 (b).

Table 2.2: Comparison of the communication cost for SPCA

Type	**Parameters** (m,n,p,q,s,θ)	**Method**	**Communication**
Synchronous	(12,10000,20,100,20,0.1)	**AD-ADMM**	7.8
		PS	2.6
		EDANNI	**1**
	(30,200,50,100,50,0.1)	**AD-ADMM**	30.0
		PS	2.5
		EDANNI	**1**
	(3,20,500,1000,500,0.1)	**AD-ADMM**	5.2
		PS	1.7
		EDANNI	**1**
Asynchronous	(12,10000,20,100,20,0.1) ($\tau = 3$)	**AD-ADMM**	7.9
		PS	3.2
		EDANNI	**1**
	(30,200,50,100,50,0.1) ($\tau = 10$)	**AD-ADMM**	31.1
		PS	7.2
		EDANNI	**1**
	(3,20,500,1000,500,0.1) ($\tau = 3$)	**AD-ADMM**	6.3
		PS	2.0
		EDANNI	**1**

2.4.3 Additional Experiments

Above we mainly compare the distributed algorithms with the same hierarchical (master/worker) structure. In the following, we also compare with some advanced fully decentralized algorithms, PG-EXTRA [42] and Asynchronous Primal-Dual [51], that are also applicable to nonsmooth problems. As discussed in the Introduction section, by now there are no fully decentralized asynchronous methods that have guaranteed

convergence rate for solving nonsmooth problems in both strongly convex and nonconvex settings that are considered in this paper. Moreover, compared with algorithms with the master/worker structure, fully decentralized algorithms also have higher computation complexity since all nodes need to solve nonsmooth subproblems to update their local variable estimation. Here for the purpose of comparison only, we still compare the proposed method with the advanced fully decentralized algorithms. In the synchronous setting, we compare with the algorithm PG-EXTRA in [42]. Moreover, it is known that the PG-EXTRA diverges without synchronization. Therefore, in asynchronous scenarios we compare it with another advanced algorithm called the Asynchronous Primal-Dual method [51] whose synchronous version is comparable to PG-EXTRA. As above discussions, the assumptions in Theorem 2.3.2 are satisfied with $L = 1.10$, $\rho = 12$, $\sigma^2 = 0.35$, $\delta = 1$ in the LASSO problems. It is shown in Figure 2.9 that EDANNI and Parameter Server converge faster than AD-ADMM and the fully decentralized algorithms, PG-EXTRA and Asynchronous Primal-Dual. Furthermore, compared with the other three approaches, the proposed method requires the lowest communication cost.

For the sparse PCA problems here, the assumptions in Theorem 2.3.1 are satisfied with $L = 1.21$, $\rho = 14$, $\delta = 0.85$. It can be observed in Figure 2.10 that the fully decentralized algorithms converge slower than the other three approaches in both synchronous cases and asynchronous cases. Furthermore, it is also shown that EDANNI consistently achieves better communication efficiency than other competing algorithms. Similar results can be observed generally in Table 2.3 for other choices of parameters. We also refer to Figure 12 and Figure 13 in the supplement of [10] for more results.

In Figure 2.11 we test the acceleration of the algorithm over increasing numbers of the nodes used in the distributed implementation for solving the sparse PCA problems. When increasing the number of nodes to 16, the algorithm achieves 7.8 fold speedup. The gap between the actual speedup and the ideal one is due to the overheads of the variable updates by the master and the communications. Similar speedup can be observed for solving the LASSO problems. The results verify that the distributed implementation indeed saves the runtime significantly compared with the serial implementation.

2.5 Conclusions and Remarks

This chapter proposes a communication-efficient distributed algorithm (EDANNI) solving a general problem (2.2) in signal processing and machine learning under an asynchronous protocol. Theoretically, we prove the proposed algorithm converges to a stationary point in a sublinear rate, even in nonconvex nonsmooth scenarios. Moreover, unlike the previous work, the linear convergence is established in strongly convex scenarios without any statistical assumptions on the local data. In experiments, we compare EDANNI with other state-of-the-art distributed algorithms in different applications, and the results show the superior performance of the proposed algorithm in terms of the communication efficiency in both synchronous and asynchronous scenarios. Note that here we study an asynchronous distributed algorithm with a master/worker structure. One interesting direction for future research would be to consider the generalization of the proposed method to other decentralized structures like the ones used in [25, 45, 46]. Moreover, this work gives the convergence conditions for the worst cases, so there exists a gap between the practical performance and the theoretical results. Therefore it is also meaningful to exploit the structure of specific problems to tighten the bound.

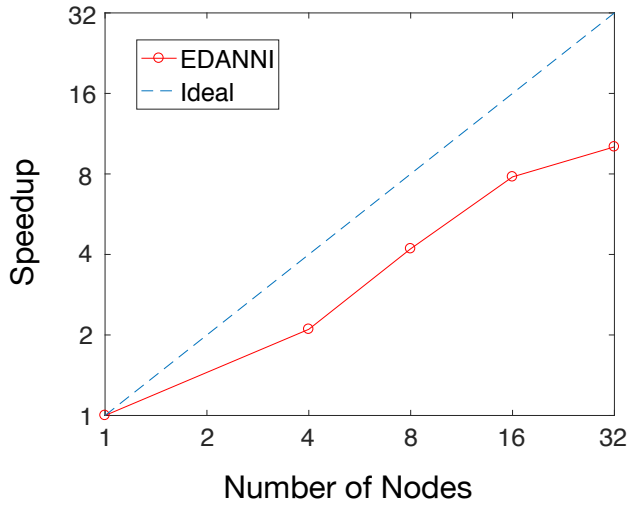


Figure 2.11: The speedup of the proposed algorithm with different numbers of workers when $m * n = 80000$, $p = 20$, $q = 100$, $s = 30$, and $\theta = 0.1$.

Table 2.3: Comparison of the communication cost for various settings of m, n, s, p, q, θ.

Convex	**Parameters** (m,n,p,s,θ)	**Method**	**Communication**
LASSO	(10,10000,500,5,0.01) (Syn)	**PG-EXTRA**	19.5
		AD-ADMM	20.3
		PS	1.8
		EDANNI	**1**
	(20,5000,500,5,0.01) (Asyn, $\tau = 10$)	**Asyn Primal-Dual**	27.6
		AD-ADMM	32.1
		PS	1.9
		EDANNI	**1**
Nonconvex	**Parameters** (m,n,p,q,s,θ)	**Method**	**Communication**
Sparse PCA	(12,10000,50,100,10,0.1) (Syn)	**PG-EXTRA**	22.3
		AD-ADMM	7.8
		PS	2.6
		EDANNI	**1**
	(20,5000,50,20,10,0.1) (Asyn, $\tau = 10$)	**Asyn Primal-Dual**	12.7
		AD-ADMM	5.3
		PS	3.1
		EDANNI	**1**

Chapter 3

Hierarchical Communication Structures of Multi-Agent Policy Evaluation Algorithms

Policy evaluation problems in multi-agent reinforcement learning (MARL) have attracted growing interest recently. In this setting, agents collaborate to learn the value of a given policy with private local rewards and jointly observed state-action pairs. However, existing fully decentralized algorithms treat each agent equally, without considering the communication structure of the agents over a given network, and the corresponding effects on communication and computation efficiency. In this chapter, we propose a hierarchical distributed algorithm that differentiates the roles of each of the agents during the evaluation process. This method allows us to freely choose various mixing schemes (and corresponding mixing matrices that are not necessarily symmetric or doubly stochastic), in order to reduce the communication and computation cost, while still maintaining convergence at rates as fast as or even faster than the previous distributed algorithms. Theoretically, we show the proposed method, which contains existing distributed methods as a special case, achieves the same order of convergence rate as state-of-the-art methods. Extensive numerical experiments on real datasets verify that the performance of our approach indeed improves - sometimes significantly - over other advanced algorithms in terms of convergence and total communication efficiency. The complete version of the works in this chapter has been accepted for publication on the Journal of Computational Science with title "Communication-Efficient Hierarchical Distributed Optimization for Multi-Agent Policy Evaluation".

3.1 Introduction

Reinforcement learning has recently witnessed enormous progress in applications such as strategy games [76] and intelligent control [77]. In single-agent reinforcement learning tasks, an agent uses function approximations or deep neural networks to learn from the environment and adaptively makes optimal decisions that maximize a reward based on samples. Compared with single-agent reinforcement learning, multi-agent reinforcement learning (MARL) tasks are more challenging since they require that each agent interacts with not only the environment, but also other agents. In this chapter, we study the policy evaluation problem in MARL with local rewards. In such scenarios, all the agents share a joint state whose transition depends on the local rewards and actions of individual agents. However, because of practical constraints, each agent only observes its own local rewards but does not know those of the others. To achieve the optimal global rewards, which is the sum of local rewards, the agents need to exchange their information with others. This type of setting is motivated by broad applications like traffic signal control [7], sensor networks [3], controller synchronization [78], and swarm robotics [8].

A trivial decentralized approach is to assign a central controller that collects and broadcasts the reward information, and determines the action of each agent. However such a structure is very susceptible to the possible failure of the central node. Furthermore, the scalability considerations of the central agent make this approach challenging in large-scale applications when it does not have sufficient computing and storage ability. Moreover, in applications where local rewards are not accessible due to privacy concerns, this structure is no longer feasible.

To make the policy evaluation in MARL more scalable and robust, existing works [79, 80] propose fully decentralized methods, where each agent only communicates with its neighbors over a network. However, like most of the predecessors, their approach requires undirected connections or doubly stochastic mixing matrices generated directly from the adjacency matrices of undirected graphs. For directed graphs (digraphs), however, only a special type of them admits a doubly stochastic mixing matrix, and even for them, a general method to construct such matrices is still lacking [81, 82]. Moreover, in previous work that requires doubly stochastic mixing matrices, all agents' roles are essentially the same in the sense that they all need to send variables and

gradients information to neighbors and receive such information from them. However, for these fully decentralized methods, the agents may communicate too frequently when they are already well connected. In this chapter, a hierarchical algorithm is proposed to handle these problems. The proposed algorithm has two advantages. One advantage is that it only requires row or column stochastic mixing matrices (here a matrix A is said to be row (column) stochastic if $A\mathbf{1} = \mathbf{1}$ ($\mathbf{1}^\top A = \mathbf{1}^\top$)). Such an improvement makes it easily applicable to a hierarchical communication structures and digraphs. Moreover, by properly designing the mixing matrices in the proposed algorithm, it is possible for an agent to receive information from another agent to update its variable without sending out such information to that agent; similarly, it can send its own gradient information to that agent without receiving gradient information from it. In this way, agents over the network are differentiated and play different roles in the evaluation process (analogous to the Master-Workers relationship [18] in a star network with a central node). Another advantage of the proposed algorithm is that it not only includes the mixing scheme in the previous work as a special case, but also includes other mixing schemes that have not been considered before. Our experiments show that the hierarchical structure and the new mixing schemes can save communication costs in each iteration and still have better convergence performance than the previous approaches. Theoretically, we prove that the proposed algorithm converges linearly with the same order of rate as the state-of-the-art approach though it is much more general than the latter.

As we know, the tracking technique has been frequently used in general decentralized gradient descent methods such as [44, 83]. Among those [84] studies uncoordinated step-size , [56] handles non-convexity, and [45] considers time-varying graphs. Moreover, decentralized variance reduction methods like SAGA and SVRG using the tracking technique are considered by [85]. However, these works require that the communication matrices are doubly stochastic. Therefore, recently, gradient descent methods under protocols like the push-sum and the push-pull are proposed by [46, 86] to eliminate the need of constructing a doubly stochastic matrix in reaching consensus. They mainly focus on minimizing a general sum of convex local cost functions. Moreover, hierarchical structures and their merits on communication efficiency have been observed in recent works on decentralized ADMM [24, 25]. In this work, we initially adopt a different hierarchical structure to solve the convex-concave saddle-point problems and study its

benefits on saving communication costs. Specifically, we use row stochastic matrices for mixing the variables and column stochastic matrices for tracking the average gradients. This structure is shown to work in both fully decentralized graphs and two-tier (hierarchical) graphs.

Related Work The study of MARL dates back to [87, 88]. For more recent works see also [89–91]. However, most of them suffer from the curse of dimensionality since they only consider the tabular setting, where the value functions defined in the state and action spaces can be represented as arrays, or tables. To resolve this issue, policy evaluation with linear function approximation and actor-critic algorithms (that learn approximations to both policy and value functions, where "actor" refers to the learned policy, and "critic" refers to the learned value function) are studied by [79, 80] and [92] respectively under the collaborative MARL framework. Moreover, recent work [93] considered a communication-efficient distributed reinforcement learning algorithm that involves a central controller and corresponding learners. Besides, works like [94] used deep neural networks as function approximators, which are known as deep MARL. Though they work well empirically, most of them lack theoretical guarantees.

For the policy evaluation in signal-agent RL, the primal-dual formulation is studied in papers like [95–97]. In those works, the authors consider concentrated algorithms that work on one node or in systems with a central control node. In that case, the node in general requires much higher computing and storage capabilities and may not be feasible due to privacy concerns or network availability in practical applications. Therefore, [80, 98] further considered this formulation in the multi-agent setting. Our work is more related to [80] in the sense that both develop a distributed primal dual algorithm solving the convex-concave formulation of the policy evaluation problem in MARL. Both algorithms achieve the geometric rate of convergence in the setting with batched samples. In contrast, our algorithm is more general in that it allows hierarchical structures of communication. More flexibility of choosing mixing schemes and information exchange patterns not only makes the proposed algorithm applicable on directed graphs, but also leads to significant communication saving. In the classical works that generally minimize a sum of convex local cost functions such as [13, 43, 44, 46, 99, 100], our algorithm is closely related to stochastic average or incremental gradient [46, 100, 101], with the difference that our objective function is a double sum of convex-concave functions

and we consider hierarchical structures and new mixing schemes as well as their effects on communication efficiency. To the best of our knowledge, the proposed algorithm is the first to work on directed graphs with hierarchical structures, and allows rich options of mixing matrices and schemes to solve decentralized convex-concave saddle-point problems in MARL.

Contribution In summary, this work has the following contributions: (i) We propose an general decentralized algorithm that accommodates both fully decentralized and hierarchical structures. (ii) We prove the deterministic geometric convergence rate for the general framework. (iii) We design mixing schemes and matrices that make the proposed algorithm perform better and more efficiently than other advanced competing algorithms in experiments. (iv) The proposed framework and analysis are of independent interest for solving general saddle-point problems with convex-concave cost where the data samples are stored on decentralized agents.

Notation For a vector $\mathbf{a} = (a_1, \cdots, a_s)^\top \in \mathbb{R}^s$ we denote $\|\mathbf{a}\| := (\sum_{i=1}^s |a_i|^2)^{1/2}$. For a matrix $\mathbf{A} = [a_{ij}] \in \mathbb{R}^{s\times k}$, we define

$$\|\mathbf{A}\| := (\sum_{i=1}^{s}\sum_{j=1}^{k} |a_{ij}|^2)^{1/2}, \quad \|\mathbf{A}\|_{1,\infty} := \max_{i=1,\cdots,s}\{\sum_{j=1}^{k} |a_{ij}|\},$$

and denote its spectral norm by $\|\mathbf{A}\|_S$. Given a positive semidefinite matrix $\mathbf{H} \in \mathbb{R}^{s\times s}$, norm $\|\cdot\|_{\mathbf{H}}$ is defined as $\|\mathbf{v}\|_{\mathbf{H}} = \sqrt{\mathbf{v}^\top \mathbf{H}\mathbf{v}}$ for any vector $\mathbf{v} \in \mathbb{R}^s$.

3.2 Problem Formulation

In this section, we introduce the multi-agent Markov decision process (MDP) as the generalization of the single-agent MDP. Then as shown in [97], we can reformulate the policy evaluation problem as a primal-dual convex-concave optimization problem.

In a finite MDP for a single agent, we denote the sets of states, actions, and rewards by $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{R}$ [102]. Given time t, the random variables $\mathbf{R}_t$ and $\mathbf{S}_t$ are dependent on the preceding state and action. Specifically, given values of the preceding state and action, the probability for $\mathbf{s}' \in \mathcal{S}$ and $\mathbf{r} \in \mathcal{R}$ to occur at time t is given by

$$p(\mathbf{s}', \mathbf{r}|\mathbf{s}, \mathbf{a}) := Pr\big\{\mathbf{S}_t = \mathbf{s}', \mathbf{R}_t = \mathbf{r}|\mathbf{S}_{t-1} = \mathbf{s}, \mathbf{A}_{t-1} = \mathbf{a}\big\},$$

for all $\mathbf{s}', \mathbf{s} \in \mathcal{S}$, $\mathbf{r} \in \mathcal{R}$, and $\mathbf{a} \in \mathcal{A}(\mathbf{s})$. The dynamics of the MDP is determined by the four-argument function p. From this dynamics function, we can compute the state-transition probabilities:

$$\begin{aligned} p(\mathbf{s}'|\mathbf{s}, \mathbf{a}) :=& Pr\big\{\mathbf{S}_t = \mathbf{s}'|\mathbf{S}_{t-1} = \mathbf{s}, \mathbf{A}_{t-1} = \mathbf{a}\big\} \\ =& \textstyle\sum_{\mathbf{r}\in\mathcal{R}} p(\mathbf{s}', \mathbf{r}|\mathbf{s}, \mathbf{a}). \end{aligned}$$

The state transition matrix $\mathcal{P}^{\mathbf{a}}$ under action $\mathbf{a}$ is defined by $[\mathcal{P}^{\mathbf{a}}]_{\mathbf{s},\mathbf{s}'} = p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$. The expected rewards for state-action pairs can be defined as a two-argument function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$:

$$\begin{aligned} r(\mathbf{s}, \mathbf{a}) :=& \mathbb{E}\big[\mathbf{R}_t|\mathbf{S}_{t-1} = \mathbf{s}, \mathbf{A}_{t-1} = \mathbf{a}\big] \\ =& \textstyle\sum_{\mathbf{r}\in\mathcal{R}} \mathbf{r} \sum_{\mathbf{s}'\in\mathcal{S}} p(\mathbf{s}', \mathbf{r}|\mathbf{s}, \mathbf{a}). \end{aligned} \tag{3.1}$$

Now consider a network of N agents. We are interested in the multi-agent finite MDP:

$$\big(p, \mathcal{S}, \{\mathcal{A}_i\}_{i=1}^N, \{\mathcal{R}_i\}_{i=1}^N, \gamma\big),$$

where $\mathcal{S}$ denotes the state space, $\mathcal{A}_i$ is the action space for agent i, $\mathcal{R}_i$ is the reward space for agent i, and $\gamma \in (0, 1)$ is the discount factor. The function p, as in the single-agent case, determines the dynamics of the MDP. However, its four arguments are now the joint states $\mathbf{s}, \mathbf{s}' \in \mathcal{S}$, joint action $\mathbf{a} := (\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_N) \in \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_N$, and joint reward $\mathbf{r} := (\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_N) \in \mathcal{R}_1 \times \mathcal{R}_2 \times \cdots \times \mathcal{R}_N$. The function $r_i(\mathbf{s}, \mathbf{a})$ can be defined similarly by (3.1) with $\mathcal{R}$ being replaced by $\mathcal{R}_i$. It is the local reward of agent i after taking joint action $\mathbf{a}$ at state $\mathbf{s}$. Both $\mathbf{s}$ and $\mathbf{a}$ are observable by all agents, while the reward r_i is private for agent i.

We know in this setting the agents are coupled together by the state transition matrix $\mathcal{P}^{\mathbf{a}} \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$ defined above under a joint action $\mathbf{a}$. As a motivation, this scenario arises from large-scale applications such as sensor networks [3, 103], robotics [104], and power grids [1]. Moreover, a policy $\boldsymbol{\pi}(\mathbf{a}|\mathbf{s})$ is the conditional probability of taking joint action $\mathbf{a}$ given the current state $\mathbf{s}$. We define the reward function of $\boldsymbol{\pi}$ as an average of the

local rewards:

$$R_c^{\boldsymbol{\pi}}(\mathbf{s}) := \tfrac{1}{N}\sum_{i=1}^{N} R_i^{\boldsymbol{\pi}}(\mathbf{s}), \tag{3.2}$$

where $R_i^{\boldsymbol{\pi}}(\mathbf{s}) := \mathbb{E}_{\mathbf{a}\sim\boldsymbol{\pi}(\cdot|\mathbf{s})}\big[r_i(\mathbf{s},\mathbf{a})\big]$.

The goal for the agents is to collaboratively find a joint action-selection $\boldsymbol{\pi}$ that maximizes the global return. Thus, without a central controller, it is essential for them to exchange information with each other so as to attain the global minimum. Note that any fixed policy $\boldsymbol{\pi}$ induces a transition matrix $\mathbf{P}^{\boldsymbol{\pi}}$ over $\mathcal{S}$, whose $(\mathbf{s},\mathbf{s}')$-th element is given by

$$[\mathbf{P}^{\boldsymbol{\pi}}]_{\mathbf{s},\mathbf{s}'} := \mathbb{E}_{\boldsymbol{\pi}(\cdot|\mathbf{s})}[\mathcal{P}^{\mathbf{a}}]_{\mathbf{s},\mathbf{s}'} = \sum_{\mathbf{a}\in\mathcal{A}} \boldsymbol{\pi}(\mathbf{a}|\mathbf{s}) p(\mathbf{s}'|\mathbf{s},\mathbf{a}). \tag{3.3}$$

Policy Evaluation A key step in reinforcement learning is policy evaluation. Efficient estimation of the value function of a given policy is important for MARL. For any given joint policy $\boldsymbol{\pi}$, the value function $V^{\boldsymbol{\pi}}\colon \mathcal{S}\to\mathbb{R}$, is defined as

$$V^{\boldsymbol{\pi}}(\mathbf{s}) := \mathbb{E}\big[\textstyle\sum_{p=1}^{\infty}\gamma^{p-1} R_c^{\boldsymbol{\pi}}(\mathbf{s}_p)|\mathbf{s}_1=\mathbf{s},\boldsymbol{\pi}\big]. \tag{3.4}$$

Let us construct the vector $\mathbf{V}^{\boldsymbol{\pi}}\in\mathbb{R}^{|\mathcal{S}|}$ by stacking up $V^{\boldsymbol{\pi}}(\mathbf{s})$ in (3.4) for all $\mathbf{s}$. Then, $\mathbf{V}^{\boldsymbol{\pi}}$ satisfies the Bellman equation

$$\mathbf{V}^{\boldsymbol{\pi}} = \mathbf{R}_c^{\boldsymbol{\pi}} + \gamma\mathbf{P}^{\boldsymbol{\pi}}\mathbf{V}^{\boldsymbol{\pi}}, \tag{3.5}$$

where $\mathbf{R}_c^{\boldsymbol{\pi}}$ is formed by stacking up $R_c^{\boldsymbol{\pi}}(\mathbf{s})$ and $\mathbf{P}^{\boldsymbol{\pi}}$ is defined in (3.3). Furthermore, $\mathbf{V}^{\boldsymbol{\pi}}$ can be shown as the unique solution of (3.5) (see [80]).

One approach to scaling up when the state space size $|\mathcal{S}|$ is large is to approximate $V^{\boldsymbol{\pi}}(\mathbf{s})$ using the family of linear functions $\big\{V_{\boldsymbol{\theta}}(\mathbf{s}) := \boldsymbol{\phi}^{\top}(\mathbf{s})\boldsymbol{\theta} : \boldsymbol{\theta}\in\mathbb{R}^d\big\}$, where $\boldsymbol{\theta}\in\mathbb{R}^d$ is the parameter, $\boldsymbol{\phi}(\mathbf{s}) : \mathcal{S}\to\mathbb{R}^d$ is a feature map consisting of d features, e.g., a dictionary induced by tile coding [102]. We define $\boldsymbol{\Phi} := \big(\cdots;\boldsymbol{\phi}^{\top}(\mathbf{s});\cdots\big)\in\mathbb{R}^{|\mathcal{S}|\times d}$ and let $\mathbf{V}_{\boldsymbol{\theta}}\in\mathbb{R}^{|\mathcal{S}|}$ be formed by stacking up $\big\{V_{\boldsymbol{\theta}}(\mathbf{s})\big\}_{\mathbf{s}\in\mathcal{S}}$. Our aim is to find $\boldsymbol{\theta}\in\mathbb{R}^d$ such that $\mathbf{V}_{\boldsymbol{\theta}}\approx\mathbf{V}^{\boldsymbol{\pi}}$, which specifically minimizes the mean squared projected Bellman error

(MSPBE) defined as

$$\text{MSPBE}^*(\boldsymbol{\theta}) := \tfrac{1}{2}\big\|\boldsymbol{\Pi}_{\boldsymbol{\Phi}}\big(\mathbf{V}_{\boldsymbol{\theta}} - \gamma \mathbf{P}^{\boldsymbol{\pi}}\mathbf{V}_{\boldsymbol{\theta}} - \mathbf{R}_c^{\boldsymbol{\pi}}\big)\big\|_{\mathbf{H}}^2 + \tfrac{\rho}{2}\|\boldsymbol{\theta}\|^2, \tag{3.6}$$

where $\mathbf{H} = \text{diag}[\{\mu^{\boldsymbol{\pi}}(\mathbf{s})\}_{\mathbf{s}\in\mathcal{S}}] \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$ is a diagonal matrix whose diagonal elements are the stationary distribution of $\boldsymbol{\pi}$, $\boldsymbol{\Pi}_{\boldsymbol{\Phi}} : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$ is the projection onto subspace $\{\boldsymbol{\Phi}\boldsymbol{\theta} : \boldsymbol{\theta} \in \mathbb{R}^d\}$, and $\rho \geq 0$ is a regularization parameter. When $\boldsymbol{\Phi}^\top\mathbf{H}\boldsymbol{\Phi}$ is invertible, (3.6) can be further reformed as

$$\text{MSPBE}^*(\boldsymbol{\theta}) = \tfrac{1}{2}\|\mathbf{A}\boldsymbol{\theta} - \boldsymbol{b}\|_{\mathbf{D}^{-1}}^2 + \tfrac{\rho}{2}\|\boldsymbol{\theta}\|^2, \tag{3.7}$$

where $\mathbf{A} := \mathbb{E}\big[\boldsymbol{\phi}(\mathbf{s}_p)\big(\boldsymbol{\phi}(\mathbf{s}_p) - \gamma\boldsymbol{\phi}(\mathbf{s}_{p+1})\big)^\top\big]$, $\mathbf{D} := \mathbb{E}\big[\boldsymbol{\phi}(\mathbf{s}_p)\boldsymbol{\phi}^\top(\mathbf{s}_p)\big]$, and $\boldsymbol{b} := \mathbb{E}\big[\mathcal{R}_c^{\boldsymbol{\pi}}(\mathbf{s}_p)\boldsymbol{\phi}(\mathbf{s}_p)\big]$. Here the expectations are taken with respect to the stationary distribution $\mu^{\boldsymbol{\pi}}$. Moreover, it can be shown that (3.7) has a unique minimizer when $\mathbf{A}$ is full rank and $\mathbf{D}$ is positive definite.

In practice, quantities in these expectations are often unknown, and we only have access to a finite dataset with M transitions $\{\mathbf{s}_p, \mathbf{a}_p\}_{p=1}^M$ simulated from the multi-agent MDP using joint policy $\boldsymbol{\pi}$. The next state $\mathbf{s}_{M+1}$ of $\mathbf{s}_M$ are also observed. Then the empirical versions of $\mathbf{A}$, $\mathbf{D}$, $\boldsymbol{b}$, denoted respectively by $\widehat{\mathbf{A}}$, $\widehat{\mathbf{D}}$, $\widehat{\boldsymbol{b}}$, are defined as

$$\begin{aligned}
&\widehat{\mathbf{A}} := \frac{1}{M}\sum_{p=1}^M \mathbf{A}_p,\ \widehat{\mathbf{D}} := \frac{1}{M}\sum_{p=1}^M \mathbf{D}_p,\ \widehat{\boldsymbol{b}} := \frac{1}{M}\sum_{p=1}^M \boldsymbol{b}_p, \text{with} \\
&\mathbf{A}_p := \boldsymbol{\phi}(\mathbf{s}_p)\big(\boldsymbol{\phi}(\mathbf{s}_p) - \gamma\boldsymbol{\phi}(\mathbf{s}_{p+1})\big)^\top,\ \mathbf{D}_p := \boldsymbol{\phi}(\mathbf{s}_p)\boldsymbol{\phi}^\top(\mathbf{s}_p), \\
&\boldsymbol{b}_p := r_c(\mathbf{s}_p, \mathbf{a}_p)\boldsymbol{\phi}(\mathbf{s}_p),
\end{aligned} \tag{3.8}$$

where $r_c(\mathbf{s}_p, \mathbf{a}_p) := N^{-1}\sum_{i=1}^N r_i(\mathbf{s}_p, \mathbf{a}_p)$. Here we assume that M is sufficiently large such that $\widehat{\mathbf{D}}$ is invertible and $\widehat{\mathbf{A}}$ is full rank. With the terms defined in (3.8), the empirical MSPBE is given by

$$\text{EM-MSPBE}(\boldsymbol{\theta}) := \tfrac{1}{2}\|\widehat{\mathbf{A}}\boldsymbol{\theta} - \widehat{\boldsymbol{b}}\|_{\widehat{\mathbf{D}}^{-1}}^2 + \tfrac{\rho}{2}\|\boldsymbol{\theta}\|^2. \tag{3.9}$$

Let $\widehat{\boldsymbol{\theta}}$ be a minimizer of the empirical MSPBE; our estimation of $\mathbf{V}^{\boldsymbol{\pi}}$ is given by $\boldsymbol{\Phi}\widehat{\boldsymbol{\theta}}$.

Primal-dual Formulation of EM-MSPBE For any $i \in \{1, \cdots, N\}$ and any

$p \in \{1, \cdots, M\}$, we first define that $\boldsymbol{b}_{p,i} := r_i(\mathbf{s}_p, \mathbf{a}_p)\boldsymbol{\phi}(\mathbf{s}_p)$ and $\widehat{\boldsymbol{b}}_i := M^{-1}\sum_{p=1}^{M} \boldsymbol{b}_{p,i}$. Recall that under the multi-agent MDP, agent i is able to compute $\widehat{\mathbf{A}}$ and $\widehat{\mathbf{D}}$ defined in (3.8), but only have access to its own $\widehat{\boldsymbol{b}}_i$ instead of $\widehat{\boldsymbol{b}}$. To address this, we first notice that minimizing (3.9) is equivalent to

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^d} \frac{1}{N}\sum_{i=1}^{N} \text{EM-MSPBE}_i(\boldsymbol{\theta}) \tag{3.10}$$

where $\text{EM-MSPBE}_i(\boldsymbol{\theta}) := \frac{1}{2}\|\widehat{\mathbf{A}}\boldsymbol{\theta} - \widehat{\boldsymbol{b}}_i\|^2_{\widehat{\mathbf{D}}^{-1}} + \frac{\rho}{2}\|\boldsymbol{\theta}\|^2$.

Observe that the summations in the definition of $\widehat{\mathbf{A}}$ and $\widehat{\boldsymbol{b}}$ are still within the norm operation. Thus (3.10) does not fall into the class of multi-agent optimization problems in [43, 44] whose objective function is a summation of local functions with a common parameter. To solve this problem, as inspired by [97,105], we transform $\text{EM-MSPBE}_i(\boldsymbol{\theta})$ to its conjugate form using Fenchel duality:

$$\frac{1}{2}\|\widehat{\mathbf{A}}\boldsymbol{\theta} - \widehat{\boldsymbol{b}}_i\|^2_{\widehat{\mathbf{D}}^{-1}} + \frac{\rho}{2}\|\boldsymbol{\theta}\|^2 = \max_{\mathbf{w}_i\in\mathbb{R}^d} \big(\mathbf{w}_i^\top(\widehat{\mathbf{A}}\boldsymbol{\theta} - \widehat{\boldsymbol{b}}_i) - \frac{1}{2}\mathbf{w}_i^\top\widehat{\mathbf{D}}\mathbf{w}_i\big) + \frac{\rho}{2}\|\boldsymbol{\theta}\|^2. \tag{3.11}$$

By (3.11), problem (3.10) is equivalent to a multi-agent, finite-sum, and primal-dual problem:

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^d}\max_{\mathbf{w}_i\in\mathbb{R}^d, i=1,\cdots,N} \frac{1}{NM}\sum_{i=1}^{N}\sum_{p=1}^{M}\big(\mathbf{w}_i\mathbf{A}_p\boldsymbol{\theta} - \boldsymbol{b}_{p,i}^\top\mathbf{w}_i - \frac{1}{2}\mathbf{w}_i^\top\mathbf{D}_p\mathbf{w}_i + \frac{\rho}{2}\|\boldsymbol{\theta}\|^2\big). \tag{3.12}$$

Define $J_{i,p}(\boldsymbol{\theta}, \mathbf{w}_i) := \big(\mathbf{w}_i\mathbf{A}_p\boldsymbol{\theta} - \boldsymbol{b}_{p,i}^\top\mathbf{w}_i - \frac{1}{2}\mathbf{w}_i^\top\mathbf{D}_p\mathbf{w}_i + \frac{\rho}{2}\|\boldsymbol{\theta}\|^2\big)$, then the global objective function is denoted by $\mathbf{J}(\boldsymbol{\theta}, \{\mathbf{w}_i\}_{i=1}^N) := 1/(NM)\sum_{i=1}^{N}\sum_{p=1}^{M} J_{i,p}(\boldsymbol{\theta}, \mathbf{w}_i)$, which is convex *w.r.t.* the primal variable $\boldsymbol{\theta}$ and is concave *w.r.t.* the dual variable $\{\mathbf{w}_i\}_{i=1}^N$.

Although fully decentralized algorithms have been recently proposed to solve such convex-concave saddle-point problems, the use of non-doubly stochastic mixing matrices and their corresponding communication and computation efficiency have not been considered yet. Here, we study this problem by proposing a general hierarchical decentralized first-order algorithm that endows more flexibility of mixing matrices and schemes. With this more general framework, the proposed method is proved to converge to an optimal solution of (3.12) in a linear rate. Moreover, experiment results

show that various mixing matrices in the proposed algorithm can achieve better convergence than the one in [80] does, while still having less communication and computation cost.

3.3 Algorithm and Analysis

In this section, we will first explain the development of our algorithm and then give the theoretical analysis of its convergence property.

3.3.1 Algorithm Development

We first assume that the N agents transmit information over a network specified by a connected digraph $G = (V, E)$, where $V = [N] := \{1, \cdots, N\}$ and $E \subseteq V \times V$ are the vertex set and directed edge set, respectively. Over G, it is easy to define two row stochastic matrices $\mathbf{R}_1$, $\mathbf{R}_2$ (and two column stochastic matrices $\mathbf{C}_1$, $\mathbf{C}_2$) such that $(\mathbf{R}_1)_{ij} = (\mathbf{R}_2)_{ij} = 0$ $\big((\mathbf{C}_1)_{ij} = (\mathbf{C}_2)_{ij} = 0$ respectively$\big)$ if $(j, i) \notin E$. In addition, we assume $\mathbf{R}_1$ and $\mathbf{R}_2$ share a same left eigenvector. Note the choices of such matrices are abundant, e.g., let $\mathbf{R}_1 = \mathbf{R}_2$ be row stochastic and $\mathbf{C}_1 = \mathbf{C}_2 = \mathbf{R}_1^\top$. It is also worth noting that the differences of this communication model from the one in PD-DistIAG [80] are three-fold. First it is a more general framework which includes the latter as a special case. Assume we have a doubly stochastic matrices $\mathbf{W}$ in [80] over undirected graph G, then (as we will see) by letting $\mathbf{R}_1 = \mathbf{W}, \mathbf{R}_2 = \mathbf{I}, \mathbf{C}_1 = \mathbf{W}$, and $\mathbf{C}_2 = \mathbf{I}$, the proposed algorithm model will reduce to PD-DistIAG. Second, it is much easier and more flexible to construct such matrices over a given graph G compared with the doubly stochastic ones in PD-DistIAG. Therefore we have the freedom to choose good mixing matrices from them that achieve better convergence and have less communication and computation cost. Third, the proposed model also works on directed graphs to which PD-DistIAG may not be applicable. In this setting, it is possible that $(\mathbf{R}_1)_{i,j} \neq 0$ but $(\mathbf{R}_1)_{j,i} = 0$, for $(j, i) \in E$. This means that for estimating parameters in the proposed algorithm, agent i receives information from agent j without transmitting its own information to agent j. As the experiments will show later, such cutting can save the communication cost in each iteration with comparable convergence rate.

Algorithm 2: Primal Dual Hierarchical Gradient Method (PD-H) for Multi-agent Policy Evaluation

Input: Initial variables $\{\boldsymbol{\theta}_i^1, \mathbf{w}_i^1\}_{i\in[N]}$, initial gradient surrogates $\mathbf{s}_i^0 = \mathbf{d}_i^0 = \mathbf{0}$, $\forall i \in [N]$, step sizes $\gamma_1, \gamma_2 > 0$, and initial counter $\tau_p^0 = 0$, $\forall p \in [M]$. Set proper row stochastic matrices $(\mathbf{R}_1, \mathbf{R}_2)$ and column stochastic matrices $(\mathbf{C}_1, \mathbf{C}_2)$ (that satisfy Assumption 3.3.3 and Assumption 3.3.4 in the following).

for $k \geq 1$ **do**

The agents pick a common sample with index $p_k \in \{1, \cdots, M\}$.

Update the counter variables by

$$\tau_p^k = \begin{cases} k, & p = p_k \\ \tau_p^{k-1}, & p \neq p_k. \end{cases} \tag{3.13}$$

for $i \in \{1, 2, \cdots, N\}$ **do**

Agent i updates the gradient surrogates as:

$$\mathbf{s}_i^k = \textstyle\sum_{j=1}^N (\mathbf{C}_1)_{ij} \mathbf{s}_j^{k-1} + \frac{1}{M} \sum_{j=1}^N (\mathbf{C}_2)_{ij}$$
$$\cdot \left[\nabla_{\boldsymbol{\theta}} J_{j,p_k}(\boldsymbol{\theta}_j^k, \mathbf{w}_j^k) - \nabla_{\boldsymbol{\theta}} J_{j,p_k}(\, \boldsymbol{\theta}_j^{\tau_{p_k}^{k-1}}, \mathbf{w}_j^{\tau_{p_k}^{k-1}}\,)\right] \tag{3.14}$$

$$\mathbf{d}_i^k = \mathbf{d}_i^{k-1} + \tfrac{1}{M}\left[\nabla_{\mathbf{w}_i} J_{i,p_k}(\boldsymbol{\theta}_i^k, \mathbf{w}_i^k) - \nabla_{\mathbf{w}_i} J_{i,p_k}(\boldsymbol{\theta}_i^{\tau_{p_k}^{k-1}}, \mathbf{w}_i^{\tau_{p_k}^{k-1}})\right], \tag{3.15}$$

where we define the initial values of gradients $\nabla_{\boldsymbol{\theta}} J_{i,p}(\boldsymbol{\theta}_i^0, \mathbf{w}_i^0) := \mathbf{0}$ and $\nabla_{\mathbf{w}_i} J_{i,p}(\boldsymbol{\theta}_i^0, \mathbf{w}_i^0) := \mathbf{0}$, for $p \in [M]$.

Update the primal and dual variables using the gradients surrogates $\mathbf{s}_i^k$ and $\mathbf{d}_i^k$ w.r.t. $\boldsymbol{\theta}_i$ and $\mathbf{w}_i$:

$$\boldsymbol{\theta}_i^{k+1} = \textstyle\sum_{j=1}^N (\mathbf{R}_1)_{ij} \boldsymbol{\theta}_j^k - \gamma_1 \sum_{j=1}^N (\mathbf{R}_2)_{ij} \mathbf{s}_j^k \tag{3.16}$$

$$\mathbf{w}_i^{k+1} = \mathbf{w}_i^k + \gamma_2 \mathbf{d}_i^k. \tag{3.17}$$

end for

end for

Generally speaking, our method utilizes new mixing matrices and formats to estimate the parameter variable and track the gradient over space (across N agents) and time (across M samples). Before discussing it in detail, we first introduce a centralized and batch algorithm for solving (3.12).

Centralized Primal-dual Gradient Method For any $k \geq 1$, at the k-th iteration, the centralized primal-dual gradient algorithm updates the primal and dual variables

by

$$\begin{aligned}\boldsymbol{\theta}^{k+1} &= \boldsymbol{\theta}^k - \gamma_1 \nabla_{\boldsymbol{\theta}} \mathbf{J}(\boldsymbol{\theta}^k, \{\mathbf{w}_i^k\}_{i=1}^N), \\ \mathbf{w}_i^{k+1} &= \mathbf{w}_i^k + \gamma_2 \nabla_{\mathbf{w}_i} \mathbf{J}(\boldsymbol{\theta}^k, \{\mathbf{w}_i^k\}_{i=1}^N), i \in [N],\end{aligned} \tag{3.18}$$

where $\gamma_1, \gamma_2 > 0$ are step sizes. As shown by [97], when $\widehat{\mathbf{A}}$ is full rank, $\widehat{\mathbf{D}}$ is invertible and $\rho \geq 0$, iterates generated from (3.18) converges linearly to the optimal solution of (3.12) within a certain range of step size (γ_1, γ_2).

Proposed Method The primal-dual gradient method in (3.18) is a prototype solving (3.12) in the single-agent setting. In the MARL model, however, it is challenging to implement this algorithm. Recall in this setting agent i only has access to the functions of its own and the neighbor agents $\{J_{j,p}(\cdot) : (j,i) \in E\}$. Moreover, computing the gradient requires summing up over M samples, which is expensive when $M \gg 1$ as the computation complexity would be $\mathcal{O}(Md)$.

We tackle these issues by combining the gradient tracking idea of [44] with an incremental update scheme from [100] in the following primal-dual hierarchical distributed incremental aggregated gradient (PD-H) method. Here, sequences $\{\mathbf{s}_i^k\}_{k\geq 1}$ and $\{\mathbf{d}_i^k\}_{k\geq 1}$ are introduced to track the gradients *w.r.t.* $\boldsymbol{\theta}_i$ and $\mathbf{w}_i$, respectively. Each agent $i \in [N]$ maintains a local copy of the primal parameter, i.e., $\{\boldsymbol{\theta}_i^k\}_{i\in[N]}$. At the k-th iteration, we update the dual variable via gradient update using $\mathbf{d}_i^k$. However, different from previous works, here each primal variable $\boldsymbol{\theta}_i^{k+1}$ is obtained by first averaging both variables $\{\boldsymbol{\theta}_j^k\}$ and $\{\mathbf{s}_j^k\}$ over its neighbors specified by matrix $\mathbf{R}_1$ and $\mathbf{R}_2$, and then updating the averaged variable with the averaged gradient. Since agents can exchange more information with their neighbors in this approach, the primal variables are able to achieve better consensus. But will this cause more communication cost? The answer is usually no. If we choose $\mathbf{R}_2$ such that its nonzero locations are within $\mathbf{R}_1$'s nonzero locations, e.g. $\mathbf{R}_2 = \mathbf{R}_1$ or $\mathbf{R}_2 = I$, then transmitting the second part of (3.16) causes no extra communication since it can be transmitted in the summation with the first part.

Note that here $\mathbf{s}_i^k$ and $\mathbf{d}_i^k$ represent the surrogate functions for the primal and dual gradients; $\underline{\mathbf{s}}^k := [\mathbf{s}_1^k, \cdots, \mathbf{s}_N^k]$ and $\underline{\mathbf{d}}^k := [\mathbf{d}_1^k, \cdots, \mathbf{d}_N^k]$ are the matrices with $\mathbf{s}_i^k$ and $\mathbf{d}_i^k$ being their i-th column respectively. We also define $\underline{\boldsymbol{\theta}}^k := [\boldsymbol{\theta}_1^k, \cdots, \boldsymbol{\theta}_N^k]$; $\underline{\mathbf{w}}^k := [\mathbf{w}_1^k, \cdots, \mathbf{w}_N^k]$. Moreover, the counter variable is defined as $\tau_p^k = \max\{l \geq 0 : l \leq k, p_l =$

$p\}$, which represents the last iteration when the p-th sample is visited before iteration k, and we set $\tau_p^k = 0$ if the p-th sample has never been visited. The details of our method are presented in Algorithm 2. Like PD-DistIAG, the per-iteration complexity of Algorithm 2 is $\mathcal{O}(d^2)$ which is independent of M throughout the iterative process. Furthermore, Algorithm 2 only requires that each agent stores the features $\phi(\mathbf{s}_p)$ of size $\mathcal{O}(d)$ instead of storing the entire matrices $\mathbf{A}_p$, $\mathbf{D}_p$ of size $\mathcal{O}(d^2)$.

3.3.2 Convergence Analysis

Now we analyze the convergence of the proposed approach. We begin by defining the following concept that is needed in the analysis.

Definition 3.3.1. *A spanning tree of a directed graph (digraph) is a directed tree that connects the root to all other vertices in the graph (see [106]).*

For any nonnegative matrix $\mathbf{Q} \in \mathbb{R}^{n\times n}$, we let $G_{\mathbf{Q}} := (V_{\mathbf{Q}}, E_{\mathbf{Q}})$ denote the directed graph induced by the matrix $\mathbf{Q}$, where $V_{\mathbf{Q}} = \{1, 2, ..., n\}$ and $(j, i) \in E_{\mathbf{Q}}$ iff $\mathbf{Q}_{ij} > 0$. We define $R_{\mathbf{Q}}$ as the set of roots of directed spanning trees in the graph $G_{\mathbf{Q}}$.

Moreover, we introduce important conditions that are used commonly in previous work. The first is to ensure that each sample is picked up frequently in the algorithm.

Assumption 3.3.1. *It holds that $|k - \tau_p^k| \leq M$, $\forall\ p \in [M]$, $k \geq 1$, i.e., each sample is selected at least once per M iterations.*

Remark 3.3.1. *The requirement $|k - \tau_p^k| \leq M$ can be relaxed to $|k - \tau_p^k| \leq C \cdot M$ for some constant $C \geq 1$. This assumption is easier to be satisfied but will not change the linear convergence proved latter.*

The following assumptions are important to establish the linear convergence rate.

Assumption 3.3.2. *The empirical correlation matrix $\widehat{\mathbf{A}}$ defined in* (3.8) *is full rank, and $\widehat{\mathbf{D}}$ defined in* (3.8) *is non-singular.*

In addition, the proof of convergence also relies on assumptions about the mixing matrices.

Assumption 3.3.3. *$\mathbf{R}_1, \mathbf{R}_2 \in \mathbb{R}^{d\times d}$ are nonnegative and row stochastic with a shared left eigenvector $\mathbf{u}$ of eigenvalue 1 and $\mathbf{C}_1, \mathbf{C}_2 \in \mathbb{R}^{d\times d}$ are nonnegative and column stochastic. In addition, $(\mathbf{R}_1)_{ii} > 0$ and $(\mathbf{C}_1)_{ii} > 0$ for all $i \in V$.*

Remark 3.3.2. *Given $\mathbf{R}_1$, $\mathbf{R}_2$ can be set as $\lambda\mathbf{I} + (1-\lambda)\mathbf{R}_1$, for $\lambda \in [0,1]$ such that Assumption 3.3.3 holds. Moreover, here $(\mathbf{R}_1)_{ii} > 0$ and $(\mathbf{C}_1)_{ii} > 0$ are necessary to ensure that $\sigma_{\mathbf{R}_1} < 1$ and $\sigma_{\mathbf{C}_1} < 1$, where $\sigma_{\mathbf{R}_1}$ and $\sigma_{\mathbf{C}_1}$ are the spectral radii of $(\mathbf{R}_1 - \boldsymbol{1}\mathbf{u}^\top/N)$ and $(\mathbf{C}_1 - \mathbf{v}\boldsymbol{1}^\top/N)$, respectively. Note though not mentioned explicitly, [80] also requires the diagonal entries of the doubly stochastic mixing matrix $\mathbf{W}$ are positive to ensure $\|\mathbf{W} - N^{-1}\boldsymbol{1}\boldsymbol{1}^\top\|_{1,\infty} < 1$.*

Remark 3.3.3. *Here the assumption about the non-doubly stochastic property of the mixing matrices is a main difference between our paper and [80]. Namely, the approach in [80] cannot be applied here to establish the relation of the error metrics in adjacent iterations as in (3.22). As we will discuss later, to exploit the weaker assumption we design new matrices in (5.54), (5.55), (5.58), and (5.59), prove different results regarding the weighted consensus errors of $\underline{\mathbf{s}}$ and $\underline{\boldsymbol{\theta}}$ with Lemma 2 and Lemma 3, and conduct different spectral analyses to bound the spectrum of matrix Q(γ) (in the sequel).*

Assumption 3.3.4. *Each of the graphs $G_{\mathbf{R}_1}$ and $G_{\mathbf{C}_1^\top}$ contains at least one spanning tree. Further, $R_{\mathbf{R}_1} \cap R_{\mathbf{C}_1^\top} \neq \emptyset$.*

We know Assumption 3.3.4 essentially ensures sufficient connections given by $\mathbf{R}_1$ and $\mathbf{C}_1$. By Assumption 3.3.3, we directly have the following result.

Lemma 3.3.1. *Under Assumption 3.3.3, the matrix $\mathbf{C}_1$ has a nonnegative right eigenvector $\mathbf{v}$ with eigenvalue 1 satisfying $\boldsymbol{1}^\top\mathbf{v} = N$ (see [107]).*

Let $(\boldsymbol{\theta}^*, \{\mathbf{w}_i^*\}_{i=1}^N)$ be the optimal solution of (3.12). Based on these assumptions, now we can present the main theorem.

Theorem 3.3.1. *Suppose that Assumptions 3.3.1-3.3.4 hold. Set the step sizes as $\gamma_2 = \beta\gamma, \gamma_1 = \gamma$ with $\beta := 8r'(\rho + \lambda_{\max}(\widehat{\mathbf{A}}^\top\widehat{\mathbf{D}}^{-1}\widehat{\mathbf{A}}))/\lambda_{\min}(\widehat{\mathbf{D}})$, where $r' := \mathbf{u}^\top\mathbf{v}/N$ and $\gamma > 0$. Define $\bar{\boldsymbol{\theta}}^k = \frac{1}{N}\underline{\boldsymbol{\theta}}^k\mathbf{u}$ as the weighted average of parameters. When the primal step*

size γ is sufficiently small[1]*, there exists a constant $\sigma \in (0,1)$ such that*

$$\|\bar{\boldsymbol{\theta}}^k - \boldsymbol{\theta}^*\|^2 + (r')^2/(\beta N)\sum_{i=1}^N \|\mathbf{w}_i^k - \mathbf{w}_i^*\|^2 = \mathcal{O}(\sigma^k),$$
$$\tfrac{1}{N}\sum_{i=1}^N \|\boldsymbol{\theta}_i^k - \bar{\boldsymbol{\theta}}^k\| = \mathcal{O}(\sigma^k).$$

If $M, N \gg 1$ and we have $\max\{\sigma_{\mathbf{R}_1}, \sigma_{\mathbf{C}_1}\} = 1 - \kappa/N$ for some $\kappa > 0$, then setting $\gamma = \mathcal{O}(1/\max\{M^2, N^2\})$ yields a convergence rate $\sigma = 1 - \mathcal{O}(1/\max\{MN^2, M^3\})$.

The above theorem shows that the iterates $(\bar{\boldsymbol{\theta}}^k, \{\mathbf{w}_i^k\}_{i=1}^N)$ generated by the proposed algorithm converge to the optimum in a linear rate. Moreover, the consensus error of local primal variables $\frac{1}{N}\sum_{i=1}^N \|\boldsymbol{\theta}_i^k - \bar{\boldsymbol{\theta}}^k\|$ also converges to 0 linearly.

The proof of Theorem 3.3.1 needs the following lemmata.

Lemma 3.3.2 ([46])**.** *Suppose $R_{\mathbf{R}_1} \neq \emptyset$ and $R_{\mathbf{C}_1^\top} \neq \emptyset$. Then under Assumption 3.3.3, it holds that $R_{\mathbf{R}_1} \cap R_{\mathbf{C}_1^\top} \neq \emptyset$ if and only if $\mathbf{u}^\top \mathbf{v} \neq 0$.*

Lemma 3.3.3. *Under Assumption 3.3.1-3.3.4, we have the following linear inequalities:*

$$\begin{bmatrix} \|\underline{\boldsymbol{\theta}}^{k+1} - \bar{\boldsymbol{\theta}}^{k+1}\mathbf{1}^\top\| \\ \|\underline{\mathbf{s}}^{k+1} - \bar{\mathbf{s}}^{k+1}\mathbf{v}^\top\| \end{bmatrix} \leq \mathbf{Q}_0 \begin{bmatrix} \max_{(k-2M)_+ \leq s \leq k} \|\underline{\boldsymbol{\theta}}^s - \bar{\boldsymbol{\theta}}^s \mathbf{1}^\top\| \\ \max_{(k-M)_+ \leq s \leq k} \|\underline{\mathbf{s}}^s - \bar{\mathbf{s}}^s \mathbf{v}^\top\| \\ \max_{(k-2M)_+ \leq s \leq k} \|\underline{\boldsymbol{v}}^s\| \end{bmatrix}, \tag{3.19}$$

where matrix $\mathbf{Q}_0 = [q_{ij}]$ is defined as

$$\begin{bmatrix} q_{11} \\ q_{21} \end{bmatrix} = \begin{bmatrix} \sigma_{\mathbf{R}_1} + \gamma\sigma_{\mathbf{R}_2}\|\mathbf{v} - \mathbf{1}\|\frac{\rho}{\sqrt{N}} \\ \sigma_{\mathbf{C}_2}\rho\|\mathbf{R}_1 - \mathbf{I}\|_S + \gamma(\sigma_{\mathbf{C}_2}\rho^2\|\mathbf{R}_2\mathbf{v}\|\frac{1}{\sqrt{N}} + \sigma_{\mathbf{C}_2}\bar{A}^2\beta) \end{bmatrix},$$

$$\begin{bmatrix} q_{12} \\ q_{22} \end{bmatrix} = \begin{bmatrix} \gamma\sigma_{\mathbf{R}_2} \\ \sigma_{\mathbf{C}_1} + \sigma_{\mathbf{C}_2}\rho\gamma\|\mathbf{R}_2\|_S \end{bmatrix},$$

$$\begin{bmatrix} q_{13} \\ q_{23} \end{bmatrix} = \begin{bmatrix} \sqrt{2}\gamma\sigma_{\mathbf{R}_2}\|\mathbf{v} - \mathbf{1}\|L\max(1, \frac{\sqrt{\beta}}{r'}) \\ \sqrt{2}\big(\sigma_{\mathbf{C}_2}L^2\beta\gamma\sqrt{N} + \sigma_{\mathbf{C}_2}L^2\gamma\|\mathbf{R}_2\mathbf{v}\|\big)\max\{1, \frac{\sqrt{\beta}}{r'}\} \end{bmatrix}. \tag{3.20}$$

[1]such that the right hand side of (5.78) is less than 1.

Here for $s \geq 0$, we define

$$\underline{\boldsymbol{v}}^s := \begin{bmatrix} \bar{\boldsymbol{\theta}}^s - \boldsymbol{\theta}^* \\ \frac{r'}{\sqrt{\beta N}}(\mathbf{w}_1^s - \mathbf{w}_1^*) \\ \vdots \\ \frac{r'}{\sqrt{\beta N}}(\mathbf{w}_N^s - \mathbf{w}_N^*) \end{bmatrix}, \qquad L := \max\{\rho, \bar{A}, \bar{D}\}$$

with $\bar{A}$ and $\bar{D}$ being defined as

$$\bar{A} := \max_{1,\cdots,M} \|\mathbf{A}_p\|_S, \qquad \bar{D} := \max_{1,\cdots,M} \|\mathbf{D}_p\|_S. \tag{3.21}$$

Lemma 3.3.3 gives the progress of the consensus errors of $\underline{\boldsymbol{\theta}}$ and $\underline{\mathbf{s}}$. This will help establish the relation between the optimality gap progress and the consensus error progress in one iteration in the proof of Theorem 3.3.1. By analyzing this inequality system, we can find the sufficient condition that guarantees the linear convergence of the proposed algorithm.

The proofs of the lemmata and the theorem are in the Appendix. Now in the following we briefly prove Theorem 3.3.1.

Proof Sketch To analyze the convergence rate of the proposed algorithm, the first step is to measure the progress of the primal and dual optimality gap characterized by $\|\underline{\boldsymbol{v}}^k\|$ and the consensus error characterized by $\|\underline{\boldsymbol{\theta}}^k - \bar{\boldsymbol{\theta}}^k \mathbf{1}^\top\|$ and $\|\underline{\mathbf{s}}^k - \bar{\mathbf{s}}^k \mathbf{v}^\top\|$. The latter is established in Lemma 3.3.3 and the optimality gap progress is obtained in the detailed proof of Theorem 3.3.1. The challenges in this step are the lack of the desirable doubly stochastic property of the mixing matrices and the imperfect tracking of the gradient over samples and agents. The second step is to analyze the coupled system of optimality gap progress and consensus error progress in one iteration; that is, to analyze the spectral norm of its coefficient matrix. Lastly, we find the sufficient condition of the step size γ such that each term in the coupled system will converge linearly to 0.

Specifically, by exploiting the row and column stochastic properties in Assumption 3.3.3 and the updates of $\underline{\boldsymbol{\theta}}^k$ and $\underline{\mathbf{s}}^k$, one can derive that

$$\underline{\boldsymbol{\theta}}^{k+1} - \bar{\boldsymbol{\theta}}^{k+1}\mathbf{1}^\top = (\underline{\boldsymbol{\theta}}^k - \bar{\boldsymbol{\theta}}^k\mathbf{1}^\top)\big(\mathbf{R}_1 - \frac{\mathbf{1}\mathbf{u}^\top}{N}\big)^\top - \gamma_1(\underline{\mathbf{s}}^k - \bar{\mathbf{s}}^k\mathbf{1}^\top)\big(\mathbf{R}_2 - \frac{\mathbf{1}\mathbf{u}^\top}{N}\big)^\top,$$

and

$$\begin{aligned}\underline{\mathbf{s}}^{k+1} - \bar{\mathbf{s}}^{k+1}\mathbf{v}^\top = \frac{1}{M}\big[\nabla_{\underline{\boldsymbol{\theta}}}\mathbf{J}_{p_{k+1}}(\underline{\boldsymbol{\theta}}^{k+1}, \underline{\mathbf{w}}^{k+1}) - \nabla_{\underline{\boldsymbol{\theta}}}\mathbf{J}_{p_{k+1}}(\underline{\boldsymbol{\theta}}^{\tau^k_{p_{k+1}}}, \underline{\mathbf{w}}^{\tau^k_{p_{k+1}}})\big] \\ \cdot \big(\mathbf{C}_2 - \frac{\mathbf{v}\mathbf{1}^\top}{N}\big)^\top + (\underline{\mathbf{s}}^k - \bar{\mathbf{s}}^k\mathbf{v}^\top)\big(\mathbf{C}_1 - \frac{\mathbf{v}\mathbf{1}^\top}{N}\big)^\top,\end{aligned}$$

where for $p \geq 1$, $\mathbf{J}_p(\underline{\boldsymbol{\theta}}^k, \underline{\mathbf{w}}^k)$ is defined by

$$\mathbf{J}_p(\underline{\boldsymbol{\theta}}^k, \underline{\mathbf{w}}^k) := \big[J_{1,p}(\boldsymbol{\theta}_1^k, \mathbf{w}_1^k), \cdots, J_{N,p}(\boldsymbol{\theta}_N^k, \mathbf{w}_N^k)\big].$$

Combining the above equalities with the optimality of $(\boldsymbol{\theta}^*, \{\mathbf{w}_i^*\}_{i=1}^N)$ and the Lipschitz continuity of $\nabla_{\underline{\boldsymbol{\theta}}}\mathbf{J}_p(\underline{\boldsymbol{\theta}}, \underline{\mathbf{w}})$ yields the inequality (3.19).

To consider the optimality gap, first in the appendix we carefully construct matrices $\underline{\boldsymbol{v}}^k$ in (5.59), $\mathbf{G}$, $\mathbf{G}_p$'s in (5.54), (5.55), and $\underline{\mathbf{h}}(k)$ in (5.58) with the parameter r'. Then the bound of $\|\underline{\boldsymbol{v}}^{k+1}\|$ follows from the norm bounds of the two summands in (5.66), which eventually leads to (5.75). Lastly, let $\underline{\widehat{\boldsymbol{v}}}^k$ denotes a vector whose norm satisfies $\|\underline{\widehat{\boldsymbol{v}}}^k\| = \Theta(\|\underline{\boldsymbol{v}}^k\|)$. We show that by combining the above results it holds that

$$\begin{bmatrix} \|\underline{\boldsymbol{\theta}}^{k+1} - \bar{\boldsymbol{\theta}}^{k+1}\mathbf{1}^\top\| \\ \|\underline{\mathbf{s}}^{k+1} - \bar{\mathbf{s}}^{k+1}\mathbf{v}^\top\| \\ \|\underline{\widehat{\boldsymbol{v}}}^{k+1}\| \end{bmatrix} \leq Q(\gamma) \begin{bmatrix} \max_{(k-2M)_+ \leq s \leq k} \|\underline{\boldsymbol{\theta}}^s - \bar{\boldsymbol{\theta}}^s\mathbf{1}^\top\| \\ \max_{(k-M)_+ \leq s \leq k} \|\underline{\mathbf{s}}^s - \bar{\mathbf{s}}^s\mathbf{v}^\top\| \\ \max_{(k-2M)_+ \leq s \leq k} \|\underline{\widehat{\boldsymbol{v}}}^s\| \end{bmatrix}, \tag{3.22}$$

where matrix $Q(\gamma)$ is defined by

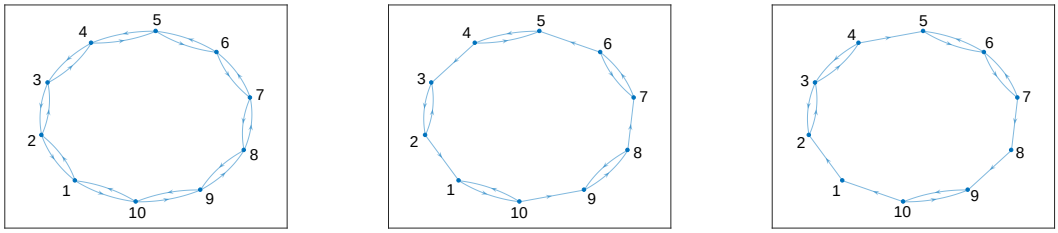
$$Q(\gamma) := \begin{bmatrix} \sigma_{\mathbf{R}_1} + \gamma\sigma_{\mathbf{R}_2}\|\mathbf{v} - \mathbf{1}\|\frac{\rho}{\sqrt{N}} & \gamma\sigma_{\mathbf{R}_2} & C_6(\gamma) \\ C_7(\gamma) & C_4(\gamma) & C_5(\gamma) \\ \frac{1}{\sqrt{N}}C_2(\gamma) & C_3(\gamma) & C_8(\gamma) \end{bmatrix},$$

and the $C_i(\gamma)$'s are defined in (5.76) and (5.77).

Based on this inequality system, we derive an upper bound for the spectral norm of the coefficient matrix $Q(\gamma)$. As a result, if there exists a step size $\gamma > 0$ such that

the spectral norm of $Q(\gamma)$ is less than 1, then it can be shown that the optimality gap and the consensus error converge linearly to 0. Moreover, the analysis in the proof of Theorem 3.3.1 proves the existence and gives the set of such step sizes. Therefore the linear convergence is established when the step size is chosen in this range. This concludes the proof sketch of Theorem 3.3.1.

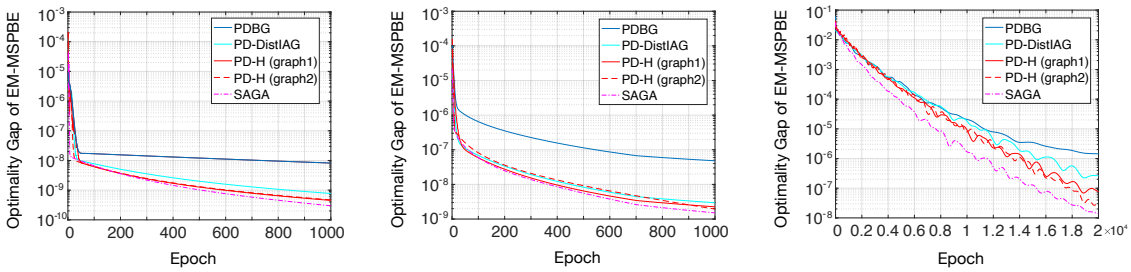
3.4 Experiments



(a) (b) (c)

Figure 3.1: This figure shows the communication graphs in the algorithms (Ring graph), which correspond to (a) PD-DistIAG, (b) PD-H graph1 (reduced by **25%**), and (c) PD-H graph2 (reduced by **25%**). The number of edges in the graphs is in proportion to the communication cost over agents in each iteration. Therefore the communication cost per iteration is reduced by 25% in the right two panels.

Now we test our algorithm on the policy evaluation problems for the random Markov Decision Process (MDP) task with $d = 400$ features and $M = 2000$ samples as well as the Mountain Car task [102]. To collect the dataset of the Mountain Car task, we first obtain a good policy by running Sarsa [102] with $d = 300$ features. Then we run this policy to generate trajectories of $M = 5000$ samples. Given a sample p, each agent is assigned randomly with local reward such that the average of the local rewards equals $r_c(\mathbf{s}_p, \mathbf{a}_p)$. We consider both cases when $\rho = 0.1$, $\rho = 0.01$, and $\rho = 0$ over different connection graphs: the ring, grid, and Erdos-Renyi (ER) graphs. In these settings, we compare various advanced algorithms: (1) PDBG: the centralized method described in (3.18); (2) SAGA: the centralized method proposed in [97]; (3) PD-DistIAG: the decentralized method based on undirected communication graphs in [80]; (4) PD-H: the



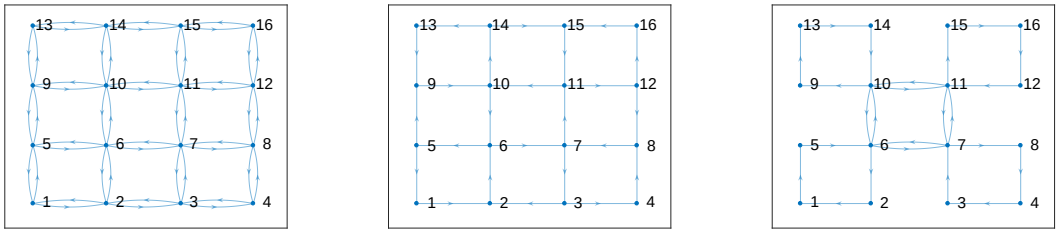
(a) $\rho = 0.1$ (b) $\rho = 0.01$ (c) $\rho = 0$

Figure 3.2: Convergence Comparison on random MDP (Ring graph).

proposed method in the chapter.

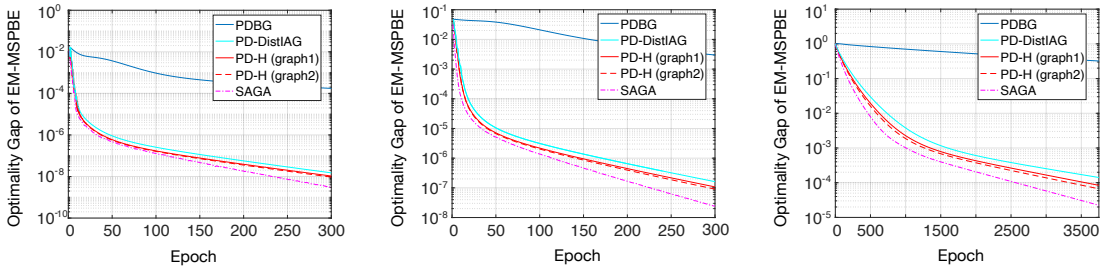
For the decentralized methods, we simulate networks of $N = 10$ agents with the ring graph, $N = 16$ agents with the grid graph, and $N = 100$ agents with the ER graph. The ring graph is adopted on the random MDP task; the grid graph and the ER graph are adopted on the Mountain Car task. We first compare these algorithms' communication cost in each iteration, that is, the total number of transmissions between the nodes in each round of update. Then their optimality gaps vs. epoch number are examined. Here we choose the step sizes that give the best empirical performance of the competing algorithms. Figures 3.1, 3.3, and 3.5 show the communication graphs corresponding to the mixing matrices $\mathbf{W}$ in the PD-DistIAG and $\mathbf{R}_1$ in the proposed method for different topologies. Note that each edge in the graphs represents the communication between the connected nodes in each iteration. We set $\mathbf{R}_2 = \mathbf{C}_2 = I$, $\mathbf{C}_1 = \mathbf{R}_1^\top$ in the first and second networks and $\mathbf{R}_2 = \mathbf{R}_1$, $\mathbf{C}_1 = \mathbf{C}_2 = \mathbf{R}_1^\top$ in the third one. Note the left undirected communication graphs of PD-DistIAG in Figures 3.1, 3.3, and 3.5 can be regarded as special digraphs with each undirected edge being equivalent to two in-and-out directed edges. Given the graph in Figure 3.1a, the right graphs in Figure 3.1b and Figure 3.1c are formed by cutting its directed edges alternatively in one direction. In contrast, if the left graph is considered as an undirected graph as in PD-DistIAG, then cutting any two of its undirected edges would result in disconnectedness that prevents PD-DistIAG from converging. A similar advantage of the proposed algorithm also occurs in graphs with a grid topology. Specifically, we observe that the number of

directed edges in Figure 3.3b and Figure 3.3c is half of that in Figure 3.3a. However, it would also cause its disconnectedness if we cut half of the undirected edges of the grid graph in Figure 3.3a. In the ER graph case, we generate the graph in Figure 3.5a with connection probability $1.1\log(N)/N$. Based on this graph, the right digraphs in Figure 3.5b and Figure 3.5c are formed by randomly cutting 21% of its directed edges. Note that to achieve the same communication deduction, PD-DistIAG needs an *undirected* ER subgraph with a connection probability less than $0.98\log(N)/N$. However, this will again lead to disconnectedness with high probability that prevents its convergence [108].



(a) (b) (c)

Figure 3.3: This figure shows the communication graphs in the algorithms (Grids graph), which correspond to (a) PD-DistIAG, (b) PD-H graph1 (reduced by **50%**), and (c) PD-H graph2 (reduced by **50%**). The communication cost per iteration is reduced by 50% in the right two panels as compared with panel (a).

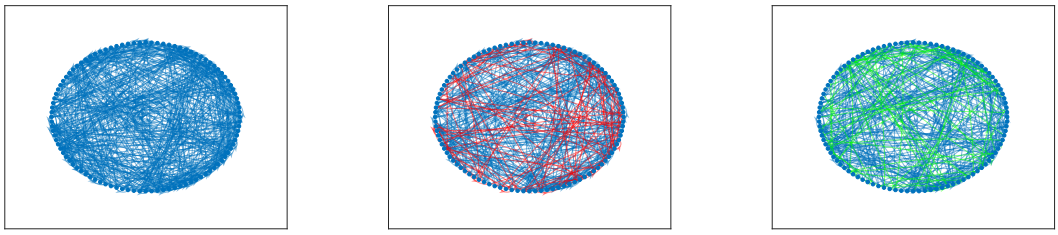


(a) $\rho = 0.1$ (b) $\rho = 0.01$ (c) $\rho = 0$

Figure 3.4: Convergence Comparison on Mountain Car (Grids graph).

Figures 3.2, 3.4, and 3.6 compare the optimality gaps of the objective function MSPBE versus the epoch number, which is defined as t/M. Recall the number of

links in the graphs is in proportion to the communication cost and the number of summations over agents in each iteration. Therefore it can be observed from Figure 3.1, Figure 3.3, and Figure 3.5 that in these settings the proposed approach saves more than 20% communication per iteration. Moreover, Figure 3.2, Figure 3.4, and Figure 3.6 show that when $\rho > 0$ the convergence of PD-H is close to that of the centralized method SAGA for both graph1 and graph2. When $\rho = 0$, though slower than the centralized method SAGA, the advantages of PD-H over other methods are more obvious as more epochs are needed to reach the minimum due to the adversarial conditional number. It can be observed that PD-H consistently converges faster than PD-DistIAG, while requiring less communication in each iteration. Moreover, such results can also be observed in other settings of parameters as shown in the following Table 3.1.

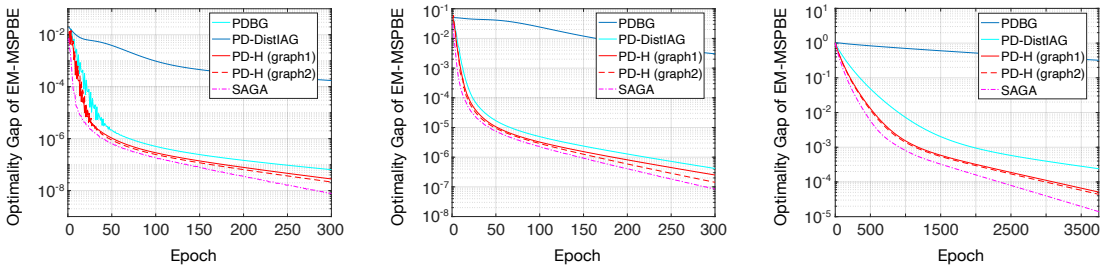


(a) (b) (c)

Figure 3.5: This figure shows the communication graphs in the algorithms (ER graph), which correspond to (a) PD-DistIAG, (b) PD-H graph1 (reduced by **21%**), and (c) PD-H graph2 (reduced by **21%**). The differences between the graphs' edges in (b) and (c) are red colored and green colored respectively. As compared with the first, the communication cost per iteration is reduced by 21% in the second and third panels.

3.5 Conclusions

This chapter proposes a communication-efficient primal dual hierarchical distributed gradient algorithm (PD-H) solving the multi-agent policy evaluation problem (3.12). Theoretically, we prove the proposed algorithm converges deterministically to the optimum in a linear rate under reasonable assumptions of the empirical correlation matrices and communication graphs. Moreover, unlike in the previous work, the proposed algorithm no longer requires doubly stochastic mixing matrices generated from undirected graphs. The freedom of choosing broader classes of mixing matrices and schemes enables



(a) $\rho = 0.1$ (b) $\rho = 0.01$ (c) $\rho = 0$

Figure 3.6: Convergence Comparison on Mountain Car (ER Graph).

Table 3.1: Evaluation in various settings of the parameters

(N,Epoch,ρ)	Graph	Task	Communication	Optimality Gap
(10, 300, 0.01)	Star	Mountain Car	reduced by **17%**	improved by 7.3e-09
(10, 1000, 0.01)	Star	MDP	reduced by **17%**	improved by 5.3e-08
(10, 300, 0.1)	Star	Mountain Car	reduced by **17%**	improved by 1.2e-09
(10, 1000, 0.1)	Star	MDP	reduced by **17%**	improved by 9.8e-09
(10, 5000, 0)	ER	Mountain Car	reduced by **21%**	improved by 6.6e-05
(10, 50000, 0)	ER	MDP	reduced by **21%**	improved by 3.1e-07
(500, 300, 0.01)	Ring	Mountain Car	reduced by **25%**	improved by 3.9e-09
(100, 2000, 0.01)	Ring	MDP	reduced by **25%**	improved by 2.6e-08
(500, 5000, 0)	ER	Mountain Car	reduced by **21%**	improved by 5.6e-05
(500, 50000, 0)	ER	MDP	reduced by **21%**	improved by 2.1e-06
(500, 300, 0.1)	Ring	Mountain Car	reduced by **25%**	improved by 2.8e-10
(100, 2000, 0.1)	Ring	MDP	reduced by **25%**	improved by 3.6e-09
(64, 2000, 0)	Grids	Mountain Car	reduced by **50%**	improved by 1.8e-06
(64, 5000, 0)	Grids	MDP	reduced by **50%**	improved by 8.4e-06
(64, 300, 0.1)	Grids	Mountain Car	reduced by **50%**	improved by 5.2e-08
(64, 800, 0.1)	Grids	MDP	reduced by **50%**	improved by 3.8e-07

the algorithm to be applicable on digraphs with hierarchical communication structures. In experiments, we compare PD-H with other state-of-the-art algorithms on multi-agent policy evaluation applications, and the results show the superior performance of the proposed algorithm in terms of convergence and communication efficiency.

Acknowledgements

This work was partially supported by the DARPA Young Faculty Award N66001-14-1-4047 and the Beijing Institute of Technology Research Fund Program for Young Scholars.

Chapter 4

Discussions of other works and future areas

This chapter discusses some other topics we have been working on including communication efficient algorithms with the use of sparsification, dictionary-based robust outlier pursuit in statistical signal processing, and asynchronous distributed ADMM algorithm with multiple masters as well as their possible refinements and combinations with the results presented above.

The work in Section 4.1 studies a sparsity-aware communication-efficient distributed algorithm [26][1] inspired by the important benefits of sparsity in solving signal processing and statistics problems. By exploring the sparsity of the transmitted variables, we propose a novel distributed algorithm with vector compression based on an update similar to the one in Chapter 2. It is shown in the analysis that the proposed algorithm achieves the convergence rate at the same order of the original one while the communication costs at each iteration is in proportion to the sparsity (the number of the nonzero elements) in the iterates, therefore reducing the overall communication overheads.

Section 4.2 is a component of our paper [27] in which the model of robust PCA with dictionary-based outlier decomposition and column-wise sparsity was first considered and analyzed in the context of solving a convex demixing optimization problem. The original paper appeared in the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2018) [109] and the contents in Section 4.2 come from the

[1] ©[2020] IEEE. Reprinted, with permission, from [J. Ren, X. Li, and J. Haupt, Communication-Efficient Distributed Optimization for Sparse Learning via Two-Way Truncation, IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), December 2017]

journal version in the IEEE Transactions on Signal Processing [27][2].

Section 4.3 considers an extension of the hybrid decentralized ADMM proposed in [24] which can comprise multiple masters and accelerate the convergence by assigning cliques of worker machines according to the network connection. With a similar advantageous structure, we further propose an asynchronous hybrid decentralized ADMM that works for more general cost functions with a nonsmooth regularizer. The convergence guarantee of the proposed algorithm to the optimum point and its competitive performance compared with other alternatives are also presented in the more general scenario. The original work appeared in the IEEE Global Conference on Signal and Information Processing (GlobalSIP 2018) [25][3].

Notice that here we treat each section as an essentially stand-alone entity and there could be some notational differences across sections.

[2] ©[2020] IEEE. Reprinted, with permission, from [S. Rambhatla, X. Li, J. Ren, and J. Haupt, A Dictionary-Based Generalization of Robust PCA With Applications to Target Localization in Hyperspectral Imaging, IEEE Transactions on Signal Processing, March 2020]

[3] ©[2020] IEEE. Reprinted, with permission, from [M. Ma, J. Ren, G. B. Giannakis, and J. Haupt, Fast Asynchronous Decentralized Optimization: Allowing Multiple Masters, IEEE Global Conference on Signal and Information Processing (GlobalSIP), November 2018]

4.1 Sparsity-Aware Communication-Efficient Algorithms

The study of sparsity is of significant importance in the statistical signal processing communities. In general, sparsity describes the scenario where a large dataset is essentially representable by only a small number of coefficients based on its underlining structure. Thus in sparsity-aware distributed algorithms it is not hard to image the use of such succinctness in saving the cost during communications. For example, many works including [110] propose sparsified Stochastic Gradient Descent (SGD) that compresses the gradients at each iteration and prove that the algorithm converges at the same rate as vanilla SGD. As a result, its communication is reduced by a factor depending on the degrees of the compression.

Moreover, the idea of reusing historical information of gradients to achieve a better estimation of the gradient at iterates and to save the communication rounds is discussed by works like [44, 111] in the decentralized gradient descent background. Besides, [112, 113] present a class of gradient-based algorithms (called LAG and LAQ) that does not need to compute and transmit the gradients at each iterations. Instead, this class of algorithms can adaptively skip the gradient calculations when the gradients at recent iterations are slowly-varing. They prove that under certain heterogeneous condition, their method achieves a targeted accuracy with reduced communication rounds thanks to the adaptive reuse of lagged gradients.

Here we propose an efficient algorithm for distributed learning that leverages the sparsity of exchanging parameters and the information in previous iterations for saving communications. At each iteration, local machines compute local gradients on their own local data and using these, a master machine solves a shifted ℓ_1 regularized minimization problem. Here, our contribution reduces the communication cost per transmission from the order of the parameter dimension to the order of the number of nonzero entries in the parameter via a Two-Way Truncation procedure. Theoretically, we prove that the estimation error of the proposed algorithm decreases exponentially and matches that of the centralized method under certain conditions. Numerical experiments on both simulated data and real data support that the proposed algorithm is efficient and has statistical performance comparable with the centralized method.

4.1.1 Introduction

We consider the following minimization of expected loss,

$$\min_{\theta} \mathbb{E}_{\mathbf{X},Y\sim\mathcal{D}} \left[l(Y, \langle \mathbf{X}, \theta \rangle) \right], \tag{4.1}$$

where $l(\cdot,\cdot)$ is a loss function and $(\mathbf{X}, Y) \in \mathcal{X} \times \mathcal{Y} \subseteq \mathbb{R}^d \times \mathcal{Y}$ has a distribution $\mathcal{D}$. In practice, the minimizer θ^* of (4.1), also known as the true model parameter for a given observation model, is estimated from N observations $\{\mathbf{x}_i, y_i\}_{i=1}^N$.

For high-dimensional learning problem, where d is large compared with the number of observations, the effective variables are often supported on a small subset $S \subseteq [d]$, i.e., $S := \text{support}\{\theta^*\} = \{i \in [d] : \theta_i^* \neq 0\}$ with sparsity $s := |S| \ll d$. Extensive efforts have been made to develop batch algorithms [114–116], which provide good convergence guarantees in optimization. However, when N is large, batch algorithms are inefficient, taking at least $\mathcal{O}(N)$ time per iteration. Therefore, an emerging recent interest is observed to address this problem using the distributed optimization frameworks [32,33, 117], which is more efficient than the stochastic algorithms. One important limitation of existing distributed optimization for sparse learning is that they did not take advantage of the sparse structure, thus they have the same communication costs with general dense problems. In this section, we propose a novel communication-efficient distributed algorithm to explicitly leverage sparse structure for solving large scale sparse learning problems. This allows us to reduce the communication cost from $\mathcal{O}(d)$ in existing works to $\mathcal{O}(s)$, while still maintaining nearly the same performance under certain assumptions.

Notation For two sequences of numbers $\{a_n\}$ and $\{b_n\}$, we say $b_n = \mathcal{O}(a_n)$ if there exists a constant $C > 0$ and an integer M such that $b_n \leq C \cdot a_n$ for all $n \geq M$. We say $a_n \lesssim b_n$ if $a_n = \mathcal{O}(b_n)$ and $a_n \gtrsim b_n$ if $b_n = \mathcal{O}(a_n)$. The notation $a_n \asymp b_n$ denotes that $a_n = \mathcal{O}(b_n)$ and $b_n = \mathcal{O}(a_n)$. For a vector $\mathbf{v} \in \mathbb{R}^d$, the l_p-norm of $\mathbf{v}$ is defined as $\|\mathbf{v}\|_p = (\sum_{i=1}^d |\mathbf{v}_i|^p)^{1/p}$, where $p > 0$; the l_0-norm of $\mathbf{v}$ is defined as the number of its nonzero entries; the support of $\mathbf{v}$ is defined as $\text{supp}(\mathbf{v}) = \{i : \mathbf{v}_i \neq 0\}$. We use $[d]$ to denote the set $\{1, \cdots, d\}$. For a matrix $A = (a_{ij}) \in \mathbb{R}^{n_1 \times n_2}$, we define the l_∞-norm of A as $\|A\|_\infty = \max_{i\in[n_1], j\in[n_2]} |a_{ij}|$. Given a number $k \leq d$, the hard thresholding $\mathcal{H}_k(\mathbf{v})$ of a vector $\mathbf{v} \in \mathbb{R}^d$ is defined by keeping the largest k entries of $\mathbf{v}$ (in magnitude) and setting the rest to be zero. Given a subset S of index set $\{1, \cdots, d\}$, the projection

$\mathcal{P}_S(\mathbf{v})$ of a vector $\mathbf{v}$ on S is defined by

$$\mathcal{P}_S(\mathbf{v})_j = 0, \quad \text{if } j \notin S \quad \text{and} \quad \mathcal{P}_S(\mathbf{v})_j = \mathbf{v}_j, \quad \text{if } j \in S.$$

$\mathcal{P}_S(\mathbf{v})$ is also denoted as $(\mathbf{v})_S$ for convenience.

much previous work

Related work

There is a large body of work on distributed optimizations such as [58,63,64,66,117–119]. Initially, most distributed algorithms used averaging estimators formed by local machines [63, 66]. More recently, [58, 70, 120] propose more communication-efficient distributed optimization algorithms. Moreover, using ideas of the approximate Newton-type method, [32, 33] further improved the computational efficiency of this type of method.

Many gradient hard thresholding approaches are proposed in recent years such as [121, 122]. They showed that under suitable conditions, the hard thresholding type first-order algorithms attain linear convergence to a solution which has optimal estimation accuracy with high probability. However, to the best of our knowledge, hard thresholding techniques applied to approximate Newton-type distributed algorithms has not been considered yet. In this section, we present some initial theoretical and experimental results on this topic.

4.1.2 Algorithm

In this section, we introduce our approach to estimating the θ^* that minimizes the expected loss. Without loss of generality, suppose that the total observation satisfies $N = nm$ and the observations of j-th local machine are $\{\mathbf{x}_{ji}, y_{ji}\}_{i=1}^n$ for all $j \in [m] := \{1, \ldots, m\}$.

Assume we have m machines: one master machine and $m-1$ local machines. The master machine is connected to other $m-1$ local machines. During each iteration, the master machine solves a subproblem to obtain an updated parameter, communicates this to the local machines, and the local machines all compute gradients at the updated parameter θ. These gradients are then communicated back to the master machine, and

the process continues.

Formally, the empirical loss at each machine is defined as

$$\mathcal{L}_j(\theta) = \frac{1}{n}\sum_{i=1}^{n} l(y_{ji}, \langle \mathbf{x}_{ji}, \theta\rangle), \quad \text{where} \quad j \in [m].$$

To evaluate the minimizer θ^* of (4.1), we solve a l_1 regularized subproblem to get an initial estimate: the master machine solves

$$\gamma^0 = \operatorname*{argmin}_{\theta} \mathcal{L}_1(\theta) + \mu_0\|\theta\|_1. \tag{4.2}$$

The initial point θ^0 is formed by keeping the largest k elements of the resulting minimizer γ^0 and setting the other elements to be zero, i.e., $\theta^0 = \mathcal{H}_k(\gamma^0)$. Then θ^0 is broadcast to the local machines, where it is used to compute a gradient of local empirical loss at θ^0, that is, $\nabla\mathcal{L}_j(\theta^0)$. The local machines project $\nabla\mathcal{L}_j(\theta^0)$ on the support S^0 of θ^0 and transmit the projection $\mathcal{P}_{S^0}\left[\nabla\mathcal{L}_j(\theta^0)\right]$ back to the master machine. Later at $(h+1)$-th iteration ($h \geq 0$), the master solves a shifted ℓ_1 regularized minimization subproblem:

$$\begin{aligned}\gamma^{h+1} = \operatorname*{argmin}_{\theta} \quad & \mathcal{L}_1(\theta) + \mu_{h+1}\|\theta\|_1 \\ & + \left\langle \mathcal{P}_{S^h}\left[\frac{1}{m}\sum_{j=1}^{m}\nabla\mathcal{L}_j(\theta^h)\right] - \nabla\mathcal{L}_1(\theta^h), \theta\right\rangle.\end{aligned} \tag{4.3}$$

Again the minimizer γ^{h+1} is truncated to form θ^{h+1}, and this quantity is communicated to the local machines, where it is used to compute the local gradient as before.

Solving subproblem (4.3) is inspired by the approach of Wang et al. [32] and Jordan et al. [33], and is designed to take advantage of both global first-order information and local higher-order information. Indeed, when $\mu_{h+1} = 0$ and $\mathcal{L}_j$ is quadratic, (4.3) has the following closed form solution:

$$\gamma^{h+1} = \theta^h - \nabla^2\mathcal{L}_1(\theta^h)^{-1}\left(\mathcal{P}_{S^h}\left[\frac{1}{m}\sum_{j=1}^{m}\nabla\mathcal{L}_j(\theta^h)\right]\right),$$

which is similar to a Newton updating step. The more general case has a *proximal Newton* flavor; see, e.g., [71] and the references therein. Note that here we add a projection procedure $\mathcal{P}_{S^h}\left[\frac{1}{m}\sum_{j=1}^{m}\nabla\mathcal{L}_j(\theta^h)\right]$ to reduce the number of nonzeros that need to be communicated to the master machine. This procedure is reasonable intuitively.

First, when θ^h is close to θ^*, the elements of $\frac{1}{m}\sum_{j=1}^{m}\nabla\mathcal{L}_j(\theta^h)$ outside the support S^h should be very small, so nominally little error is incurred in the truncation step. Second, when θ^{h+1} is also close to θ^*, the lost part has even more minimal effects on the inner product in subproblem (4.3). Third, we leave $-\nabla\mathcal{L}_1(\theta^h)$ in (4.3) out of the truncation to maintain the formulation as unbiased. The detailed algorithm is summarized in Algorithm 3.

Algorithm 3: Two-Way Truncation Distributed Sparse Learning

Input: Loss function $l(\cdot,\cdot)$, data $\{\mathbf{x}_{ji}, y_{ji}\}_{i\in[n],j\in[m]}$.
Local machines:
Initializaiton: The master solves the local ℓ_1 regularized loss minimization problem (4.2) to get a solution γ^0. Set $\theta^0 = \mathcal{H}_k(\gamma^0)$.
for $h = 0, 1, \ldots$ **do**
  for $j = 2, 3, \ldots, m$ **do**
    if Receive θ^h from the master **then**
    Calculate gradient $\nabla\mathcal{L}_j(\theta^h)$ and get the projection $\mathcal{P}_{S^h}\left[\nabla\mathcal{L}_j(\theta^h)\right]$ of the gradient on support S^h and transmit it to the master.
    end
  end for
  Master:
  if Receive $\{\nabla\mathcal{L}_j(\theta^h)\}_{j=2}^m$ from local machines **then**
    Solve the shifted ℓ_1 regularized problem (4.3) to obtain γ^{h+1}.
    Do hard thresholding $\theta^{h+1} = \mathcal{H}_k(\gamma^{h+1})$.
    Let $S^{h+1} = \mathrm{supp}(\theta^{h+1})$.
    Broadcast θ^{h+1} to every local machine.
  end
end for

4.1.3 Theoretical Guarantees

Computational Analysis

First, we start with introducing important conditions that are keys to our analysis.

Assumption 4.1.1. *The loss $l(\cdot,\cdot)$ is a L-smooth function of the second argument, i.e.,*

$$\left|\frac{\partial l(x,y)}{\partial y}\Big|_{y=y_1} - \frac{\partial l(x,y)}{\partial y}\Big|_{y=y_2}\right| \leq L|y_1 - y_2|, \quad \forall x, y_1, y_2 \in \mathbb{R}.$$

Moreover, the third derivative with respect to its second argument is bounded by a constant M, i.e.,

$$\left|\frac{\partial^3 l(x,y)}{\partial y^3}\right| \leq M, \quad \forall x, y \in \mathbb{R}$$

Assumption 4.1.2 (Restricted Strong Convexity)**.** *The empirical loss function computed on the first machine satisfies that: $\forall \Delta \in \mathcal{C}(S,3)$, we have*

$$\mathcal{L}_1(\theta^* + \Delta) - \mathcal{L}_1(\theta^*) - \langle \nabla \mathcal{L}_1(\theta^*), \Delta \rangle \geq \kappa \|\Delta\|_2^2,$$

where $\mathcal{C}(S,3)$ is defined as

$$\mathcal{C}(S,3) = \{\Delta \in \mathbb{R}^d | \ \|\Delta_{S^c}\|_1 \leq 3\|\Delta_S\|_1\}.$$

Assumption 4.1.3. *The γ^{h+1}, S^{h+1} and S^h defined in Algorithm 3 satisfy the following condition: there exists some positive constants H and τ_1 and τ_2 such that for $h \geq H$,*

$$\left\| \left(\gamma^h - \theta^*\right)_{(S^h)^c} \right\|_1 \leq \tau_1 \left\| \gamma^h - \theta^* \right\|_1 \tag{4.4}$$

$$\left\| \left(\gamma^{h+1} - \theta^*\right)_{S^{h+1} \setminus S^h} \right\|_1 \leq \tau_2 \left\| \gamma^{h+1} - \theta^* \right\|_1 . \tag{4.5}$$

Remark 4.1.1. *The last assumption is a somewhat stringent, per-iteration condition. In practice, though, we note that both τ_1 and τ_2 are very small even after only one round of communication and will decrease to 0 quickly in later steps.*

For simplicity, we define the following notation:

$$\begin{aligned}
\overline{\mathcal{L}_1}(\theta^*, \theta^h) &:= \mathcal{L}_1(\theta^*) + \left\langle \tfrac{1}{m} \textstyle\sum_{j=1}^m \nabla \mathcal{L}_j(\theta^h) - \nabla \mathcal{L}_1(\theta^h), \theta \right\rangle, \\
\widetilde{\mathcal{L}_1}(\theta^*, \theta^h) &:= \mathcal{L}_1(\theta^*) \\
&\quad + \left\langle \mathcal{P}_{S^h}\left[\tfrac{1}{m} \textstyle\sum_{j=1}^m \nabla \mathcal{L}_j(\theta^h) - \nabla \mathcal{L}_1(\theta^h) \right], \theta \right\rangle.
\end{aligned}$$

Now we are ready state our main result for the computational guarantees.

Theorem 4.1.1. *Suppose that Assumption 4.1.1, 4.1.2, and 4.1.3 hold. Let $\rho = \tau_1 + \tau_2$,*

$k = C_1 \cdot s$ with $C_1 > 1$ and

$$\begin{aligned}\mu_{h+1} = \quad & 4\left\|\tfrac{1}{m}\textstyle\sum_{j=1}^{m}\nabla\mathcal{L}_j(\theta^*)\right\|_\infty \\ & + 2L\left(\max_{j,i}\|\mathbf{x}_{ji}\|_\infty^2\right)\cdot\left[2\sqrt{\tfrac{\log(2d/\delta)}{n}}+\rho\right]\|\theta^h-\theta^*\|_1 \\ & + 2M\left(\max_{j,i}\|\mathbf{x}_{ji}\|_\infty^3\right)\|\theta^h-\theta^*\|_1^2, \end{aligned} \tag{4.6}$$

Then with probability at least $1-\delta$, we have

$$\begin{aligned}\|\theta^{h+1}-\theta^*\|_1 \leq & \tfrac{C_2 s}{\kappa}\left\|\tfrac{1}{m}\textstyle\sum_{j=1}^{m}\nabla\mathcal{L}_j(\theta^*)\right\|_\infty \\ & + \tfrac{C_2 s}{2\kappa}L\cdot\max_{j,i}\|\mathbf{x}_{ji}\|_\infty^2\cdot\left[2\sqrt{\tfrac{\log(2d/\delta)}{n}}+\rho\right]\|\theta^h-\theta^*\|_1 \\ & + \tfrac{C_2 s}{2\kappa}M\cdot\max_{j,i}\|\mathbf{x}_{ji}\|_\infty^3\cdot\|\theta^h-\theta^*\|_1^2,\end{aligned}$$

where $C_2 = 24\sqrt{1+2(C_1-1)^{-\frac{1}{2}}}\cdot\sqrt{C_1+1}$ is a positive constant independent of m, n, s, d.

The theorem immediately implies the following convergence result.

Corollary 4.1.1. *Let $\rho = \tau_1 + \tau_2$. Suppose that for all h*

$$\begin{aligned}& M\cdot\left(\max_{j,i}\|\mathbf{x}_{ji}\|_\infty^3\right)\|\theta^h-\theta^*\|_1 \leq \\ & L\cdot\max_{j,i}\|\mathbf{x}_{ji}\|_\infty^2\left[2\sqrt{\tfrac{\log(2d/\delta)}{n}}+\rho\right].\end{aligned} \tag{4.7}$$

Then under the assumption of Theorem 4.1.1 we have

$$\begin{aligned}\|\theta^h-\theta^*\|_1 \leq & \tfrac{1-a_n^h}{1-a_n}\cdot\tfrac{C_2 s}{\kappa}\cdot\left\|\tfrac{1}{m}\textstyle\sum_{j=1}^{m}\nabla\mathcal{L}_j(\theta^*)\right\|_\infty \\ & + a_n^{h+1}\|\theta^0-\theta^*\|_1,\end{aligned}$$

where C_2 is defined in Theorem 4.1.1,

$$a_n = \tfrac{C_2 s}{\kappa}L\cdot\max_{j,i}\|\mathbf{x}_{ji}\|_\infty^2\cdot\left[2\sqrt{\tfrac{\log(2d/\delta)}{n}}+\rho\right].$$

Remark 4.1.2. *From the theory, we see that the hard thresholding parameter k can be chosen as $C_1 \cdot s$, where C_1 can be a moderate constant larger than 1. By contrast, previous work, such as [122] that solves a nonconvex minimization problem subject to*

l_0 norm constraint $\|\theta\|_0 \leq k$, requires that $k \asymp \kappa_s^2 s$ to have good computational convergence guarantees, where κ_s is the condition number of the objective function. Moreover, instead of only hard thresholding on the solution of l_1 regularized subproblems, we also perform projection on the gradients in (4.3)*. These help us reduce the communication cost from $\mathcal{O}(d)$ to $\mathcal{O}(s)$ per iteration, which is a significant improvement of communication efficiency when $s \ll d$.*

Statistical Analysis

We present the statistical guarantees of our proposed algorithm on two popular statistical models for illustration.

Sparse Linear Regression

In the sparse linear regression, data $\{\mathbf{x}_{ji}, y_{ji}\}_{i\in[n],j\in[m]}$ are generated according to the model

$$y_{ji} = \langle \mathbf{x}_{ji}, \theta^* \rangle + \epsilon_{ji}, \tag{4.8}$$

where the ϵ_{ji} represent noises. One representative scenario occurs when ϵ_{ji} are i.i.d. Gaussian random variables with zero mean and variance σ, and $\{\mathbf{x}_{ji}\}$ are drawn from a mean zero random Gaussian distribution with variance $\sigma_{\mathbf{x}}$. We consider the squared loss function $l(y_{ji}, \langle \theta, \mathbf{x}_{ji} \rangle) = \frac{1}{2}(y_{ji} - \langle \theta, \mathbf{x}_{ji} \rangle)^2$, which is 1-smooth.

Combining Corollary 4.1.1 with some intermediate results obtained from [123, 124] and [125], we have the following bound for the estimation error.

Corollary 4.1.2. *Suppose the design matrix and noise are as above, Assumption 4.1.3 holds and μ_h is defined as* (4.6)*. Then under the sparse linear model, we have the following estimation error bounds with probability at least $1 - 2\delta$:*

$$\begin{aligned} \|\theta^h - \theta^*\|_1 \leq \tfrac{1-a_n^h}{1-a_n} \cdot \tfrac{C_2 s\sigma\sigma_{\mathbf{x}}}{\kappa} \sqrt{\tfrac{\log(d/\delta)}{mn}} \\ + a_n^{h+1} \tfrac{s\sigma\sigma_{\mathbf{x}}}{\kappa} \sqrt{\tfrac{\log(nd/\delta)}{n}}, \end{aligned}$$

where C_2 is defined in Theorem 4.1.1, and where

$$a_n = \frac{C_2 s}{\kappa}\sigma_{\mathbf{x}}^2 \log\left(\frac{mnd}{\delta}\right)\left[2\sqrt{\frac{\log(2d/\delta)}{n}} + \rho\right].$$

Remark 4.1.3. *We can further simplify the bound and have an insight of the relation between n, m, s, d. When $n \geq s^2 \log d$, by the definition of μ_h in Theorem 4.1.1, we have*

$$\mu_h \asymp \sqrt{\frac{\log d}{mn}} + \sqrt{\frac{\log d}{n}}\left[s\left(\sqrt{\frac{\log d}{n}} + \rho\right)\right]^h$$

and $k = C_1 s$, then we have the following error bounds with high probability:

$$\|\theta^h - \theta^*\|_1 \lesssim s\sqrt{\frac{\log d}{mn}} + s\sqrt{\frac{\log d}{n}}\left[s\left(\sqrt{\frac{\log d}{n}} + \rho\right)\right]^h.$$

Therefore when $s < \left(\sqrt{\frac{\log d}{n}} + \rho\right)^{-1}$, we have θ^h converges linearly to an estimation of θ^* with the same estimation accuracy as the centralized method's.

Sparse Logistic Regression

Logistic regression is a model used for classification problem where the binary label $y_{ji} \in \{-1, 1\}$ is drawn from a Bernoulli distribution such that:

$$\mathbb{P}(y_{ji} = 1|\mathbf{x}_{ji}) = \frac{\exp\left(\langle\theta^*, \mathbf{x}_{ji}\rangle\right)}{\exp\left(\langle\theta^*, \mathbf{x}_{ji}\rangle\right) + 1}, \text{ and}$$
$$\mathbb{P}(y_{ji} = -1|\mathbf{x}_{ji}) = \frac{1}{\exp\left(\langle\theta^*, \mathbf{x}_{ji}\rangle\right) + 1}.$$

Maximizing the log-likelihood leads to the logistic loss function $l(y_{ji}, \langle\theta, \mathbf{x}_{ji}\rangle) = \log\left(1 + \exp\left(-y_{ji}\langle\theta, \mathbf{x}_{ji}\rangle\right)\right)$, which is $\frac{1}{4}$-smooth.

Combining Corollary 4.1.1 with some intermediate results obtained from [32] and [126], we now can give a similar result about the estimation error bound for sparse logistic regression: under suitable conditions, when $n \gtrsim s^2 \log d$, by choosing

$$\mu_h \asymp \sqrt{\frac{\log d}{mn}} + \sqrt{\frac{\log d}{n}}\left[s\left(\sqrt{\frac{\log d}{n}} + \rho\right)\right]^h$$

and $k = C_1 s$, then we have the following error bounds with high probability:

$$\|\theta^h - \theta^*\|_1 \lesssim s\sqrt{\tfrac{\log d}{mn}} + s\sqrt{\tfrac{\log d}{n}}\left[s\left(\sqrt{\tfrac{\log d}{n}} + \rho\right)\right]^h.$$

4.1.4 Future Improvement

As shown in the experiments in [26], the proposed algorithm indeed achieves preferable communication efficiency over other comparing methods. Note that the conditions under which the proposed method converges require that the inequalities in Assumption 4.1.3 hold at each iteration. So a meaningful future improvement of the analysis for this type of algorithms is to remove the strong assumption like Assumption 4.1.3, or replace it with a mild one that is easy to satisfy in the implementation. Moreover, it is also interesting to study whether this algorithm converges in the more general asynchronous nonconvex cases like Algorithm 1 in the previous chapter does. Similarly, the combinations with the hierarchical communication structures presented above are promising to further expand the applications of the proposed algorithm to general networks as well.

4.2 Dictionary Based Robust PCA via Outlier Pursuit

In this section we discuss the decomposition of a data matrix which is a superposition of a low-rank matrix and a component which is sparse in a known dictionary, using a convex demixing method. This scenario can be found in target identification applications in hyperspectral (HS) imaging [109], where the priori knowledge of the target signatures (dictionary) is available for helping localization. Here we consider the column-wise sparsity structures for the sparse factor of the dictionary sparse component, and provide a unified analysis, encompassing both the undercomplete and the overcomplete dictionary cases, to show that the constituent matrices can be successfully recovered under some relatively mild conditions on incoherence, sparsity, and rank. We leverage these results to localize targets of interest in a hyperspectral (HS) image based on their spectral signature(s) using the a priori known characteristic spectral responses of the target. In [27], we also show the performance of our approach via experimental evaluations and comparisons to related techniques.

Specifically, we studied the following model for $\mathbf{M}$:

$$\mathbf{M} = \mathbf{L} + \mathbf{DS} \tag{4.9}$$

and identified the conditions under which components $\mathbf{L}$ and $\mathbf{S}$ can be recovered given obeservation $\mathbf{M}$ and dictionary $\mathbf{D}$ by solving appropriate convex formulations. Without loss of generality, we assume the sizes of matrices $\mathbf{M}$, $\mathbf{L}$, $\mathbf{D}$, and $\mathbf{S}$ to be $n \times m$, $n \times m$, $n \times d$, and $d \times m$ respectively.

First, we considered a case where $\mathbf{S}$ has s_c non-zero columns (column-wise sparse case). For this formulation, we developed the conditions under which solving

$$\min_{\mathbf{L},\mathbf{S}} \|\mathbf{L}\|_* + \lambda_c \|\mathbf{S}\|_{1,2} \text{ s.t. } \mathbf{M} = \mathbf{L} + \mathbf{DS} \quad (\text{ D-RPCA(C) }) \tag{4.10}$$

will recover $\mathbf{L}$ and $\mathbf{S}$ for certain regularization parameter $\lambda_c \geq 0$, given the data $\mathbf{M}$ and the dictionary $\mathbf{D}$. The known dictionary $\mathbf{D}$ here can be overcomplete (fat, i.e., $d > n$) or undercomplete (thin, i.e., $d \leq n$). Here, "D-RPCA" refers to "Dictionary based Robust Principal Component Analysis," while "C" indicates column-wise sparsity pattern. In addition, $\|\cdot\|_*, \|\cdot\|_1$, and $\|\cdot\|_{1,2}$ refer to the nuclear norm, ℓ_1- norm of the vectorized

matrix, and $\ell_{1,2}$ norm (sum of column ℓ_2 norms), respectively, which serve as convex relaxations of rank, sparsity, and column sparsity inducing regularization, respectively.

The model in (4.9) is also closely related to the one in [127], which explores the overcomplete dictionary setting with applications to network traffic anomaly detection. However, the analysis therein applies to a case where the $\mathbf{D}$ is overcomplete with orthogonal rows, and the coefficient matrix $\mathbf{S}$ has a small number of non-zero elements per row and column. We analyze the variation of [127] for column-wise sparsity cases that includes a scenario where the dictionary has more rows than columns, i.e., is thin, while removing the orthogonality constraint for both the thin and the fat dictionary scenarios. In the column-wise setting, model (4.9) is also closely related to outlier identification [128, 129] which is motivated by a number of contemporary data analysis applications. Here, the column-wise sparse matrix (known as "outliers") can be used to identify malicious responses in collaborative filtering applications [130], finding anomalous patterns in network traffic [131], or estimating visually salient regions of images [132–134].

4.2.1 Preliminaries

In the following we introduce the parameters and notions that are related to our theorectical analysis of the proposed methods.

First of all, in the the column-wise sparsity setting, due to the inherent ambiguity in the model (4.9), we can only hope to recover the column-space for the low-rank matrix and the identities of the non-zero columns for the sparse matrix. Therefore, in this case any solution in the Oracle Model (defined below) is deemed to be optimal.

Definition D.1 (Oracle Model for Column-wise Sparsity Case): Let the pair $(\mathbf{L}, \mathbf{S})$ be the matrices forming the data $\mathbf{M}$ as per (4.9) and define the oracle model $\{\mathbf{M}, \mathcal{U}, \mathcal{I}_{\mathcal{S}_c}\}$. Then, any pair $(\mathbf{L}_0, \mathbf{S}_0)$ is in the Oracle Model $\{\mathbf{M}, \mathcal{U}, \mathcal{I}_{\mathcal{S}_c}\}$, if $\mathcal{P}_{\mathcal{U}}(\mathbf{L}_0) = \mathbf{L}$ $\mathcal{P}_{\mathcal{S}_c}(\mathbf{DS}_0) = \mathbf{DS}$ and $\mathbf{L}_0 + \mathbf{DS}_0 = \mathbf{L} + \mathbf{DS} = \mathbf{M}$ hold simultaneously, where $\mathcal{P}_{\mathcal{U}}$ and $\mathcal{P}_{\mathcal{S}_c}$ are projections onto the column space $\mathcal{U}$ of $\mathbf{L}$ and column support $\mathcal{I}_{\mathcal{S}_c}$ of $\mathbf{S}$, respectively.

Moreover, we require that the dictionary $\mathbf{D}$ follows the generalized frame property (GFP) defined as follows.

Definition D.2: A matrix $\mathbf{D}$ satisfies the generalized frame property (GFP), on

vectors $\mathbf{v} \in \mathcal{R}$, if for any fixed vector $\mathbf{v} \in \mathcal{R}$ and some vector space $\mathcal{R}$, we have

$$\alpha_\ell \|\mathbf{v}\|_2^2 \leq \|\mathbf{D}\mathbf{v}\|_2^2 \leq \alpha_u \|\mathbf{v}\|_2^2,$$

where α_ℓ and α_u are the lower and upper frame bounds with $0 < \alpha_\ell \leq \alpha_u < \infty$.

The GFP shown above is met as long as the vectors $\mathbf{v}$ are not in the null-space of the matrix $\mathbf{D}$ for finite $\|\mathbf{D}\|$. Therefore, for the thin dictionary setting $d \leq n$, $\mathcal{R}$ can be the entire space, and GFP is satisfied as long as $\mathbf{D}$ has full column rank. For example, $\mathbf{D}$ being a frame suffices; see also [135]. On the other hand, for the fat dictionary setting, we need the space $\mathcal{R}$ to have a union-of-subspace structure such that GFP is met for the column-wise sparsity case.

Further, let $(\mathbf{L}_0, \mathbf{S}_0)$ denote a solution pair in the oracle model $\{\mathbf{M}, \mathcal{U}, \mathcal{I}_{\mathcal{S}_c}\}$ in **D.1**, obtained by solving D-RPCA(C). For the low-rank matrix $\mathbf{L}$, let the compact singular value decomposition (SVD) be defined as

$$\mathbf{L} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top,$$

where $\mathbf{U} \in \mathbb{R}^{n \times r}$ and $\mathbf{V} \in \mathbb{R}^{m \times r}$ are the left and right singular vectors of $\mathbf{L}$, respectively, and $\mathbf{\Sigma}$ is the diagonal matrix with singular values on the diagonal. Here, matrices $\mathbf{U}$ and $\mathbf{V}$ each have orthogonal columns, and the non-negative entries $\mathbf{\Sigma}_{ii} = \sigma_i$ are arranged in descending order. We define $\mathcal{L}$ as the linear subspace consisting of matrices spanning the same row or column space as $\mathbf{L}$, i.e., for $\mathbf{W}_1 \neq 0$ or $\mathbf{W}_2 \neq 0$

$$\mathcal{L} := \left\{ \mathbf{U}\mathbf{W}_1^\top + \mathbf{W}_2\mathbf{V}^\top, \mathbf{W}_1 \in \mathbb{R}^{m \times r}, \mathbf{W}_2 \in \mathbb{R}^{n \times r} \right\}.$$

Next, let $\mathcal{S}_c$ be the space spanned by $d \times m$ matrices with the same non-zero column support (denoted as csupp$(\cdot)$) as $\mathbf{S}$, and let $\mathcal{I}_{\mathcal{S}_c}$ denote the index set containing the non-zero column index set of $\mathbf{S}$, then we denote the space spanned by the dictionary sparse component $\mathcal{D}$ as $\mathcal{D} := \{\mathbf{D}\mathbf{H}\}$, where csupp$(\mathbf{H}) \subseteq \mathcal{I}_{\mathcal{S}_c}$. Also, we denote the corresponding complements of the spaces described above by appending '$\perp$'. In addition, we use calligraphic '$\mathcal{P}_{\mathcal{G}}(\cdot)$ ' to denote the projection operator onto a subspace $\mathcal{G}$, and $\mathbf{P}_{\mathbf{G}}$ to denote the corresponding projection matrix. For instance, we define $\mathcal{P}_{\mathcal{U}}(\cdot)$ and $\mathcal{P}_{\mathcal{V}}(\cdot)$ as the projection operators corresponding to the column space $\mathcal{U}$ and row space

$\mathcal{V}$ of the low-rank component $\mathbf{L}$. Therefore, for a given matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$

$$\mathcal{P}_{\mathcal{U}}(\mathbf{X}) = \mathbf{P_U X} \text{ and } \mathcal{P}_{\mathcal{V}}(\mathbf{X}) = \mathbf{X P_V},$$

where $\mathbf{P_U} = \mathbf{U U}^\top$ and $\mathbf{P_V} = \mathbf{V V}^\top$. With this, the projection operators onto, and orthogonal to, the subspace $\mathcal{L}$ are respectively defined as

$$\mathcal{P}_{\mathcal{L}}(\mathbf{X}) = \mathbf{P_U X} + \mathbf{X P_V} - \mathbf{P_U X P_V},$$

and

$$\mathcal{P}_{\mathcal{L}^\perp}(\mathbf{X}) = (\mathbf{I} - \mathbf{P_U})\, \mathbf{X}\, (\mathbf{I} - \mathbf{P_V})\,.$$

Next, we employ various notions of incoherence to identify the conditions under which our procedures succeed. To this end, we first define the incoherence parameter μ, which characterizes the relationship between the low-rank part $\mathbf{L}$ and the dictionary sparse part $\mathbf{DS}$ as

$$\mu := \max_{\mathbf{Z} \in \mathcal{D} \setminus \{0\}} \frac{\|\mathcal{P}_{\mathcal{L}}(\mathbf{Z})\|_{\mathbf{F}}}{\|\mathbf{Z}\|_{\mathbf{F}}}. \tag{4.11}$$

The parameter $\mu \in [0, 1]$ is the measure of degree of similarity between the low-rank part and the dictionary sparse component. Here, a larger μ implies that the dictionary sparse component is close to the low-rank part, while a small μ indicates otherwise. In addition, we also define the parameter $\beta_{\mathbf{U}}$ as

$$\beta_{\mathbf{U}} := \max_{\|\mathbf{u}\|=1} \frac{\|(\mathbf{I} - \mathbf{P_U})\, \mathbf{D u}\|^2}{\|\mathbf{D u}\|^2}, \tag{4.12}$$

which measures the similarity between the orthogonal complement of the column-space $\mathcal{U}$ and the dictionary $\mathbf{D}$.

The next two measures of incoherence can be interpreted as a ways to identify the cases where for $\mathbf{L}$ with SVD as $\mathbf{L} = \mathbf{U \Sigma V}^\top$: (a) $\mathbf{U}$ resembles the dictionary $\mathbf{D}$, and / or (b) $\mathbf{V}$ resembles the sparse coefficient matrix $\mathbf{S}$. In these cases, the low-rank part may mimic the dictionary sparse component. To this end, similar to [127], we define

the following to measure these properties respectively as

$$\text{(a) } \gamma_{\mathbf{U}} := \max_i \frac{\|\mathbf{P_U D e}_i\|^2}{\|\mathbf{D e}_i\|^2} \text{ and (b) } \gamma_{\mathbf{V}} := \max_i \|\mathbf{P_V e}_i\|^2 . \tag{4.13}$$

Here, $\gamma_{\mathbf{U}} \in [0,1]$, and achieves the upper bound when a dictionary element is exactly aligned with the column space $\mathcal{U}$ of $\mathbf{L}$. Moreover, $\gamma_{\mathbf{V}} \in [r/m, 1]$ achieves the upper bound when the row-space of $\mathbf{L}$ is "spiky," i.e., a certain row of V is 1 -sparse, meaning that a column of $\mathbf{L}$ is supported by (can be expressed as) a linear combination of a column of $\mathbf{U}$. The lower bound here is attained when it is "spread-out," i.e., each column of $\mathbf{L}$ is a linear combination of all columns of $\mathbf{U}$. In general, our recovery of the two components is easier when the incoherence parameters $\gamma_{\mathbf{U}}$ and $\gamma_{\mathbf{V}}$ are closer to their lower bounds. Further, for notational convenience, we define

$$\xi_c := \left\| \mathbf{D}^\top \mathbf{U} \mathbf{V}^\top \right\|_{\infty,2} . \tag{4.14}$$

Here, ξ_c measures the closeness of columns of $\mathbf{D}$ to the singular vectors of $\mathbf{L}$ under a column-wise maximum ℓ_2-norm metric.

4.2.2 Theoretical Results

Recall that we consider the oracle model in this case as described in **D.1** owing to the intrinsic ambiguity in recovery of $(\mathbf{L}, \mathbf{S})$. To demonstrate its recoverability, the following lemma establishes the sufficient conditions for the existence of an optimal pair $(\mathbf{L}_0, \mathbf{S}_0)$.

Lemma 4.2.1. *Given* $\mathbf{M}, \mathbf{D}$, *and* $(\mathcal{L}, \mathcal{S}_c, \mathcal{D})$, *any pair* $(\mathbf{L}_0, \mathbf{S}_0) \in \{\mathbf{M}, \mathcal{U}, \mathcal{I}_{\mathcal{S}_c}\}$ *satisfies* $\text{span}\{\text{col}(\mathbf{L}_0)\} = \mathcal{U}$ *and* $\text{csupp}(\mathbf{S}_0) = \mathcal{I}_{\mathcal{S}_c}$ *if* $\mu < 1$.

In the following, we show the existence of a non-empty interval $\left[\lambda_c^{\min}, \lambda_c^{\max}\right]$ for the regularization parameter λ_c, for which solving D-RPCA(C) recovers an optimal pair as per Lemma 4.2.1. Here, for a constant $C_c := \frac{\alpha_u}{\alpha_\ell} \frac{1}{(1-\mu)^2} \gamma_{\mathbf{V}} \beta_{\mathbf{U}}$, $\lambda_c^{\min}$ and $\lambda_c^{\max}$ are defined as

$$\lambda_c^{\min} := \frac{\xi_c + \sqrt{r s_c \alpha_u} \mu C_c}{1 - s_c C_c} \text{ and } \lambda_c^{\max} := \frac{\sqrt{\alpha_\ell}(1-\mu) - \sqrt{r\alpha_u}\mu}{\sqrt{s_c}}. \tag{4.15}$$

Then, our main result for the column-wise case is as follows.

Theorem 4.2.1. *Suppose* $\mathbf{M} = \mathbf{L}+\mathbf{DS}$ *with* $(\mathbf{L}, \mathbf{S})$ *defining the oracle model* $\{\mathbf{M}, \mathcal{U}, \mathcal{I}_{\mathcal{S}_c}\}$, *where* $\text{rank}(\mathbf{L}) = r$, *and* $|\mathcal{I}_{\mathcal{S}_c}| = s_c$ *for* $s_c \leq s_c^{\max} := \frac{\alpha_\ell}{\alpha_u \gamma_{\mathbf{V}}} \cdot \frac{(1-\mu)^2}{\beta_{\mathbf{U}}}$. *Given* $\mu \in [0, 1), \beta_{\mathbf{U}}$ $\gamma_{\mathbf{V}} \in [r/m, 1]$, ξ_c *defined in* (4.11), (4.12), (4.13), (4.14) *and any* $\lambda_c \in \left[\lambda_c^{\min}, \lambda_c^{\max}\right]$, *for* $\lambda_c^{\max} > \lambda_c^{\min} \geq 0$ *defined in* (4.15), *solving D-RPCA(C) will recover a pair of components* $(\mathbf{L}_0, \mathbf{S}_0) \in \{\mathbf{M}, \mathcal{U}, \mathcal{I}_{\mathcal{S}_c}\}$, *if the space* $\mathcal{R}$ *is structured such that the dictionary* $\mathbf{D} \in \mathbb{R}^{n \times d}$ *obeys the generalized frame property* **D.2** *with frame bounds* $[\alpha_\ell, \alpha_u]$, *for* $\alpha_\ell > 0$.

Theorem 4.2.1 states the conditions under which the solution to the optimization problem D-RPCA(C) will be in the oracle model defined in **D.1**. The condition on the column sparsity $s_c \leq s_c^{\max}$ is a result of the constraint that $\lambda_c^{\min} \geq 0$. Here requiring $\lambda_c^{\max} > \lambda_c^{\min}$ leads to the following sufficient condition on the rank r in terms of the sparsity s_c for $\mu > 0$,

$$r < \left(\sqrt{\frac{\alpha_\ell}{\alpha_u}} \frac{1-\mu}{\mu} - \frac{\xi_c}{\sqrt{\alpha_u}\mu} \sqrt{s_c} \right)^2 .$$

For $\mu = 0$ the condition boils down to that $s_c \leq s_c^{\max}$. Moreover, suppose that α_l and α_u are both close to 1, which can be easily met by a tight frame when $d < n$, or a RIP type condition when $d > n$. Then, if $\frac{(1-\mu)^2}{\beta_{\mathbf{U}}}$ is a constant, since $\gamma_{\mathbf{V}} = \Theta\left(\frac{r}{m}\right)$, we have that $s_c^{\max} = \mathcal{O}\left(\frac{m}{r}\right)$. This is of the same order with the upper bound of s_c in the Outlier Pursuit (OP) [128]. Our numerical results in Section V further show that D-RPCA(C) can be much more robust than OP, and may recover $\{\mathcal{U}, \mathcal{I}_{\mathcal{S}_c}\}$ even when the rank of $\mathbf{L}$ is high and outliers s_c are a constant proportion of m.

Remark 4.2.1. *In essence, Theorem 4.2.1 guarantee recovery of the components as long as the incoherence parameters,* $\mu, \gamma_{\mathbf{V}}$ *and* $\gamma_{\mathbf{U}}$ *are small. As stated in Section 4.2.1, these parameters measure if the low-rank component and the dictionary sparse component can be teased apart from the given data. Specifically, here* μ *measures how close the low-rank component is to the dictionary sparse component. Both* $\gamma_{\mathbf{U}}$ *and* $\beta_{\mathbf{U}}$ *measure how close the column space of the low-rank part* $\mathcal{U}$ *is to the dictionary* $\mathbf{D}$, *while* $\gamma_{\mathbf{V}}$ *measures if the row space of* $\mathbf{L}$ *is sparse. These measures ensure that the components can be identified successfully.*

Finally, to consider the practical performance of the proposed procedure, it is also shown in the simulation results that solving the convex demixing problem (4.10) achieves

better outlier recognition accuracy than other competing methods. We refer to [27] for the details as well as the corresponding entry-wise results. In the next portion, we move forward to discuss the future directions of the research based on this work and the ideas of distributed algorithms.

4.2.3 Discussions of Future Directions

As we know, the computation in solving the optimization problems (4.10) can be heavy when the number of samples is large and the data dimension is high. One interesting direction for future research would be to design distributed algorithms/implementations to speed up the processing. Moreover, there are already some previous works which have proposed the distributed algorithms for PCA and robust PCA [65, 136, 137] as well as matrix factorization and dictionary learning [138, 139]. Among these, [139] proposes a distributed algorithm for collaborative learning of a union-of-subspaces structure underlying distributed data and [137] analyzed the recovery performance for their distributed robust PCA algorithm in both theories and simulations. In contrast, for the proposed dictionary-based outlier pursuit approach considered here, its distributed correspondent has not been considered yet. On the other hand, this type of distributed approaches can be applied naturally in the scenarios where there are multiple agents knowing only a part of the priori target signatures. Therefore it is also meaningful to study the computational and theoretical properties of the distributed dictionary-based robust PCA as compared with those predecessor methods.

4.3 Fast Asynchronous Decentralized Optimization: Allowing Multiple Masters

The section studies asynchronous decentralized optimization over networks. Existing related algorithms are either *centralized* relying on a specific topology, where a single master connects all workers, or *decentralized* without any master by only exchanging information between single-hop neighbors. The present work bridges the gap of current approaches with a novel *hybrid* framework that is able to accommodate multiple masters. Moreover, it enables considerable acceleration of decentralized approaches without physically deploying masters, thus making it possible to achieve a favorable tradeoff between convergence and communication/computation complexity by choosing the configurations. Experimental results showcase its advantages over decentralized counterparts.

Specifically, consider the following distributed optimization problem over a network with N computing nodes (henceforth called workers) which have node-specific cost functions and privately available data

$$\min_{\mathbf{x}} \quad \sum_{i=1}^{N} f_i(\mathbf{x}) + h(\mathbf{x}), \tag{4.16}$$

where $\mathbf{x} \in \mathbb{R}^p$ is the optimization variable, f_i denotes the local cost function at node i and h a (not necessarily smooth) regularizer. The goal is to find the optimal solution by cooperatively solving the per-worker subproblems. This setting arises frequently in estimation, learning and control tasks [15, 16, 26, 140, 141]. Among various solvers, alternating direction method of multiplier (ADMM) has gained popularity for its decomposability and flexibility [15, 16]. Typically, ADMM-based solvers come in two formats: i) *centralized*, where a single master node is connected all workers [15]; and ii) *decentralized*, where no master is present and workers talk to single-hop neighbors only [140, 142]. Until recently [24], there has been no principled approaches to dealing with multiple masters. However, [24] dealt with *synchronous* algorithms that require global coordination, thus challenging their implementation.

Asynchronous algorithms may be more appealing in some settings, especially for decentralized optimization, since they do not need global coordination and consequently

are more efficient when workers differ significantly in computing speed [143–145]. With prior art dealing with either *centralized* or *decentralized* operations, the main contribution of this work to develop an ADMM-based asynchronous decentralized solver of (4.16) using multiple masters is well motivated.

Related work. From the plethora of distributed optimization schemes, we will focus on ADMM-based ones, which can be split in two categories: synchronous and asynchronous.

Synchronous distributed optimization has been studied extensively for decades; see e.g., [16]. These methods may be *centralized* [15] or *decentralized* [140, 142], depending on whether a master (fusion center) is present or not. This setup of multiple masters remains a largely uncharted territory. Progress was made recently in [146] where a cluster of workers are handled together, but no explicit means to accommodate multiple masters. A novel synchronous approach that is capable of handling multiple masters was proposed in [24].

Similarly, asynchronous algorithms are either *centralized* or *decentralized*. Centralized ones are popular, and are easier to analyze and implement [19, 59, 143, 147], but the single master operation faces single-point failure and bottleneck related challenges that limit the overall system performance. Consequently, decentralized alternatives have been developed [143,144]. But no method is available to accommodate multiple masters except for the asynchronous version of [148].

Contributions. We classify our contributions as follows: C1) we develop AH-ADMM that is able to accommodate multiple masters; C2) we show that AH-ADMM enables topology-aware acceleration of decentralized algorithms without changing the underlying network; and C3) we establish convergence of AH-ADMM for nonconvex functions.

The rest of the section is organized as follows. Sec. 4.3.1 presents detailed derivation of AH-ADMM and topology-aware acceleration, Sec. 4.3.2 presents convergence analysis, Sec. 4.3.3 shows numerical tests, and Sec. 4.3.4 concludes this part.

4.3.1 Algorithm

This section provides background about optimization with multiple masters, followed by development of AH-ADMM.

Accommodating multiple masters

The presence of multiple masters makes communication among workers much more

complicated. Some workers may be connected to masters and also other worker, thus necessitating exchange of information to both. We refer to such communication constraints as *hybrid constraints*.

Communication constraints can be effectively described by graphs. The centralized case corresponds to a star graph, with the master at the center and workers around. The decentralized case can be described by a connected graph, whose nodes corresponds to workers and edges represent communication between neighbors. Hybrid constraints are best depicted by *hypergraphs*. Each master is described by a hyperedge consisting of all its connected workers. Fig. 4.1 is an example of hybrid constraints with a master connected to workers 1, 2 and 3.

Problem formulation

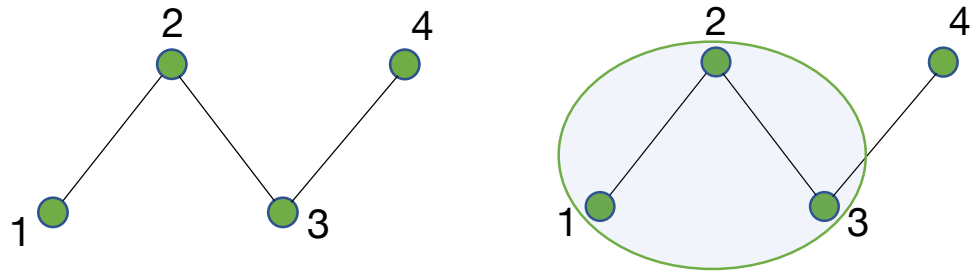
A hypergraph is a tuple $\mathcal{H} := (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \ldots, N\}$ is the vertex set, and $\mathcal{E} := \{\mathcal{E}_i | \mathcal{E}_i \subset \mathcal{V}\}$ denotes the set of hyperedges. Let N be the number of nodes, and $M = |\mathcal{E}|$ the number of (hyper)edges. A vertex i and an edge $\mathcal{E}_j$ are said to be *incident* if $i \in \mathcal{E}_j$. A simple edge can be seen as an hyperedge consisting of two nodes.

By creating copies $\mathbf{x}_i$ at each node and assigning each hyperedge $\mathcal{E}_i$ an auxiliary variable $\mathbf{z}_i$, we can write hybrid constraints uniformly as $\mathbf{x}_i = \mathbf{z}_j$, $\forall i \in \mathcal{E}_j$. Let T be the number of constraints. Consider now vectors $\mathbf{x} \in \mathbb{R}^{Np}$, $\mathbf{z} \in \mathbb{R}^{Mp}$ concatenating $\{\mathbf{x}_i\}, \{\mathbf{z}_j\}$, and also matrices $\widetilde{\mathbf{A}} \in \mathbb{R}^{T \times N}$, $\widetilde{\mathbf{B}} \in \mathbb{R}^{T \times M}$ whose t-th row $\widetilde{A}_{ti} = 1$, $\widetilde{B}_{tj} = 1$ corresponds to t-th constraint $\mathbf{x}_i = \mathbf{z}_j$, $i \in \mathcal{E}_j$. Upon defining $\mathbf{A} = \widetilde{\mathbf{A}} \otimes \mathbf{I}_p$, $\mathbf{B} = \widetilde{\mathbf{B}} \otimes \mathbf{I}_p$, where $\otimes$ is the Kronecker product, problem (4.16) can be formulated as

$$\min_{\mathbf{x}_i} \sum_{i=1}^{N} f_i(\mathbf{x}_i) + \sum_{j=1}^{M} h_j(\mathbf{z}), \quad \text{s. to } \mathbf{A}\mathbf{x} - \mathbf{B}\mathbf{z} = \mathbf{0}. \tag{4.17}$$

where $h_j := h/M$. Let $\widetilde{\mathrm{c}} \in \mathbb{R}^{N \times M}$ be the signless incidence matrix of the hypergraph, meaning $\widetilde{C}_{ij} = 1$ if node i and edge j are incident, and $\widetilde{C}_{ij} = 0$ otherwise. Let D_i denote the degree of node i (number of incident edges), E_j the degree of hyperedge j (number of incident nodes), and diagonal matrix $\widetilde{\mathbf{D}} \in \mathbb{R}^{N \times N}$ and $\widetilde{\mathbf{E}} \in \mathbb{R}^{M \times M}$ collecting $\{D_i\}_{i=1}^N$ and $\{E_j\}_{j=1}^M$, respectively. With $\mathrm{c} := \widetilde{\mathrm{c}} \otimes \mathbf{I}_p$, $\mathbf{D} := \widetilde{\mathbf{D}} \otimes \mathbf{I}_p$, one can show that $\mathbf{A}^\top \mathbf{A} = \mathbf{D}$, $\mathbf{B}^\top \mathbf{B} = \mathbf{E}$ and $\mathbf{A}^\top \mathbf{B} = \mathrm{c}$ (see [24] for the proof).

Example In Fig. 4.1, we have $N = 4$, $M = 2$, and $T = 5$. Specifically, the constraints are $x_i = z_1, i = 1, 2, 3$; $x_3 = z_2$, $x_4 = z_2$. These are expressible in compact form as



(a) (b)

Figure 4.1: An example of hybrid constraints described by a hypergraph. (a) is the underlying graph, and (b) is a hypergraph where the shaded ellipsoid denotes an hyperedge.

$\mathbf{Ax} = \mathbf{Bz}$, where $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{B}}$ are given by

$$\widetilde{\mathbf{A}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \widetilde{\mathbf{B}} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}. \tag{4.18}$$

Asynchronous Hybrid ADMM

The setup of AH-ADMM is similar to that of synchronous hybrid ADMM [24], except that the former has to cope with asynchrony. The high-level description of AH-ADMM is as follows. i) Each worker solves its subproblem individually and communicates the solution to *incident* master(s), then starts waiting until responses from its masters arrive; ii) each master updates its solution whenever updates from a preselected number of workers have been received. Received values can be outdated; and iii) after updating, each master sends results to all *received* workers it received updates from, in order for them to update their associated dual variables.

To describe the process mathematically, we first introduce some definitions. Let k denote a *virtual* iteration counter that is increased by 1 when *any* master finishes its update. By definition, at iteration k, there is exactly *one* master updating, denoted by j_k. Let $\mathcal{A}^k$ be the active set at iteration k, containing received workers of master j_k.

The updates of AH-ADMM are obtained by optimizing the augmented Lagrangian with respect to corresponding variables. However, due to asynchrony, $\mathbf{x}_i^{k+1}$ may depend on outdated values $\widehat{\mathbf{z}}_j$ as incident masters could have updated again while worker i is computing. On the other hand, the update of $\mathbf{z}_{j_k}^{k+1}$ always has access to latest values of $\{\mathbf{x}_i^{k+1}, i \in \mathcal{A}^k\}$, thanks to the updating order, and likewise for $\boldsymbol{\lambda}_t^{k+1}$. All other values not involved remain the same. Equivalently, updates of active workers $i \in \mathcal{A}^k$ can be seen as occuring right before the update of master j_k. Specifically, the updates are

$$\mathbf{x}_i^{k+1} \;=\; \operatorname*{argmin}_{\mathbf{x}} f_i(\mathbf{x}_i) \;+\; \mathbf{x}_i^\top \mathbf{u}_i^k \;+\; \frac{\rho}{2}\Big(D_i\|\mathbf{x}_i\|^2 \;-\; 2\mathbf{x}_i^\top \sum_{i\sim j} \widehat{\mathbf{z}}_j\Big), i \;\in\; \mathcal{A}^k \tag{4.19a}$$

$$\mathbf{z}_{j_k}^{k+1} = \operatorname*{argmin}_{\mathbf{z}_{j_k}} h_{j_k}(\mathbf{z}_{j_k}) - \mathbf{z}_{j_k}^\top \mathbf{v}_{j_k}^k + \frac{\gamma}{2}\|\mathbf{z}_{j_k} - \mathbf{z}_{j_k}^k\|^2 + \frac{\rho}{2}\Big(E_{j_k}\|\mathbf{z}_{j_k}\|^2 - 2\mathbf{z}_{j_k}^\top \sum_{i\in\mathcal{A}^k} \mathbf{x}_i^{k+1}\Big) \tag{4.19b}$$

$$\boldsymbol{\lambda}_t^{k+1} = \boldsymbol{\lambda}_t^k + \rho(\mathbf{A}\mathbf{x}_i^{k+1} - \mathbf{B}\mathbf{z}_j^{k+1}),\; t = \mathcal{T}(i,j) \tag{4.19c}$$

where $\mathcal{T}(i,j) = t$ describes the mapping of worker i and master j to the associated multiplier $\boldsymbol{\lambda}_t$, while $\mathbf{u} := [\mathbf{u}_1^\top, \ldots, \mathbf{u}_N^\top]^\top = \mathbf{A}^\top\boldsymbol{\lambda}$ and $\mathbf{v} := [\mathbf{v}_1^\top, \ldots, \mathbf{v}_M^\top]^\top = \mathbf{B}^\top\boldsymbol{\lambda}$ are change of variables. We include a proximal term in (4.19b) to guarantee the convergence of AH-ADMM, see Theorem 4.3.1. The AH-ADMM algorithm is better understood by considering masters and workers separately, see Algorithms 4 and 5.

Topology-aware Acceleration

Hybrid ADMM generally converges faster than its decentralized counterparts, which is not surprising due to the presence of multiple masters. When no masters are available, AH-ADMM reduces to AD-ADMM, no longer providing any performance gain. Therefore, we are more interested in the question "*Can we benefit from AH-ADMM even if no master exists?*" The answer is *yes*. The idea is to create *virtual masters* inside workers and employ AH-ADMM afterwards. This technique transforms a decentralized optimization problem to one that can be tackled using hybrid ADMM without physically adding nodes or edges.

The first step to apply this technique is to select workers as hosts that serves as *virtual masters* inside. Subsequently, we connect each virtual master to all neighbors

Algorithm 4: AH-ADMM: worker side

Input: ρ, $\boldsymbol{\lambda}^0, \mathbf{z}^0$
while *stop criterion not satisfied* **do**
  for *worker i = 1,2,. . . , N* **do**
    update $\mathbf{x}_i$ by (4.19a)
    while *not enough masters are received* **do**
      wait
    end
    update $\boldsymbol{\lambda}_t$ by (4.19c)
    send $\{\mathbf{x}_i, \boldsymbol{\lambda}_t\}$ to incident masters and neighboring workers
  end
end

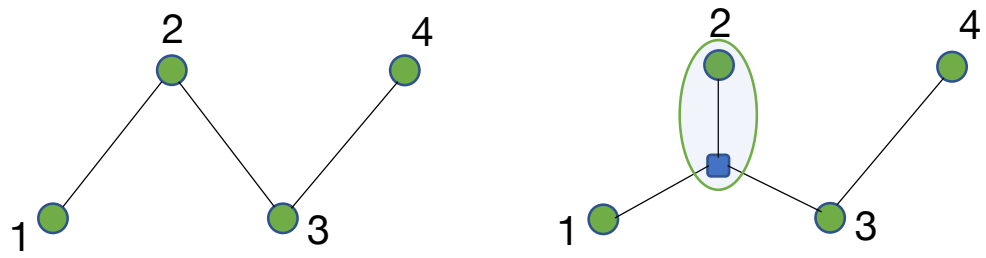
Algorithm 5: AH-ADMM: master side

Input: ρ, $\boldsymbol{\lambda}^0, \mathbf{x}^0$, $\mathbf{z}^0$
while *stop criterion not satisfied* **do**
  for *master j = 1,2,. . . , M* **do**
    while *not enough workers are received* **do**
      wait
    end
    update $\mathbf{z}_j$ by (4.19b)
    send $\mathbf{z}_j$ to all received workers
  end
end

of its host and the host itself, which can be done using all edges of the host. Repeating this procedure creates multiple masters, with the help of whom it becomes possible to employ AH-ADMM subsequently; see also Fig. 4.2 for an example. Notice that in the process, no *physical* nodes or edges have been deployed since virtual masters are just *logical* entities. However, host nodal updates increase complexity. AH-ADMM is also flexible to allow the deployment of any number of virtual masters, a balanced means of boosting performance.

4.3.2 Convergence Analysis

In this section, we analyze the convergence of AH-ADMM. Let τ be the maximum delay, which means that every worker performs update at least once during τ iterations;



(a) (b)

Figure 4.2: Illustration of topology-aware acceleration. The underlying graph is (a), and node 2 is selected as host to serve as virtual master, depicted by the square. The shaded ellipsoid in (b) plays the same role as node 2 in (a).

and $F^\star$ the optimal objective value of (4.17). Inspired by [19], the following theorem establishes the convergence of AH-ADMM. The analysis in [19] assumes a single master. It is nontrivial to extend the reasoning in [19] to multiple masters, because the dual variables in $\boldsymbol{\lambda}$ are coupled.

Theorem 4.3.1. *Suppose that the maximum delay is finite $\tau < \infty$, f_i is twice differentiable and its gradient ∇f_i is Lipschitz with constant L, while h is proper and convex. Then sequences $\{\mathbf{x}_i^k\}_{i=1}^N$, $\{\mathbf{z}_j^k\}_{j=1}^M$ and $\{\boldsymbol{\lambda}_t^k\}_{t=1}^T$ converge to some limit points satisfying the KKT condition of problem* (4.17)*, provided that*

$$0 \leq L_\rho(\mathbf{x}^0, \mathbf{z}^0, \boldsymbol{\lambda}^0) - F^\star < \infty \tag{4.20}$$

$$\rho > \frac{D_{min} + L + \sqrt{(D_{min} + L)^2 + 8L^2 D_{min}}}{2D_{min}} \tag{4.21}$$

$$\gamma \geq \frac{S_m(\tau - 1)^2\rho^2 - \rho E_{min}}{2} \tag{4.22}$$

where $S_m = \max_k |\mathcal{A}^k|$, $D_{min} = \min_i D_i$, $E_{min} = \min_j E_j$.

Theorem 4.3.1 shows that the solution given by AH-ADMM is guaranteed to converge to some KKT points of (4.17), as long as ρ and γ are large enough. Note that f_i does not need to be convex. Different from [143, 144], Theorem 4.3.1 does not need assumptions of random activation of workers, thus being able to cope also with deterministic settings.

Theorem 4.3.1 implies that ρ and γ should be sufficiently large to guarantee the convergence of AH-ADMM. Speicifically, (4.22) suggests that a large γ is needed when the maximum delay is large. Also, (4.21) implies that ρ can be smaller when the cost functions are smoother.

4.3.3 Numerical Experiments

In this section, we carry out numerical experiments to test the acceleration merits of AH-ADMM, and compare with asynchronous decentralized ADMM (AD-ADMM) [143, 144], synchronous decentralized ADMM (SD-ADMM) [140], and synchronous hybrid ADMM [24].

Different from existing works [143–145], our method does not assume every worker can activate each iteration. For this reason, it can deal with more general settings. For example, each worker has a positive activation probability in [143,145] at each iteration, while one edge is randomly selected each time in [144]. Both assumptions exclude the case of deterministic activation patterns, as verified in the following experiment.

In our experiment, the speed of each worker is deterministic and initialized randomly by drawing from a uniform distribution $U[1, 10]$, so that the fastest worker can be 10 times faster than the slowest one, which also ensures bounded delays. We evaluate the performance by plotting its relative error $\frac{\|\mathbf{x}^k-\mathbf{x}^\star\|}{\|\mathbf{x}^0-\mathbf{x}^\star\|}$ against wall clock time. The optimal solution $\mathbf{x}^\star$ can either be computed directly if it admits a closed-form solution, or be obtained using CVX [149, 150]. We also show average working and waiting time of workers to demonstrate the effects of the threshold of masters.

The decentralized *sparse compressed sensing* we tested aims at reconstructing a sparse unknown vector $\mathbf{x} \in \mathbb{R}^p$ through nodal measurements $\mathbf{b}_i = \mathbf{H}_i\mathbf{x}+\mathbf{e}_i,\ i = 1, \ldots, N$, where $\mathbf{H}_i \in \mathbb{R}^{n_i\times p}$ is the sensing matrix of node i and $\mathbf{e}_i$ represents a vector of i.i.d. Gaussian noise. When $p > \sum_{i=1}^N n_i$, there are more unknowns than measurements. The sparsity of $\mathbf{x}$ suggests solving a L_1-regularized least-squares problem

$$\min_{\mathbf{x}} \quad \sum_{i=1}^{N} \frac{1}{2}\|\mathbf{H}_i\mathbf{x} - \mathbf{b}_i\|^2 + \mu\|\mathbf{x}\|_1 \tag{4.23}$$

where $\mu > 0$ is the regularization parameter.

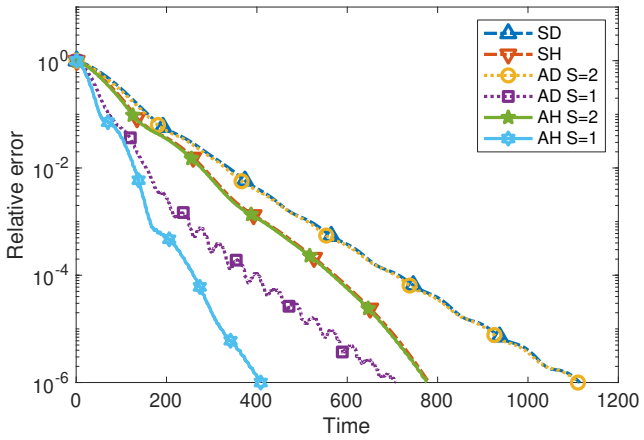


Figure 4.3: Relative error of SD-ADMM (SD), SH-ADMM (SH), AD-ADMM (AD) and AH-ADMM (AH) with different threshold (S) vs. wall clock time.

We consider a ring graph of 10 nodes, and set $n_i = 3$ and $p = 40$ such that $p > \sum_{i=1}^{N} n_i$. The entries of $\mathbf{H}_i$ are generated from the standard Gaussian distribution $N(0,1)$, and then normalized so that $\|\mathbf{H}_i\|_2 = 1$. The unknown vector $\mathbf{x}$ is drawn from $N(0,1)$ with 10% nonzero entries, and subsequently $\mathbf{b}_i$ is generated. Since (4.23) admits no closed-form solution, we solve it using CVX to obtain the optimal solution $\mathbf{x}^\star$. We set $\gamma = 0$ and tune ρ to be nearly optimal.

Fig. 4.3 depicts the relative error against wall clock time. When $S = 2$, AH-ADMM and AD-ADMM are almost equivalent to their synchronous counterparts, while AD-ADMM with $S = 1$ corresponds to the case that each node updates as soon as it receives information from any neighbor, thus eliminating waiting time. One can immediately make two observations: a) asynchronous approaches converge at least as fast as, if not faster, than their decentralized counterparts; and b) hybrid approaches always outperform their decentralized counterparts, showcasing their promising potential. These results evidently show that AH-ADMM consistently outperforms AD-ADMM and synchronous hybrid ADMM with shorter total time, rendering AH-ADMM a faster and more efficient algorithm.

4.3.4 Conclusions and Future Directions

This section presents an asynchronous distributed optimization algorithm, capable of handling multiple masters, AH-ADMM, which not only broadens the applicability of ADMM, but also yields a technique that significantly accelerates the convergence of decentralized ADMM without changing the underlying topology. A convergence result is provided and numerical experiments are performed to compare AH-ADMM against decentralized approaches. Note that here we study the distributed ADMM algorithm for unconstrained problems. One interesting direction for future research would be to consider the generalization of the proposed method to more general problems like the constrained ones in [151]. Moreover, here we present the convergence guarantee without giving the specific rate of the convergence, so more precise analysis of the algorithm's performance during the iteration process is still unexplored. Therefore it is also meaningful to study the theoretical rate characterization for this type of algorithms to guide their applications in practice.

References

[1] E. Dall'Anese, H. Zhu, and G. B. Giannakis. Distributed optimal power flow for smart microgrids. *IEEE Trans. Smart Grid*, 4(3):1464–1475, 2013.

[2] P. Srikantha and D. Kundur. Distributed optimization of dispatch in sustainable generation systems via dual decomposition. *IEEE Transactions on Smart Grid*, 6(5):2501–2509, 2014.

[3] M. Rabbat and R. Nowak. Distributed optimization in sensor networks. In *Proceedings of the International Symposium on Information Processing in Sensor Networks*, pages 20–27. ACM, 2004.

[4] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algorithms. *IEEE transactions on information theory*, 52(6):2508–2530, 2006.

[5] B. N. Oreshkin and M. J. Coates. Asynchronous distributed particle filter via decentralized evaluation of Gaussian products. In *2010 13th International Conference on Information Fusion*, pages 1–8. IEEE, 2010.

[6] Q. Li, R. Heusdens, and M. G. Christensen. Convex optimisation-based privacy-preserving distributed average consensus in wireless sensor networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5895–5899. IEEE, 2020.

[7] K. J Prabuchandran, H. K AN, and S. Bhatnagar. Multi-agent reinforcement learning for traffic signal control. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 2529–2534, 2014.

[8] P. Corke, R. Peterson, and D. Rus. Networked robots: Flying robot navigation using a sensor net. In *International Symposium on Robotics Research*, pages 234–243. Springer, 2005.

[9] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

[10] J. Ren and J. Haupt. A provably communication-efficient asynchronous distributed inference method for convex and nonconvex problems. *IEEE Transactions on Signal Processing*, 68(18):3325–3340, 2020.

[11] A. Agarwal and J. C. Duchi. Distributed delayed stochastic optimization. In *Advances in Neural Information Processing Systems*, pages 873–881, 2011.

[12] J. K. Bradley, A. Kyrola, D. Bickson, and C. Guestrin. Parallel coordinate descent for L1-regularized loss minimization. In *Proceedings of the International Conference on Machine Learning*, pages 321–328, 2011.

[13] J. Tsitsiklis, D. Bertsekas, and M. Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Transactions on Automatic Control*, 31(9):803–812, 1986.

[14] B. Recht, C. Re, S. Wright, and F. Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 693–701, 2011.

[15] 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*, 3(1):1–122, 2011.

[16] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and distributed computation: Numerical methods*, volume 23. Prentice hall Englewood Cliffs, NJ, 1989.

[17] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B. Su. Scaling distributed machine learning with the parameter server. In *OSDI*, volume 14, pages 583–598, 2014.

[18] M. Li, D. G. Andersen, A. J. Smola, and K. Yu. Communication efficient distributed machine learning with the parameter server. In *Advances in Neural Information Processing Systems*, pages 19–27, 2014.

[19] T. Chang, M. Hong, W. Liao, and X. Wang. Asynchronous distributed ADMM for large-scale optimization Part I: Algorithm and convergence analysis. *IEEE Transactions on Signal Processing*, 64(12):3118–3130, 2016.

[20] J. Ren and J. Haupt. Provably communication-efficient asynchronous distributed inference for convex and nonconvex problems. In *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 638–642. IEEE, 2018.

[21] G. Scutari, F. Facchinei, P. Song, D. P Palomar, and J. Pang. Decomposition by partial linearization: Parallel optimization of multi-agent systems. *IEEE Transactions on Signal Processing*, 62(3):641–656, 2014.

[22] Y. Xu and W. Yin. A globally convergent algorithm for nonconvex optimization based on block coordinate update. *Journal of Scientific Computing*, 72(2):700–734, 2017.

[23] P. Tseng and S. Yun. A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programming*, 117(1-2):387–423, 2009.

[24] M. Ma, A. N. Nikolakopoulos, and G. B. Giannakis. Hybrid ADMM: a unifying and fast approach to decentralized optimization. *EURASIP Journal on Advances in Signal Processing*, 2018(1):73, 2018.

[25] M. Ma, J. Ren, G. B. Giannakis, and J. Haupt. Fast asynchronous decentralized optimization: allowing multiple masters. In *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 633–637. IEEE, 2018.

[26] J. Ren, X. Li, and J. Haupt. Communication-efficient distributed optimization for sparse learning via two-way truncation. In *2017 IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pages 1–5. IEEE, 2017.

[27] S. Rambhatla, X. Li, J. Ren, and J. Haupt. A dictionary-based generalization of robust PCA with applications to target localization in hyperspectral imaging. *IEEE Transactions on Signal Processing*, 68:1760–1775, 2020.

[28] S. Meguerdichian, S. Slijepcevic, V. Karayan, and M. Potkonjak. Localized algorithms in wireless ad-hoc networks: location discovery and sensor exposure. In *2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 106–116. ACM, 2001.

[29] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[30] Q. V. Le, A. Karpenko, J. Ngiam, and A. Y. Ng. ICA with reconstruction cost for efficient overcomplete feature learning. In *Advances in Neural Information Processing Systems*, pages 1017–1025, 2011.

[31] H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *Journal of computational and graphical statistics*, 15(2):265–286, 2006.

[32] J. Wang, M. Kolar, N. Srebro, and T. Zhang. Efficient distributed learning with sparsity. In *Proceedings of the International Conference on Machine Learning*, pages 3636–3645, 2017.

[33] M. I. Jordan, J. D. Lee, and Y. Yang. Communication-efficient distributed statistical inference. *Journal of the American Statistical Association*, 114(526):668–681, 2019.

[34] L. Cannelli, F. Facchinei, V. Kungurtsev, and G. Scutari. Asynchronous parallel algorithms for nonconvex optimization. *Mathematical Programming*, pages 1–34.

[35] J. Liu and S. J. Wright. Asynchronous stochastic coordinate descent: Parallelism and convergence properties. *SIAM Journal on Optimization*, 25(1):351–376, 2015.

[36] Z. Wu, Q. Ling, T. Chen, and G. B. Giannakis. Federated variance-reduced stochastic gradient descent with robustness to byzantine attacks. *IEEE Transactions on Signal Processing*, 68:4583–4596, 2020.

[37] D. Alistarh, Z. Allen-Zhu, and J. Li. Byzantine stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 4613–4623, 2018.

[38] Y. Dong, G. B. Giannakis, T. Chen, J. Cheng, M. Hossain, V. Leung, et al. Communication-efficient robust federated learning over heterogeneous datasets. *arXiv preprint arXiv:2006.09992*, 2020.

[39] L. Li, W. Xu, T. Chen, G. B. Giannakis, and Q. Ling. Rsa: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1544–1551, 2019.

[40] W. Dai, A. Kumar, J. Wei, Q. Ho, G. Gibson, and E. P. Xing. High-performance distributed ML at scale through parameter server consistency models. In *29th AAAI Conference on Artificial Intelligence*, 2015.

[41] Y. Yang and M. Pesavento. A unified successive pseudoconvex approximation framework. *IEEE Transactions on Signal Processing*, 65(13):3313–3328, 2017.

[42] W. Shi, Q. Ling, G. Wu, and W. Yin. A proximal gradient algorithm for decentralized composite optimization. *IEEE Transactions on Signal Processing*, 63(22):6013–6023, 2015.

[43] A. Nedić and A. Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48, 2009.

[44] G. Qu and N. Li. Harnessing smoothness to accelerate distributed optimization. *IEEE Transactions on Control of Network Systems*, 5(3):1245–1260, 2018.

[45] A. Nedic, A. Olshevsky, and W. Shi. Achieving geometric convergence for distributed optimization over time-varying graphs. *SIAM Journal on Optimization*, 27(4):2597–2633, 2017.

[46] S. Pu, W. Shi, J. Xu, and A. Nedić. A push-pull gradient method for distributed optimization in networks. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 3385–3390. IEEE, 2018.

[47] P. Di Lorenzo, P. Banelli, E. Isufi, S. Barbarossa, and G. Leus. Adaptive graph signal processing: Algorithms and optimal sampling strategies. *IEEE Transactions on Signal Processing*, 66(13):3584–3598, 2018.

[48] S. Chen, A. Sandryhaila, and J. Kovačević. Distributed algorithm for graph signal inpainting. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3731–3735, 2015.

[49] X. Wang, M. Wang, and Y. Gu. A distributed tracking algorithm for reconstruction of graph signals. *IEEE Journal of Selected Topics in Signal Processing*, 9(4):728–740, 2015.

[50] Y. Tian, Y. Sun, and G. Scutari. Asy-sonata: Achieving linear convergence in distributed asynchronous multiagent optimization. In *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 543–551. IEEE, 2018.

[51] T. Wu, K. Yuan, Q. Ling, W. Yin, and A. H. Sayed. Decentralized consensus optimization with asynchrony and delays. *IEEE Transactions on Signal and Information Processing over Networks*, 4(2):293–307, 2018.

[52] G. Scutari, F. Facchinei, and L. Lampariello. Parallel and distributed methods for constrained nonconvex optimization Part I: Theory. *IEEE Transactions on Signal Processing*, 65(8):1929–1944, 2017.

[53] G. Scutari and Y. Sun. Distributed nonconvex constrained optimization over time-varying digraphs. *Mathematical Programming*, 176(1-2):497–544, 2019.

[54] Y. Sun, G. Scutari, and D. Palomar. Distributed nonconvex multiagent optimization over time-varying networks. In *50th Asilomar Conference on Signals, Systems and Computers*, pages 788–794. IEEE, 2016.

[55] M. Zhu and S. Martínez. An approximate dual subgradient algorithm for multi-agent non-convex optimization. *IEEE Transactions on Automatic Control*, 58(6):1534–1539, 2012.

[56] P. Di Lorenzo and G. Scutari. Next: In-network nonconvex optimization. *IEEE Transactions on Signal and Information Processing over Networks*, 2(2):120–136, 2016.

[57] F. Facchinei, L. Lampariello, and G. Scutari. Feasible methods for nonconvex nonsmooth problems with applications in green communications. *Mathematical Programming*, 164(1-2):55–90, 2017.

[58] Y. Zhang and X. Lin. Disco: Distributed optimization for self-concordant empirical loss. In *Proceedings of the International Conference on Machine Learning*, pages 362–370, 2015.

[59] R. Zhang and J. Kwok. Asynchronous distributed ADMM for consensus optimization. In *Proceedings of the International Conference on Machine Learning*, pages 1701–1709, 2014.

[60] G. Lan, S. Lee, and Y. Zhou. Communication-efficient algorithms for decentralized and stochastic optimization. *Mathematical Programming*, pages 1–48, 2017.

[61] S. Ram, A. Nedić, and V. Veeravalli. Distributed subgradient projection algorithm for convex optimization. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3653–3656. IEEE, 2009.

[62] X. Lian, C. Zhang, H. Zhang, C. 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*, pages 5330–5340, 2017.

[63] Y. Zhang, M. J. Wainwright, and J. C. Duchi. Communication-efficient algorithms for statistical optimization. In *Advances in Neural Information Processing Systems*, pages 1502–1510, 2012.

[64] O. Shamir and N. Srebro. Distributed stochastic optimization and learning. In *52nd Annual Allerton Conference on Communication, Control, and Computing*, pages 850–857, 2014.

[65] J. Zhu, Z. Ge, and Z. Song. Distributed parallel PCA for modeling and monitoring of large-scale plant-wide processes with big data. *IEEE Transactions on Industrial Informatics*, 13:1877–1885, 2017.

[66] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola. Parallelized stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 2595–2603, 2010.

[67] C. Huang and X. Huo. A distributed one-step estimator. *Mathematical Programming*, 174(1-2):41–76, 2019.

[68] C. Ma, V. Smith, M. Jaggi, M. I. Jordan, P. Richtárik, and M. Takác. Adding vs. averaging in distributed primal-dual optimization. In *Proceedings of the International Conference on Machine Learning*, pages 1973–1982, 2015.

[69] M. Jaggi, V. Smith, M. Takác, J. Terhorst, S. Krishnan, T. Hofmann, and M. I. Jordan. Communication-efficient distributed dual coordinate ascent. In *Advances in Neural Information Processing Systems*, pages 3068–3076, 2014.

[70] O. Shamir, N. Srebro, and T. Zhang. Communication-efficient distributed optimization using an approximate Newton-type method. In *Proceedings of the International Conference on Machine Learning*, volume 32, pages 1000–1008, 2014.

[71] J. D. Lee, Y. Sun, and M. A. Saunders. Proximal Newton-type methods for minimizing composite functions. *SIAM Journal on Optimization*, 24(3):1420–1443, 2014.

[72] M. Hong, Z. Luo, and M. Razaviyayn. Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems. *SIAM Journal on Optimization*, 26(1):337–364, 2016.

[73] C. Cartis, N. I. Gould, and P. L. Toint. On the complexity of steepest descent, Newton's and regularized Newton's methods for nonconvex unconstrained optimization problems. *SIAM Journal on Optimization*, 20(6):2833–2852, 2010.

[74] M. Hong, D. Hajinezhad, and M. Zhao. Prox-PDA: The proximal primal-dual algorithm for fast distributed nonconvex optimization and learning over networks.

In *Proceedings of the International Conference on Machine Learning*, pages 1529–1538, 2017.

[75] S. M. Kakade, S. Shalev-Shwartz, and A. Tewari. Regularization techniques for learning with matrices. *Journal of Machine Learning Research*, 13(Jun):1865–1890, 2012.

[76] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354, 2017.

[77] J. K Gupta, M. Egorov, and M. Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 66–83. Springer, 2017.

[78] C. Yu, J. Lan, Z. Guo, and Y. Hu. DROM: Optimizing the routing in software-defined networks with deep reinforcement learning. *IEEE Access*, 6:64533–64539, 2018.

[79] D. Lee, H. Yoon, and N. Hovakimyan. Primal-dual algorithm for distributed reinforcement learning: distributed GTD. In *IEEE Conference on Decision and Control (CDC)*, pages 1967–1972, 2018.

[80] H. Wai, Z. Yang, Z. Wang, and M. Hong. Multi-agent reinforcement learning via double averaging primal-dual optimization. In *Advances in Neural Information Processing Systems*, pages 9672–9683. 2018.

[81] B. Gharesifard and J. Cortés. When does a digraph admit a doubly stochastic adjacency matrix? In *American Control Conference (ACC)*, pages 2440–2445. IEEE, 2010.

[82] B. Gharesifard and J. Cortés. Distributed strategies for generating weight-balanced and doubly stochastic digraphs. *European Journal of Control*, 18(6):539–557, 2012.

[83] A. Nedić, A. Olshevsky, W. Shi, and C. A Uribe. Geometrically convergent distributed optimization with uncoordinated step-sizes. In *American Control Conference (ACC)*, pages 3950–3955. IEEE, 2017.

[84] F. Zanella, D. Varagnolo, A. Cenedese, G. Pillonetto, and L. Schenato. Newton-Raphson consensus for distributed convex optimization. In *Decision and Control and European Control Conference (CDC-ECC)*, pages 5917–5922. IEEE, 2011.

[85] R. Xin, U. A Khan, and S. Kar. Variance-reduced decentralized stochastic optimization with gradient tracking. *arXiv preprint arXiv:1909.11774*, 2019.

[86] K. Yuan, B. Ying, X. Zhao, and A. H. Sayed. Exact diffusion for distributed optimization and learning–Part I: Algorithm development. *IEEE Transactions on Signal Processing*, 67(3):708–723, 2018.

[87] M. L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine Learning Proceedings*, pages 157–163. Elsevier, 1994.

[88] M. L Littman. Value-function reinforcement learning in Markov games. *Cognitive Systems Research*, 2(1):55–66, 2001.

[89] J. Hu and M. P Wellman. Nash Q-learning for general-sum stochastic games. *Journal of machine learning research*, 4(Nov):1039–1069, 2003.

[90] G. Arslan and S. Yüksel. Decentralized Q-learning for stochastic teams and games. *IEEE Transactions on Automatic Control*, 62(4):1545–1558, 2017.

[91] L. Bu, R. Babu, and B. De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics*, 38(2):156–172, 2008.

[92] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Basar. Fully decentralized multi-agent reinforcement learning with networked agents. In *Proceedings of the International Conference on Machine Learning*, pages 5872–5881, 2018.

[93] T. Chen, K. Zhang, G. B. Giannakis, and T. Başar. Communication-efficient distributed reinforcement learning. *arXiv preprint arXiv:1812.03239*, 2018.

[94] S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *Proceedings of the International Conference on Machine Learning*, pages 2681–2690, 2017.

[95] X. Lian, M. Wang, and J. Liu. Finite-sum composition optimization via variance reduced gradient descent. In *Artificial Intelligence and Statistics*, pages 1159–1167, 2017.

[96] B. Dai, N. He, Y. Pan, B. Boots, and L. Song. Learning from conditional distributions via dual embeddings. In *Artificial Intelligence and Statistics*, pages 1458–1467, 2017.

[97] S. S Du, J. Chen, L. Li, L. Xiao, and D. Zhou. Stochastic variance reduction methods for policy evaluation. In *Proceedings of the International Conference on Machine Learning*, pages 1049–1058, 2017.

[98] S. V. Macua, J. Chen, S. Zazo, and A. H Sayed. Distributed policy evaluation under multiple behavior strategies. *IEEE Transactions on Automatic Control*, 60(5):1260–1274, 2015.

[99] W. Shi, Q. Ling, G. Wu, and W. Yin. Extra: An exact first-order algorithm for decentralized consensus optimization. *SIAM Journal on Optimization*, 25(2):944–966, 2015.

[100] M. Schmidt, N. Le Roux, and F. Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2):83–112, 2017.

[101] X. Wang, J. Zhou, S. Mou, and M. J Corless. A distributed algorithm for least squares solutions. *IEEE Transactions on Automatic Control*, 64(10):4217–4222, 2019.

[102] R. S Sutton and A. G Barto. *Reinforcement learning: An introduction, Second Edition*. MIT press, 2018.

[103] J. Cortes, S. Martinez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation*, 20(2):243–255, 2004.

[104] J. Kober, J. A Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[105] A. Nedic and D. P Bertsekas. Least squares policy evaluation algorithms with linear function approximation. *Discrete Event Dynamic Systems*, 13(1-2):79–110, 2003.

[106] Christopher D. G and Gordon F. R. Algebraic graph theory. In *Graduate texts in mathematics*, 2001.

[107] R. A Horn and C. R Johnson. *Matrix analysis*. Cambridge University Press, 1990.

[108] P. Erdos and A. Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

[109] X. Li, J. Ren, S. Rambhatla, Y. Xu, and J. Haupt. Robust PCA via dictionary based outlier pursuit. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4699–4703. IEEE, 2018.

[110] S. U Stich, J. Cordonnier, and M. Jaggi. Sparsified sgd with memory. In *Advances in Neural Information Processing Systems*, pages 4447–4458, 2018.

[111] S. Pu, W. Shi, J. Xu, and A. Nedic. Push-pull gradient methods for distributed optimization in networks. *IEEE Transactions on Automatic Control*, 2020.

[112] T. Chen, G. B. Giannakis, T. Sun, and W. Yin. Lag: Lazily aggregated gradient for communication-efficient distributed learning. In *Advances in Neural Information Processing Systems*, pages 5050–5060, 2018.

[113] J. Sun, T. Chen, G. B. Giannakis, and Z. Yang. Communication-efficient distributed learning via lazily aggregated quantized gradients. In *Advances in Neural Information Processing Systems*, pages 3370–3380, 2019.

[114] J. Friedman, T. Hastie, H. Höfling, R. Tibshirani, et al. Pathwise coordinate optimization. *The Annals of Applied Statistics*, 1(2):302–332, 2007.

[115] Amir B. and Marc T. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[116] L. Xiao and T. Zhang. A proximal-gradient homotopy method for the sparse least-squares problem. *SIAM Journal on Optimization*, 23(2):1062–1091, 2013.

[117] J. D. Lee, Q. Lin, T. Ma, and T. Yang. Distributed stochastic variance reduced gradient methods and a lower bound for communication complexity. *arXiv preprint arXiv:1507.07595*, 2015.

[118] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao. Optimal distributed online prediction using mini-batches. *Journal of Machine Learning Research*, 13(Jan):165–202, 2012.

[119] Y. Arjevani and O. Shamir. Communication complexity of distributed convex learning and optimization. In *Advances in Neural Information Processing Systems*, pages 1756–1764, 2015.

[120] J. D. Lee, Q. Liu, Y. Sun, and J. E. Taylor. Communication-efficient sparse regression. *Journal of Machine Learning Research*, 18(5):1–30, 2017.

[121] X. Yuan, P. Li, and T. Zhang. Gradient hard thresholding pursuit for sparsity-constrained optimization. In *Proceedings of the International Conference on Machine Learning*, pages 127–135, 2014.

[122] P. Jain, A. Tewari, and P. Kar. On iterative hard thresholding methods for high-dimensional m-estimation. In *Advances in Neural Information Processing Systems*, pages 685–693, 2014.

[123] M. Rudelson and S. Zhou. Reconstruction from anisotropic random measurements. In *Conference on Learning Theory*, pages 10–1, 2012.

[124] R. Vershynin. Introduction to the non-asymptotic analysis of random matrices. *arXiv preprint arXiv:1011.3027*, 2010.

[125] M. J Wainwright. Sharp thresholds for high-dimensional and noisy recovery of sparsity using $l1$-constrained quadratic programming. *IEEE Transactions on Information Theory*, 2009.

[126] S. N Negahban, P. Ravikumar, M. J Wainwright, and B. Yu. A unified framework for high dimensional analysis of m-estimators with decomposable regularizers. *Statistical Science*, 27(4):538–557, 2012.

[127] M. Mardani, G. Mateos, and G. B. Giannakis. Recovery of low-rank plus compressed sparse matrices with application to unveiling traffic anomalies. *IEEE Transactions on Information Theory*, 59(8):5186–5205, 2013.

[128] H. Xu, C. Caramanis, and S. Sanghavi. Robust pca via outlier pursuit. In *Advances in neural information processing systems*, pages 2496–2504, 2010.

[129] M. Rahmani and G. K Atia. Randomized robust subspace recovery and outlier detection for high dimensional data matrices. *IEEE Transactions on Signal Processing*, 65(6):1580–1594, 2016.

[130] B. Mehta and W. Nejdl. Attack resistant collaborative filtering. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 75–82, 2008.

[131] A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. *ACM SIGCOMM computer communication review*, 34(4):219–230, 2004.

[132] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 20(11):1254–1259, 1998.

[133] J. Harel, C. Koch, and P. Perona. Graph-based visual saliency. In *Advances in neural information processing systems*, pages 545–552, 2007.

[134] T. Liu, Z. Yuan, J. Sun, J. Wang, N. Zheng, X. Tang, and H. Shum. Learning to detect a salient object. *IEEE Transactions on Pattern analysis and machine intelligence*, 33(2):353–367, 2010.

[135] R. J Duffin and A. C Schaeffer. A class of nonharmonic fourier series. *Transactions of the American Mathematical Society*, 72(2):341–366, 1952.

[136] L. Huang, X. Nguyen, M. Garofalakis, M. Jordan, A. Joseph, and N. Taft. Distributed pca and network anomaly detection. In *In Proceedings of NIPS*, volume 2006, 2006.

[137] J. Feng, H. Xu, and S. Mannor. Distributed robust learning. *arXiv preprint arXiv:1409.5937*, 2014.

[138] J. Liang, M. Zhang, X. Zeng, and G. Yu. Distributed dictionary learning for sparse representation in sensor networks. *IEEE Transactions on Image Processing*, 23(6):2528–2541, 2014.

[139] H. Raja and W. U Bajwa. Cloud K-SVD: A collaborative dictionary learning algorithm for big, distributed data. *IEEE Transactions on Signal Processing*, 64(1):173–188, 2015.

[140] G. B. Giannakis, Q. Ling, G. Mateos, I. D Schizas, and H. Zhu. Decentralized learning for wireless communications and networking. In *Splitting Methods in Communication, Imaging, Science, and Engineering*, pages 461–497. Springer, 2016.

[141] R. Olfati-Saber, J. A Fax, and R. M Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

[142] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin. On the linear convergence of the admm in decentralized consensus optimization. *IEEE Transactions on Signal Processing*, 62(7):1750–1761, 2014.

[143] Z. Peng, Y. Xu, M. Yan, and W. Yin. Arock: an algorithmic framework for asynchronous parallel coordinate updates. *SIAM Journal on Scientific Computing*, 38(5):A2851–A2879, 2016.

[144] E. Wei and A. Ozdaglar. On the o (1/k) convergence of asynchronous distributed alternating direction method of multipliers. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 551–554. IEEE, 2013.

[145] T. Wu, K. Yuan, Q. Ling, W. Yin, and A. H Sayed. Decentralized consensus optimization with asynchrony and delays. *IEEE Transactions on Signal and Information Processing over Networks*, 4(2):293–307, 2017.

[146] F. Iutzeler, P. Bianchi, P. Ciblat, and W. Hachem. Explicit convergence rate of a distributed alternating direction method of multipliers. *IEEE Transactions on Automatic Control*, 61(4):892–904, 2015.

[147] M. Hong. A distributed, asynchronous, and incremental algorithm for nonconvex optimization: an admm approach. *IEEE Transactions on Control of Network Systems*, 5(3):935–945, 2017.

[148] F. Iutzeler, P. Bianchi, P. Ciblat, and W. Hachem. Asynchronous distributed optimization using a randomized alternating direction method of multipliers. In *52nd IEEE conference on decision and control*, pages 3671–3676. IEEE, 2013.

[149] M. Grant, S. Boyd, and Y. Ye. Matlab software for disciplined convex programming, 2008.

[150] M. Grant and S. Boyd. Graph implementations for nonsmooth convex programs. In *Recent advances in learning and control*, pages 95–110. Springer, 2008.

[151] T. Chang. A proximal dual consensus ADMM method for multi-agent constrained optimization. *IEEE Transactions on Signal Processing*, 64(14):3719–3734, 2016.

[152] O. Perron. Zur theorie der matrices. *Mathematische Annalen*, 64(2):248–263, 1907.

Chapter 5

Appendices

5.1 Analysis Details for Chapter 2

In the appendix, we prove the main results including Theorem 2.3.1, Theorem 2.3.2, their key lemmata, and the corollaries.

5.1.1 Proof of Theorem 2.3.1

Proof of Theorem 2.3.1. We begin by establishing the first conclusion of the theorem. The idea is to use the cumulative descent of the value of function $\mathbf{F}(\mathbf{x}^{t+1}, \mathbf{x}^t)$ to bound the summation of the gaps between $\mathbf{x}^t$ and $\mathbf{x}^{t+1}$. Summing inequality (2.9) in Lemma 2.3.2 over t yields

$$\mathbf{F}(\mathbf{x}^{T+1}, \mathbf{x}^T) - \mathbf{F}(\mathbf{x}^1, \mathbf{x}^0) \leq \sum_{t=1}^{T} \Big(\frac{3L}{2} - \frac{\gamma(\rho)}{2} + L\delta\tau \Big) \big\| \Delta^{(t)} \big\|^2 + \sum_{t=1}^{T} \Big(\frac{L\tau}{\delta} - \frac{\rho}{2} \Big) \big\| \Delta^{(t-1)} \big\|^2.$$

To simplify the right hand side of this inequality, now define

$$c := \min \Big\{ \frac{\gamma(\rho)}{2} - \frac{3L}{2} - L\delta\tau, \frac{\rho}{2} - \frac{L\tau}{\delta} \Big\},$$

by Assumption 2.3.2 we have $\gamma(\rho) > 3L + 2L\delta\tau$ and $\rho > \frac{2L\tau}{\delta}$, therefore $c > 0$. It holds

that

$$\mathbf{F}(\mathbf{x}^{T+1}, \mathbf{x}^T) - \mathbf{F}(\mathbf{x}^1, \mathbf{x}^0) \leq -c \sum_{t=0}^{T} \left\| \Delta^{(t)} \right\|^2. \tag{5.1}$$

Note that by Lemma 2.3.3 the LHS of (5.1) is bounded from below. By letting $T \to \infty$, it follows that

$$\left\| \Delta^{(t)} \right\| \to 0, \qquad t \to \infty.$$

Moreover, Lemma 2.3.3 shows that $\mathbf{F}(\mathbf{x}^{t+1}, \mathbf{x}^t)$ is bounded, but due to the coerciveness assumption

$$\lim_{\|\mathbf{x}\| \to \infty} \frac{1}{m} \sum_{j \in [m]} \mathbf{L}_j(\mathbf{x}) + h(\mathbf{x}) + \frac{\rho}{2} \left\| \mathbf{x} - \mathbf{x}^t \right\|^2 = +\infty, \tag{5.2}$$

so we know $\{\mathbf{x}^{t+1}\}$ is bounded. Therefore the first conclusion is proved.

We now establish the second conclusion of the theorem, which claims that every limit point of the iterates generated by Algorithm 3 is a stationary point. From (2.3), we know that

$$\begin{aligned} \mathbf{x}^{t+1} = \mathbf{Prox}_h \Big[\mathbf{x}^{t+1} - \Big(\nabla \mathbf{L}_1(\mathbf{x}^{t+1}) + \frac{1}{m} \sum_{j \in [m]} \nabla \mathbf{L}_j(\mathbf{x}^{t_j}) \\ - \nabla \mathbf{L}_1(\mathbf{x}^{t_1}) + \rho \left(\mathbf{x}^{t+1} - \mathbf{x}^t \right) \Big) \Big], \end{aligned} \tag{5.3}$$

here we recall $\mathbf{Prox}_h$ is a proximal operator defined by $\mathbf{Prox}_h[\mathbf{z}] := \underset{\mathbf{x}}{\operatorname{argmin}}\, h(\mathbf{x}) + \frac{1}{2}\|\mathbf{x} - \mathbf{z}\|^2$. Equation (5.3) further implies that

$$\begin{aligned} &\left\| \mathbf{x}^t - \mathbf{Prox}_h \left(\mathbf{x}^t - \tfrac{1}{m} \sum_{j \in [m]} \nabla \mathbf{L}_j(\mathbf{x}^t) \right) \right\| \\ &\leq \left\| \mathbf{x}^t - \mathbf{x}^{t+1} + \mathbf{x}^{t+1} - \mathbf{Prox}_h \left(\mathbf{x}^t - \frac{1}{m} \sum_{j \in [m]} \nabla \mathbf{L}_j(\mathbf{x}^t) \right) \right\| \end{aligned}$$

$$
\begin{aligned}
&\leq \left\|\mathbf{x}^t - \mathbf{x}^{t+1}\right\| \\
&+ \left\| \mathbf{Prox}_h \left[\mathbf{x}^{t+1} - \left(\nabla \mathbf{L}_1(\mathbf{x}^{t+1}) + \frac{1}{m} \sum_{j\in[m]} \nabla \mathbf{L}_j(\mathbf{x}^{t_j}) \right.\right.\right. \\
&\left.\left.\left. - \nabla \mathbf{L}_1(\mathbf{x}^{t_1}) + \rho\big(\mathbf{x}^{t+1} - \mathbf{x}^t\big) \right) \right] - \mathbf{Prox}_h \left(\mathbf{x}^t - \frac{1}{m} \sum_{j\in[m]} \nabla \mathbf{L}_j(\mathbf{x}^t) \right) \right\| \\
&\overset{(a)}{\leq} \left\| (1+\rho)(\mathbf{x}^{t+1} - \mathbf{x}^t) + \frac{1}{m} \sum_{j\in[m]} \nabla \mathbf{L}_j(\mathbf{x}^t) \right. \\
&\left. - \frac{1}{m} \sum_{j\in[m]} \nabla \mathbf{L}_j(\mathbf{x}^{t_j}) - \big(\nabla \mathbf{L}_1(\mathbf{x}^{t+1}) - \nabla \mathbf{L}_1(\mathbf{x}^{t_1})\big) \right\| + \left\|\Delta^{(t)}\right\| \\
&\leq (2+\rho) \left\|\Delta^{(t)}\right\| + 2L \sum_{k=0}^{\tau} \left\|\Delta^{(t-k)}\right\|, \qquad (5.4)
\end{aligned}
$$

which gives that $\left\|\mathbf{x}^t - \mathbf{Prox}_h(\mathbf{x}^t - \frac{1}{m} \sum_{j\in[m]} \nabla \mathbf{L}_j(\mathbf{x}^t))\right\| \to 0$, as $t \to \infty$. Note that here inequality (a) holds because of the nonexpansiveness of the operator $\mathbf{Prox}_h$. The last inequality follows from Assumption 2.3.1.

Let $\mathbf{X}^*$ be the set of stationary points of problem (4.1), and define

$$\operatorname{dist}\left(\mathbf{x}^t, \mathbf{X}^*\right) := \min_{\widehat{x}\in\mathbf{X}^*} \|\mathbf{x}^t - \widehat{\mathbf{x}}\|$$

as the distance between $\mathbf{x}^t$ and the set $\mathbf{X}^*$. Now we prove that every limit point of $\{\mathbf{x}^t\}$ is a stationary point.

We establish this conclusion by showing that the distance between $\mathbf{X}^*$ and its subsequences converges to 0. Suppose on the contrary there exists a subsequence $\{\mathbf{x}^{t_k}\}$ of $\{\mathbf{x}^t\}$ such that $\mathbf{x}^{t_k} \to \widehat{\mathbf{x}},\ k \to \infty$ but

$$\lim_{k\to\infty} \operatorname{dist}(\mathbf{x}^{t_k}, \mathbf{X}^*) \geq \gamma > 0. \qquad (5.5)$$

Then it is obvious that $\lim_{k\to\infty} \operatorname{dist}(\mathbf{x}^{t_k}, \widehat{\mathbf{x}}) = 0$. Therefore, there exists some $K(\gamma) > 0$,

such that

$$\|\mathbf{x}^{t_k} - \widehat{\mathbf{x}}\| \leq \frac{\gamma}{2}, \quad k > K(\gamma). \tag{5.6}$$

On the other hand, from (5.4) and the lower semi-continuity of $h(\mathbf{x})$ we have $\widehat{\mathbf{x}} \in \mathbf{X}^*$, so by the definition of the distance function we have

$$\operatorname{dist}(\mathbf{x}^{t_k}, \mathbf{X}^*) \leq \operatorname{dist}(\mathbf{x}^{t_k}, \widehat{\mathbf{x}}). \tag{5.7}$$

Combining (5.6) and (5.7), we must have

$$\operatorname{dist}(\mathbf{x}^{t_k}, \mathbf{X}^*) \leq \frac{\gamma}{2}, \quad k > K(\gamma).$$

This contradicts to (5.5), so the second result is proved.

We finally prove the third conclusion of the theorem, which gives the sublinear convergence rate of the optimality gap. Summing (5.4) over t yields

$$\begin{aligned}
&\sum_{t=0}^{T} \left\| \mathbf{x}^t - \mathbf{Prox}_h \left(\mathbf{x}^t - \frac{1}{m} \sum_{j \in [m]} \nabla \mathbf{L}_j(\mathbf{x}^t) \right) \right\|^2 \\
&\leq \sum_{t=0}^{T} 2(2+\rho)^2 \left\| \Delta^{(t)} \right\|^2 + 2(2L)^2 \sum_{t=0}^{T} \sum_{k=0}^{\tau} \left\| \Delta^{(t-k)} \right\|^2 \\
&\leq \left(2(2+\rho)^2 + 8L^2\tau \right) \sum_{t=0}^{T} \left\| \Delta^{(t)} \right\|^2.
\end{aligned} \tag{5.8}$$

Combining (5.1) and (5.8) we can bound the optimality gap by the cumulative descent of the Lyapunov function:

$$\sum_{t=0}^{T} \left\| \widetilde{\nabla} \mathbf{L}\big(\mathbf{x}^t\big) \right\|^2 \leq \frac{\mu}{c} \left(\mathbf{F}(\mathbf{x}^1, \mathbf{x}^0) - \mathbf{F}(\mathbf{x}^{T+1}, \mathbf{x}^T) \right),$$

where $\mu := \big(2(2+\rho)^2 + 8L^2\tau\big)$.

Let $T(\epsilon) := \min\left\{t \mid \left\|\widetilde{\nabla}\mathbf{L}(\mathbf{x}^t)\right\|^2 \leq \epsilon, t \geq 0\right\}$. Then the above inequality implies

$$T(\epsilon)\epsilon \leq \frac{\mu}{c}\left(\mathbf{F}(\mathbf{x}^1, \mathbf{x}^0) - \mathbf{F}(\mathbf{x}^{T+1}, \mathbf{x}^T)\right).$$

Thus from Lemma 2.3.3 it follows the third conclusion of Theorem 2.3.1 that

$$\epsilon \leq \frac{C \cdot \left(\mathbf{F}(\mathbf{x}^1, \mathbf{x}^0) - \underline{\mathbf{F}}\right)}{T(\epsilon)},$$

where $C := \frac{\mu}{c} > 0$. Therefore the conclusions of Theorem 2.3.1 are proved. □

5.1.2 Proof of Theorem 2.3.2

Proof of Theorem 2.3.2. We begin by defining $\widetilde{\Delta}^{(t+1)} = \mathbf{F}(\mathbf{x}^{t+1}, \mathbf{x}^t) - \mathbf{F}(\mathbf{x}^*, \mathbf{x}^*)$, the optimality gap in terms of the Lyapunov function. Note that $\mathbf{F}(\mathbf{x}, \mathbf{x}^t) = \frac{1}{m}\sum\limits_{j\in[m]}\mathbf{L}_j(\mathbf{x}) + \frac{\rho}{2}\|\mathbf{x}-\mathbf{x}^t\|^2 + h(\mathbf{x}) \geq \mathbf{F}(\mathbf{x}^*, \mathbf{x}^*) = \frac{1}{m}\sum\limits_{j\in[m]}\mathbf{L}_j(\mathbf{x}^*) + h(\mathbf{x}^*)$. Therefore $\widetilde{\Delta}^{(t+1)}$ is nonnegative.

Moreover, from Lemma 2.3.2 it follows that

$$\begin{aligned}\widetilde{\Delta}^{(t+1)} \leq \widetilde{\Delta}^{(t)} &+ \Big(\frac{3L}{2} - \frac{\rho}{2} + L\delta\tau\Big)\big\|\Delta^{(t)}\big\|^2 \\ &-\frac{\rho}{2}\big\|\Delta^{(t-1)}\big\|^2 + \frac{L}{\delta}\sum_{k=1}^{\tau}\big\|\Delta^{(t-k)}\big\|^2. \end{aligned} \tag{5.9}$$

Note that from (2.13) of Lemma 2.3.4 we have

$$\begin{aligned}\frac{1}{\frac{\rho}{2}(1+\delta_1)+\delta_1}\widetilde{\Delta}^{(t+1)} \leq &\frac{\delta_1 L + \frac{\rho}{2}(1+\delta_1)}{\frac{\rho}{2}(1+\delta_1)+\delta_1}\big\|\Delta^{(t)}\big\|^2 \\ &+\frac{1}{\frac{\rho}{2}(1+\delta_1)+\delta_1}\frac{\delta_1}{2m}L^2\sum_{j\in[m]}\|\mathbf{x}^{t_j} - \mathbf{x}^t\|^2. \end{aligned} \tag{5.10}$$

Based on these two inequalities, the conclusion can be proved. Specifically, by summing (5.10) and (5.9), one has

$$
\begin{aligned}
&\Big(1+\frac{1}{\frac{\rho}{2}(1+\delta_1)+\delta_1}\Big)\widetilde{\Delta}^{(t+1)}\\
&\leq \widetilde{\Delta}^{(t)}+\Big[\frac{\delta_1 L+\frac{\rho}{2}(1+\delta_1)}{\frac{\rho}{2}(1+\delta_1)+\delta_1}+\frac{3L}{2}-\frac{\rho}{2}+L\delta\tau\Big]\big\|\Delta^{(t)}\big\|^2\\
&-\frac{\rho}{2}\big\|\Delta^{(t-1)}\big\|^2+\frac{L}{\delta}\sum_{k=1}^{\tau}\big\|\Delta^{(t-k)}\big\|^2\\
&+\frac{1}{\frac{\rho}{2}(1+\delta_1)+\delta_1}\frac{\delta_1}{2m}L^2\sum_{j\in[m]}\|\mathbf{x}^{t_j}-\mathbf{x}^t\|^2. \qquad (5.11)
\end{aligned}
$$

Inequality (5.11) gives us an relation between $\widetilde{\Delta}^{(t+1)}$ and $\widetilde{\Delta}^{(t)}$. Let us define $\eta := 1+\frac{1}{\frac{\rho}{2}(1+\delta_1)+\delta_1}$ and

$$
(P3) := \frac{\delta_1 L+\frac{\rho}{2}(1+\delta_1)}{\frac{\rho}{2}(1+\delta_1)+\delta_1}+\frac{3L}{2}-\frac{\rho}{2}+L\delta\tau.
$$

Then by applying (5.11) recursively we have

$$
\begin{aligned}
\widetilde{\Delta}^{(t+1)} &\leq \frac{1}{\eta}\widetilde{\Delta}^{(t)}+\frac{1}{\eta}(P3)\big\|\Delta^{(t)}\big\|^2-\frac{\rho}{2\eta}\big\|\Delta^{(t-1)}\big\|^2\\
&+\frac{L}{\delta\eta}\sum_{k=1}^{\tau}\big\|\Delta^{(t-k)}\big\|^2+\frac{1}{\eta}\frac{1}{\frac{\rho}{2}(1+\delta_1)+\delta_1}\frac{\delta_1 L^2}{2m}\sum_{j\in[m]}\|\mathbf{x}^{t_j}-\mathbf{x}^t\|^2\\
&\leq \frac{1}{\eta^2}\widetilde{\Delta}^{(t-1)}+\frac{1}{\eta}\left(\frac{1}{\eta}(P3)\big\|\Delta^{(t-1)}\big\|^2-\frac{\rho}{2\eta}\big\|\Delta^{(t-2)}\big\|^2\right)\\
&+\frac{1}{\eta}(P3)\big\|\Delta^{(t)}\big\|^2-\frac{\rho}{2\eta}\big\|\Delta^{(t-1)}\big\|^2\\
&+\left(\frac{L}{\delta\eta}\sum_{k=1}^{\tau}\big\|\Delta^{(t-k)}\big\|^2+\frac{L}{\delta\eta^2}\sum_{k=1}^{\tau}\big\|\Delta^{(t-1-k)}\big\|^2\right)\\
&+\frac{1}{\eta}\frac{1}{\frac{\rho}{2}(1+\delta_1)+\delta_1}\frac{\delta_1}{2m}L^2\tau\sum_{l=0}^{1}\frac{1}{\eta^{l+1}}\sum_{j\in[m]}\sum_{k=1}^{\tau}\big\|\Delta^{(t-l-k)}\big\|^2\\
&\cdots
\end{aligned}
$$

$$
\begin{aligned}
&\leq \frac{1}{\eta^t}\widetilde{\Delta}^{(1)} + \frac{1}{\eta}(P3)\big\|\Delta^{(t)}\big\|^2 + \Big(\frac{1}{\eta^2}(P3) - \frac{\rho}{2\eta}\Big)\big\|\Delta^{(t-1)}\big\|^2 \\
&+ \big(\frac{1}{\eta^3}(P3) - \frac{\rho}{2\eta^2}\big)\big\|\Delta^{(t-2)}\big\|^2 + \cdots \\
&+ \big(\frac{1}{\eta^{t+1}}(P3) - \frac{\rho}{2\eta^t}\big)\big\|\Delta^{(0)}\big\|^2 + \Big(\frac{L}{\delta\eta}\sum_{l=0}^{t}\frac{1}{\eta^l}\sum_{k=1}^{\tau}\big\|\Delta^{(t-l-k)}\big\|^2\Big) \\
&+ \frac{1}{\frac{\rho}{2}(1+\delta_1)+\delta_1}\frac{\delta_1}{2m}L^2\tau\sum_{l=0}^{t}\frac{1}{\eta^{l+1}}\sum_{j\in[m]}\sum_{k=1}^{\tau}\big\|\Delta^{(t-l-k)}\big\|^2 \\
&\leq \frac{1}{\eta^t}\widetilde{\Delta}^{(1)} + \frac{1}{\eta}(P3)\big\|\Delta^{(t)}\big\|^2 + \left(\frac{1}{\eta^2}(P3) - \frac{\rho}{2\eta}\right)\big\|\Delta^{(t-1)}\big\|^2 \\
&+ \left(\frac{1}{\eta^3}(P3) - \frac{\rho}{2\eta^2}\right)\big\|\Delta^{(t-2)}\big\|^2 + \cdots \\
&+ \left(\frac{1}{\eta^{t+1}}(P3) - \frac{\rho}{2\eta^t}\right)\big\|\Delta^{(0)}\big\|^2 \\
&+ \Big(\frac{L}{\delta} + \frac{\frac{\delta_1}{2}L^2\tau}{\frac{\rho}{2}(1+\delta_1)+\delta_1}\Big)\frac{\eta^\tau - 1}{\eta - 1}\sum_{l=1}^{t}\frac{1}{\eta^{l+1}}\big\|\Delta^{(t-l)}\big\|^2, \qquad (5.12)
\end{aligned}
$$

where the last inequality is resulted from combining the coefficients of the same items; moreover, by the definition of $\Delta^{(l)}$, we let $\Delta^{(l)} = 0$ for $l < 0$, and use the following fact that

$$
\begin{aligned}
&\sum_{l=0}^{t}\frac{1}{\eta^{l+1}}\sum_{k=1}^{\tau}\big\|\Delta^{(t-l-k)}\big\|^2 \\
&\qquad = \eta^{-t-1}\sum_{l=0}^{t}\frac{\eta^t}{\eta^l}\sum_{k=1}^{\tau}\big\|\Delta^{(t-l-k)}\big\|^2 \\
&\qquad = \eta^{-t-1}\sum_{j=0}^{t}\eta^j\sum_{k=j-\tau}^{j-1}\big\|\Delta^{(k)}\big\|^2
\end{aligned}
$$

which further implies that

$$\begin{aligned}
&\sum_{l=0}^{t}\frac{1}{\eta^{l+1}}\sum_{k=1}^{\tau}\left\|\Delta^{(t-l-k)}\right\|^2\\
&\qquad\overset{(h)}{\leq}\eta^{-t-1}\sum_{j=0}^{t-1}\eta^j(1+\eta+\cdots+\eta^{\tau-1})\left\|\Delta^{(j)}\right\|^2\\
&\qquad\leq\frac{\eta^\tau-1}{\eta-1}\sum_{j=0}^{t-1}\frac{1}{\eta^{t-j+1}}\left\|\Delta^{(j)}\right\|^2\\
&\qquad\leq\frac{\eta^\tau-1}{\eta-1}\sum_{l=1}^{t}\frac{1}{\eta^{l+1}}\left\|\Delta^{(t-l)}\right\|^2,
\end{aligned}$$

where inequality (h) holds because the coefficient of $\left\|\Delta^{(j)}\right\|^2$ in the summation is less than $\eta^j(1+\eta+\cdots+\eta^{\tau-1})$. Therefore if $\rho>0$ satisfies that

$$(P3):=\frac{\delta_1 L+\frac{\rho}{2}(1+\delta_1)}{\frac{\rho}{2}(1+\delta_1)+\delta_1}+\frac{3L}{2}-\frac{\rho}{2}+L\delta\tau<0, \tag{5.13}$$

and

$$\begin{aligned}
&(P3)-\frac{\rho\eta}{2}+\Big(\frac{L}{\delta}+\frac{\frac{\delta_1}{2}L^2\tau}{\frac{\rho}{2}(1+\delta_1)+\delta_1}\Big)\frac{\eta^\tau-1}{\eta-1}\\
&=\frac{\delta_1 L+\frac{\rho}{2}(1+\delta_1)}{\frac{\rho}{2}(1+\delta_1)+\delta_1}+\frac{3L}{2}-\frac{\rho}{2}+L\delta\tau\\
&-\frac{\rho\eta}{2}+\Big(\frac{L}{\delta}+\frac{\frac{\delta_1}{2}L^2\tau}{\frac{\rho}{2}(1+\delta_1)+\delta_1}\Big)\frac{\eta^\tau-1}{\eta-1}<0,
\end{aligned} \tag{5.14}$$

which are the coefficients of the nonnegative terms in (5.12), then inputting (5.13) and (5.14) into (5.12) we have

$$0\leq\widetilde{\Delta}^{(t+1)}\leq\frac{1}{\eta^t}\widetilde{\Delta}^{(1)}.$$

The conclusion is proved. □

5.1.3 Proofs of Lemmata and Additional Results

Proof of Lemma 2.3.1

The conclusion can be proved by the optimality of $\mathbf{x}^{t+1}$ and the convexity assumption. Firstly, using the optimality of $\mathbf{x}^{t+1}$ in the update (2.3), we have

$$\begin{aligned} &-\Big[\nabla \mathbf{L}_1(\mathbf{x}^{t+1}) + \frac{1}{m}\sum_{j=1}^{m} \nabla \mathbf{L}_j(\mathbf{x}^{t_j}) - \nabla \mathbf{L}_1(\mathbf{x}^{t_1})\Big] \\ &\quad \in \partial \left[h(\mathbf{x}) + \tfrac{\rho}{2}\left\|\mathbf{x} - \mathbf{x}^t\right\|^2\right]\Big|_{\mathbf{x}=\mathbf{x}^{t+1}}. \end{aligned} \tag{5.15}$$

Recall that in Assumption 2.3.2 (I), we define the convex modulus of $\frac{\rho}{2}\|\mathbf{x}-\mathbf{x}^t\|^2 + h(\mathbf{x})$ by $\gamma(\rho)$. It follows that

$$\begin{aligned} &\tfrac{\rho}{2}\|\mathbf{x}^{t+1} - \mathbf{x}^t\|^2 + h(\mathbf{x}^{t+1}) - \big(\tfrac{\rho}{2}\|\mathbf{x}^t - \mathbf{x}^t\|^2 + h(\mathbf{x}^t)\big) \\ &\leq \left\langle -\Big[\nabla \mathbf{L}_1(\mathbf{x}^{t+1}) + \frac{1}{m}\sum_{j=1}^{m} \nabla \mathbf{L}_j(\mathbf{x}^{t_j}) - \nabla \mathbf{L}_1(\mathbf{x}^{t_1})\Big], \mathbf{x}^{t+1} - \mathbf{x}^t \right\rangle - \frac{\gamma(\rho)}{2}\|\mathbf{x}^{t+1} - \mathbf{x}^t\|^2 \\ &= -\left\langle \nabla \mathbf{L}_1(\mathbf{x}^{t+1}) + \frac{1}{m}\sum_{j=1}^{m} \nabla \mathbf{L}_j(\mathbf{x}^{t_j}) - \nabla \mathbf{L}_1(\mathbf{x}^{t_1}), \Delta^{(t)} \right\rangle - \frac{\gamma(\rho)}{2}\left\|\Delta^{(t)}\right\|^2, \end{aligned}$$

where we define $\Delta^{(t)} := \mathbf{x}^{t+1} - \mathbf{x}^t$. Therefore

$$\begin{aligned} &\frac{\rho}{2}\|\mathbf{x}^{t+1} - \mathbf{x}^t\|^2 + h(\mathbf{x}^{t+1}) - \frac{\rho}{2}\|\mathbf{x}^t - \mathbf{x}^{t-1}\|^2 - h(\mathbf{x}^t) \\ &= \frac{\rho}{2}\|\mathbf{x}^{t+1} - \mathbf{x}^t\|^2 + h(\mathbf{x}^{t+1}) - \frac{\rho}{2}\|\mathbf{x}^t - \mathbf{x}^t\|^2 - h(\mathbf{x}^t) \\ &+ \frac{\rho}{2}\|\mathbf{x}^t - \mathbf{x}^t\|^2 + h(\mathbf{x}^t) - \frac{\rho}{2}\|\mathbf{x}^t - \mathbf{x}^{t-1}\|^2 - h(\mathbf{x}^t) \\ &\leq -\Big\langle \nabla \mathbf{L}_1(\mathbf{x}^{t+1}) + \frac{1}{m}\sum_{j=1}^{m} \nabla \mathbf{L}_j(\mathbf{x}^{t_j}) - \nabla \mathbf{L}_1(\mathbf{x}^{t_1}), \\ &\Delta^{(t)} \Big\rangle - \frac{\gamma(\rho)}{2}\left\|\Delta^{(t)}\right\|^2 - \frac{\rho}{2}\left\|\Delta^{(t-1)}\right\|^2, \end{aligned} \tag{5.16}$$

proving Lemma 2.3.1.

Proof of Lemma 2.3.2

The proof of this lemma relies on the Lipschitz continuity of $\nabla \mathbf{L}_j(\mathbf{x})$ and Lemma 2.3.1. It follows from Assumption 2.3.1 that $\nabla \mathbf{L}_j(\mathbf{x})$ is Lipschitz continuous with constant L. Therefore we have

$$\begin{aligned}
&\frac{1}{m}\sum_{j\in[m]} \mathbf{L}_j(\mathbf{x}^{t+1}) - \frac{1}{m}\sum_{j\in[m]} \mathbf{L}_j(\mathbf{x}^{t}) \\
&\leq \left\langle \mathbf{x}^{t+1}-\mathbf{x}^t, \frac{1}{m}\sum_{j\in[m]} \nabla \mathbf{L}_j(\mathbf{x}^t) \right\rangle + \frac{L}{2}\|\mathbf{x}^{t+1}-\mathbf{x}^t\|^2 \\
&= \left\langle \Delta^{(t)}, \frac{1}{m}\sum_{j\in[m]} \nabla \mathbf{L}_j(\mathbf{x}^t) \right\rangle + \frac{L}{2}\left\|\Delta^{(t)}\right\|^2. \qquad (5.17)
\end{aligned}$$

By the above definition of function $\mathbf{F}$, combining (5.16) and (5.17) results in

$$\begin{aligned}
&\mathbf{F}(\mathbf{x}^{t+1},\mathbf{x}^t) - \mathbf{F}(\mathbf{x}^t,\mathbf{x}^{t-1}) \\
&\overset{(b)}{\leq} -\left\langle \nabla \mathbf{L}_1(\mathbf{x}^{t+1}) + \frac{1}{m}\sum_{j\in[m]} \nabla \mathbf{L}_j(\mathbf{x}^{t_j}) - \nabla \mathbf{L}_1(\mathbf{x}^{t_1}), \right. \\
&\quad \left. \Delta^{(t)} \right\rangle - \frac{\gamma(\rho)}{2}\left\|\Delta^{(t)}\right\|^2 - \frac{\rho}{2}\left\|\Delta^{(t-1)}\right\|^2 \\
&+ \left\langle \Delta^{(t)}, \frac{1}{m}\sum_{j\in[m]} \nabla \mathbf{L}_j(\mathbf{x}^t) \right\rangle + \frac{L}{2}\left\|\Delta^{(t)}\right\|^2 \\
&= -\left\langle \nabla \mathbf{L}_1(\mathbf{x}^{t+1}) + \frac{1}{m}\sum_{j\in[m]} \nabla \mathbf{L}_j(\mathbf{x}^{t}) - \nabla \mathbf{L}_1(\mathbf{x}^{t}), \right. \\
&\quad \left. \Delta^{(t)} \right\rangle - \frac{\gamma(\rho)}{2}\left\|\Delta^{(t)}\right\|^2 - \frac{\rho}{2}\left\|\Delta^{(t-1)}\right\|^2 \\
&+ \left\langle \Delta^{(t)}, \frac{1}{m}\sum_{j\in[m]} \nabla \mathbf{L}_j(\mathbf{x}^t) \right\rangle + \frac{L}{2}\left\|\Delta^{(t)}\right\|^2
\end{aligned}$$

$$
\begin{aligned}
&+\left\langle \frac{1}{m}\sum_{j\in[m]}\nabla \mathbf{L}_j(\mathbf{x}^t)-\frac{1}{m}\sum_{j\in[m]}\nabla \mathbf{L}_j(\mathbf{x}^{t_j}),\Delta^{(t)}\right\rangle\\
&+\left\langle \nabla \mathbf{L}_1(\mathbf{x}^{t_1})-\nabla \mathbf{L}_1(\mathbf{x}^t),\Delta^{(t)}\right\rangle\\
&\leq\left(\frac{3L}{2}-\frac{\gamma(\rho)}{2}\right)\|\Delta^{(t)}\|^2-\frac{\rho}{2}\|\Delta^{(t-1)}\|^2\\
&+\underbrace{\left\langle \frac{1}{m}\sum_{j\in[m]}\nabla \mathbf{L}_j(\mathbf{x}^t)-\frac{1}{m}\sum_{j\in[m]}\nabla \mathbf{L}_j(\mathbf{x}^{t_j}),\Delta^{(t)}\right\rangle}_{(P1)}\\
&\underbrace{+\left\langle \nabla \mathbf{L}_1(\mathbf{x}^{t_1})-\nabla \mathbf{L}_1(\mathbf{x}^t),\Delta^{(t)}\right\rangle}_{(P1)},
\end{aligned}
\tag{5.18}
$$

where inequality (b) is due to Lemma 2.3.1 and Assumption 2.3.1. Note that

$$
\nabla \mathbf{L}_1(\mathbf{x}^{t_1})-\nabla \mathbf{L}_1(\mathbf{x}^t)=\sum_{k=1}^{t-t_1}\big(\nabla \mathbf{L}_1(\mathbf{x}^{t-k})-\nabla \mathbf{L}_1(\mathbf{x}^{t-k+1})\big),
$$

which implies

$$
\begin{aligned}
\|\nabla \mathbf{L}_1(\mathbf{x}^{t_1})-\nabla \mathbf{L}_1(\mathbf{x}^t)\|&\leq\sum_{k=1}^{t-t_1}\left\|\nabla \mathbf{L}_1(\mathbf{x}^{t-k})-\nabla \mathbf{L}_1(\mathbf{x}^{t-k+1})\right\|\\
&\leq\sum_{k=1}^{t-t_1}L\left\|\mathbf{x}^{t-k}-\mathbf{x}^{t-k+1}\right\|\\
&\leq\sum_{k=1}^{\tau}L\left\|\mathbf{x}^{t-k}-\mathbf{x}^{t-k+1}\right\|\\
&=\sum_{k=1}^{\tau}L\|\Delta^{(t-k)}\|.
\end{aligned}
$$

Similarly, we can see

$$
\left\|\frac{1}{m}\sum_{j\in[m]}\nabla \mathbf{L}_j(\mathbf{x}^t)-\frac{1}{m}\sum_{j\in[m]}\nabla \mathbf{L}_j(\mathbf{x}^{t_j})\right\|\leq\sum_{k=1}^{\tau}L\|\Delta^{(t-k)}\|.
$$

These two inequalities result in

$$\begin{aligned}(P1) &\leq 2L\sum_{k=1}^{\tau}\left\|\Delta^{(t-k)}\right\|\left\|\Delta^{(t)}\right\| \\ &\leq L\sum_{k=1}^{\tau}\left(\frac{1}{\delta}\left\|\Delta^{(t-k)}\right\|^2+\delta\left\|\Delta^{(t)}\right\|^2\right) \\ &\leq \frac{L}{\delta}\sum_{k=1}^{\tau}\left\|\Delta^{(t-k)}\right\|^2+L\delta\tau\left\|\Delta^{(t)}\right\|^2, \end{aligned} \tag{5.19}$$

where $(P1)$ is defined in (5.18) and in the second inequality we apply the fact that

$$a\cdot b\leq\frac{1}{2}\left(\frac{1}{\delta}a^2+\delta b^2\right)$$

for any $a, b, \delta > 0$. By inserting (5.19) into (5.18) we have

$$\begin{aligned}\mathbf{F}(\mathbf{x}^{t+1},\mathbf{x}^t)-\mathbf{F}(\mathbf{x}^t,\mathbf{x}^{t-1}) \leq \left(\frac{3L}{2}-\frac{\gamma(\rho)}{2}+L\delta\tau\right)\left\|\Delta^{(t)}\right\|^2-\frac{\rho}{2}\left\|\Delta^{(t-1)}\right\|^2 \\ +\frac{L}{\delta}\sum_{k=1}^{\tau}\left\|\Delta^{(t-k)}\right\|^2,\end{aligned}$$

proving the conclusion of Lemma 2.3.2.

Proof of Lemma 2.3.3

In this lemma we prove the boundedness of $\mathbf{F}$. First of all, summing the above inequality (2.9) of Lemma 2.3.2 over t yields

$$\begin{aligned}\mathbf{F}(\mathbf{x}^{T+1},\mathbf{x}^T)-\mathbf{F}(\mathbf{x}^1,\mathbf{x}^0) \leq \sum_{t=1}^{T}\left(\frac{3L}{2}-\frac{\gamma(\rho)}{2}+L\delta\tau\right)\left\|\Delta^{(t)}\right\|^2 \\ +\sum_{t=1}^{T}\left(\frac{L\tau}{\delta}-\frac{\rho}{2}\right)\left\|\Delta^{(t-1)}\right\|^2.\end{aligned}$$

If ρ satisfies Assumption 2.3.2, then it holds that

$$\mathbf{F}(\mathbf{x}^{T+1},\mathbf{x}^T)-\mathbf{F}(\mathbf{x}^1,\mathbf{x}^0)<0.$$

By taking $\mathbf{x}^T$ as the initial point, similarly we have

$$\mathbf{F}(\mathbf{x}^{2T+1}, \mathbf{x}^{2T}) - \mathbf{F}(\mathbf{x}^{T+1}, \mathbf{x}^T) < 0.$$

Continuing this process we get a decreasing subsequence $\left\{\mathbf{F}(\mathbf{x}^{kT+1}, \mathbf{x}^{kT})\right\}_{k=0,1,\cdots}$. Therefore there exists a constant $\bar{F}_0$ such that

$$\mathbf{F}(\mathbf{x}^{kT+1}, \mathbf{x}^{kT}) < \bar{F}_0. \tag{5.20}$$

When starting with $\mathbf{x}^1, \cdots, \mathbf{x}^{T-1}$, with similar analysis we can prove that there exists constants $\bar{F}_1, \cdots, \bar{F}_{T-1}$ such that

$$\mathbf{F}(\mathbf{x}^{kT+l+1}, \mathbf{x}^{kT+l}) < \bar{F}_l, \tag{5.21}$$

for $l = 1, 2, \cdots, T-1$. Define $\bar{\mathbf{F}} := \max\left\{\bar{F}_0, \cdots, \bar{F}_{T-1}\right\}$, then

$$\mathbf{F}(\mathbf{x}^{t+1}, \mathbf{x}^t) < \bar{\mathbf{F}} < +\infty, \quad \forall t \in \mathbb{N}.$$

On the other hand, let $\underline{\mathbf{F}} := \underline{L}$, then by the definition of $\mathbf{F}(\mathbf{x}^{t+1}, \mathbf{x}^t)$ and Assumption 2.3.2, we have

$$\begin{aligned}
\mathbf{F}(\mathbf{x}^{t+1}, \mathbf{x}^t) &= \frac{1}{m}\sum_{j\in[m]} \mathbf{L}_j(\mathbf{x}^{t+1}) + \frac{\rho}{2}\left\|\mathbf{x}^{t+1} - \mathbf{x}^t\right\|^2 + h(\mathbf{x}^{t+1}) \\
&\geq \frac{1}{m}\sum_{j\in[m]} \mathbf{L}_j(\mathbf{x}^{t+1}) + h(\mathbf{x}^{t+1}) \\
&= \mathbf{L}(\mathbf{x}^{t+1}) > \underline{L} = \underline{\mathbf{F}} > -\infty,
\end{aligned}$$

for any $t \in \mathbb{N}$. Therefore the boundedness of function $\mathbf{F}$ in Lemma 2.3.3 is proved.

Proof of Lemma 2.3.4

Lemma 2.3.4 gives a bound of $\left(\mathbf{F}(\mathbf{x}^{t+1}, \mathbf{x}^t) - \mathbf{F}(\mathbf{x}^*, \mathbf{x}^*)\right)$ with respect to the distance between $\mathbf{x}^t$ and $\mathbf{x}^{t+1}$ as well as that between $\mathbf{x}^t$ and $\mathbf{x}^{t_j}$. Note that $\left(\mathbf{F}(\mathbf{x}^{t+1}, \mathbf{x}^t) - \mathbf{F}(\mathbf{x}^*, \mathbf{x}^*)\right)$

is nonnegative. The proof of this lemma relies on the strong convexity in Assumption 2.3.3. Firstly, by the optimality of $\mathbf{x}^{t+1}$ in the update (2.3), we have

$$\Big(\nabla \mathbf{L}_1(\mathbf{x}^{t+1}) + \frac{1}{m}\sum_{j\in[m]} \nabla \mathbf{L}_j(\mathbf{x}^{t_j}) - \nabla \mathbf{L}_1(\mathbf{x}^{t_1}) + \partial h(\mathbf{x}^{t+1}) + \rho(\mathbf{x}^{t+1} - \mathbf{x}^t)\Big)^\top (\mathbf{x}^{t+1} - \mathbf{x}) \leq 0, \tag{5.22}$$

for all $\mathbf{x} \in \mathbb{R}^p$. Letting $\mathbf{x} = \mathbf{x}^*$ implies

$$\Big(\nabla \mathbf{L}_1(\mathbf{x}^{t+1}) + \frac{1}{m}\sum_{j\in[m]} \nabla \mathbf{L}_j(\mathbf{x}^{t_j}) - \nabla \mathbf{L}_1(\mathbf{x}^{t_1}) + \partial h(\mathbf{x}^{t+1}) + \rho(\mathbf{x}^{t+1} - \mathbf{x}^t)\Big)^\top (\mathbf{x}^{t+1} - \mathbf{x}^*) \leq 0. \tag{5.23}$$

Note that (5.23) further implies that

$$\Big(\nabla \mathbf{L}_1(\mathbf{x}^{t+1}) - \nabla \mathbf{L}_1(\mathbf{x}^t) + \big(\frac{1}{m}\sum_{j\in[m]} \nabla \mathbf{L}_j(\mathbf{x}^{t_j}) - \frac{1}{m}\sum_{j\in[m]} \nabla \mathbf{L}_j(\mathbf{x}^{t+1})\big) + \frac{1}{m}\sum_{j\in[m]} \nabla \mathbf{L}_j(\mathbf{x}^{t+1}) + \partial h(\mathbf{x}^{t+1}) + \rho(\mathbf{x}^{t+1} - \mathbf{x}^t)\Big)^\top (\mathbf{x}^{t+1} - \mathbf{x}^*) \leq 0. \tag{5.24}$$

By the strong convexity of $\mathbf{L}_j$ one has

$$\mathbf{L}_j(\mathbf{y}) \geq \mathbf{L}_j(\mathbf{x}) + (\nabla \mathbf{L}_j(\mathbf{x}))^\top (\mathbf{y} - \mathbf{x}) + \frac{\sigma^2}{2}\|\mathbf{y} - \mathbf{x}\|^2, \quad \forall \mathbf{x},\ \mathbf{y} \in \mathbb{R}^p. \tag{5.25}$$

Setting $\mathbf{y} = \mathbf{x}^*$, $\mathbf{x} = \mathbf{x}^{t+1}$ in (5.25) we have

$$\mathbf{L}_j(\mathbf{x}^*) \geq \mathbf{L}_j(\mathbf{x}^{t+1}) + \big(\nabla \mathbf{L}_j(\mathbf{x}^{t+1})\big)^\top (\mathbf{x}^* - \mathbf{x}^{t+1}) + \frac{\sigma^2}{2}\|\mathbf{x}^* - \mathbf{x}^{t+1}\|^2,$$

which further implies that

$$\big(\nabla \mathbf{L}_j(\mathbf{x}^{t+1})\big)^\top (\mathbf{x}^{t+1} - \mathbf{x}^*) \geq \mathbf{L}_j(\mathbf{x}^{t+1}) - \mathbf{L}_j(\mathbf{x}^*) + \frac{\sigma^2}{2}\|\mathbf{x}^* - \mathbf{x}^{t+1}\|^2. \tag{5.26}$$

Summing (5.26) over $j \in [m]$ gives that

$$\frac{1}{m}\sum_{j\in[m]}\left(\nabla \mathbf{L}_j(\mathbf{x}^{t+1})\right)^\top(\mathbf{x}^{t+1}-\mathbf{x}^*) \geq \frac{1}{m}\sum_{j\in[m]}\mathbf{L}_j(\mathbf{x}^{t+1}) - \frac{1}{m}\sum_{j\in[m]}\mathbf{L}_j(\mathbf{x}^*) + \frac{\sigma^2}{2}\|\mathbf{x}^*-\mathbf{x}^{t+1}\|^2. \tag{5.27}$$

Putting (5.27) into the above inequality (5.24) that resulted from the optimality, we have

$$\begin{aligned}&\Big(\nabla \mathbf{L}_1(\mathbf{x}^{t+1}) - \nabla \mathbf{L}_1(\mathbf{x}^t) + \Big(\frac{1}{m}\sum_{j\in[m]}\nabla \mathbf{L}_j(\mathbf{x}^{t_j}) - \frac{1}{m}\sum_{j\in[m]}\nabla \mathbf{L}_j(\mathbf{x}^{t+1})\Big)\Big)^\top(\mathbf{x}^{t+1}-\mathbf{x}^*)\\ &+\Big(\frac{1}{m}\sum_{j\in[m]}\mathbf{L}_j(\mathbf{x}^{t+1}) - \frac{1}{m}\sum_{j\in[m]}\mathbf{L}_j(\mathbf{x}^*)\Big)\\ &+\frac{\sigma^2}{2}\|\mathbf{x}^*-\mathbf{x}^{t+1}\|^2 + \partial h(\mathbf{x}^{t+1})(\mathbf{x}^{t+1}-\mathbf{x}^*)\\ &+\rho(\mathbf{x}^{t+1}-\mathbf{x}^t)^\top(\mathbf{x}^{t+1}-\mathbf{x}^*)\leq 0.\end{aligned} \tag{5.28}$$

Since $h(\mathbf{x})$ is convex, we have

$$h(\mathbf{x}^{t+1}) - h(\mathbf{x}^*) \leq \partial h(\mathbf{x}^{t+1})(\mathbf{x}^{t+1}-\mathbf{x}^*).$$

Putting it into (5.28), we have

$$\begin{aligned}&\Big(\nabla \mathbf{L}_1(\mathbf{x}^{t+1}) - \nabla \mathbf{L}_1(\mathbf{x}^t) + \Big(\frac{1}{m}\sum_{j\in[m]}\nabla \mathbf{L}_j(\mathbf{x}^{t_j}) - \frac{1}{m}\sum_{j\in[m]}\nabla \mathbf{L}_j(\mathbf{x}^{t+1})\Big)\Big)^\top(\mathbf{x}^{t+1}-\mathbf{x}^*)\\ &+\Big(\frac{1}{m}\sum_{j\in[m]}\mathbf{L}_j(\mathbf{x}^{t+1}) - \frac{1}{m}\sum_{j\in[m]}\mathbf{L}_j(\mathbf{x}^*)\Big)\\ &+\frac{\sigma^2}{2}\|\mathbf{x}^*-\mathbf{x}^{t+1}\|^2 + h(\mathbf{x}^{t+1}) - h(\mathbf{x}^*)\\ &+\rho(\mathbf{x}^{t+1}-\mathbf{x}^t)^\top(\mathbf{x}^{t+1}-\mathbf{x}^*)\leq 0.\end{aligned} \tag{5.29}$$

Note that

$$\rho(\mathbf{x}^{t+1}-\mathbf{x}^t)^\top(\mathbf{x}^{t+1}-\mathbf{x}^*) = \frac{\rho}{2}\|\mathbf{x}^{t+1}-\mathbf{x}^*\|^2 + \frac{\rho}{2}\|\mathbf{x}^{t+1}-\mathbf{x}^t\|^2 - \frac{\rho}{2}\|\mathbf{x}^t-\mathbf{x}^*\|^2. \tag{5.30}$$

Then putting (5.30) into (5.29), one obtains

$$\begin{aligned}
&\Big(\nabla\mathbf{L}_1(\mathbf{x}^{t+1}) - \nabla\mathbf{L}_1(\mathbf{x}^t) + \Big(\frac{1}{m}\sum_{j\in[m]}\nabla\mathbf{L}_j(\mathbf{x}^{t_j}) - \frac{1}{m}\sum_{j\in[m]}\nabla\mathbf{L}_j(\mathbf{x}^{t+1})\Big)\Big)^\top(\mathbf{x}^{t+1}-\mathbf{x}^*)\\
&+\Big(\frac{1}{m}\sum_{j\in[m]}\mathbf{L}_j(\mathbf{x}^{t+1}) - \frac{1}{m}\sum_{j\in[m]}\mathbf{L}_j(\mathbf{x}^*)\Big)\\
&+\frac{\sigma^2}{2}\|\mathbf{x}^*-\mathbf{x}^{t+1}\|^2 + h(\mathbf{x}^{t+1}) - h(\mathbf{x}^*)\\
&+\frac{\rho}{2}\|\mathbf{x}^{t+1}-\mathbf{x}^t\|^2 + \frac{\rho}{2}\|\mathbf{x}^{t+1}-\mathbf{x}^*\|^2 - \frac{\rho}{2}\|\mathbf{x}^t-\mathbf{x}^*\|^2 \le 0,
\end{aligned}$$

which is equivalent to

$$\begin{aligned}
\mathbf{F}(\mathbf{x}^{t+1},\mathbf{x}^t) - \mathbf{F}(\mathbf{x}^*,\mathbf{x}^*) \le &-\frac{\sigma^2}{2}\|\mathbf{x}^*-\mathbf{x}^{t+1}\|^2 - \Big(\nabla\mathbf{L}_1(\mathbf{x}^{t+1}) - \nabla\mathbf{L}_1(\mathbf{x}^t)\\
&+\Big(\frac{1}{m}\sum_{j\in[m]}\nabla\mathbf{L}_j(\mathbf{x}^{t_j}) - \frac{1}{m}\sum_{j\in[m]}\nabla\mathbf{L}_j(\mathbf{x}^{t+1})\Big)\Big)^\top(\mathbf{x}^{t+1}-\mathbf{x}^*)\\
&-\frac{\rho}{2}\|\mathbf{x}^{t+1}-\mathbf{x}^*\|^2 + \frac{\rho}{2}\|\mathbf{x}^t-\mathbf{x}^*\|^2.
\end{aligned} \tag{5.31}$$

Now, notice that

$$\begin{aligned}
&\Big(\nabla\mathbf{L}_1(\mathbf{x}^{t+1}) - \nabla\mathbf{L}_1(\mathbf{x}^t) + \Big(\frac{1}{m}\sum_{j\in[m]}\nabla\mathbf{L}_j(\mathbf{x}^{t_j}) - \frac{1}{m}\sum_{j\in[m]}\nabla\mathbf{L}_j(\mathbf{x}^{t+1})\Big)\Big)^\top(\mathbf{x}^{t+1}-\mathbf{x}^*)\\
&=\Big(\nabla\mathbf{L}_1(\mathbf{x}^{t+1}) - \nabla\mathbf{L}_1(\mathbf{x}^t) + \frac{1}{m}\sum_{j\in[m]}\nabla\mathbf{L}_j(\mathbf{x}^t)\\
&-\frac{1}{m}\sum_{j\in[m]}\nabla\mathbf{L}_j(\mathbf{x}^{t+1})\Big)^\top(\mathbf{x}^{t+1}-\mathbf{x}^*)\\
&\underbrace{+\Big(\frac{1}{m}\sum_{j\in[m]}\nabla\mathbf{L}_j(\mathbf{x}^{t_j}) - \frac{1}{m}\sum_{j\in[m]}\nabla\mathbf{L}_j(\mathbf{x}^t)\Big)^\top(\mathbf{x}^{t+1}-\mathbf{x}^*)}_{(P5)}.
\end{aligned} \tag{5.32}$$

Given a matrix $\mathbf{H} \in \mathbb{R}^{s\times s}$, we first define $\|\mathbf{v}\|_{\mathbf{H}}^2 := \mathbf{v}^\top \mathbf{H}\mathbf{v}$ for any vector $\mathbf{v} \in \mathbb{R}^s$. Then by using the Mean Value Theorem we have

$$\begin{aligned}
&\left(\nabla \mathbf{L}_1(\mathbf{x}^{t+1}) - \nabla \mathbf{L}_1(\mathbf{x}^t) + \Big(\frac{1}{m}\sum_{j\in[m]} \nabla \mathbf{L}_j(\mathbf{x}^t) - \frac{1}{m}\sum_{j\in[m]} \nabla \mathbf{L}_j(\mathbf{x}^{t+1})\Big)\right)^\top (\mathbf{x}^{t+1} - \mathbf{x}^*) \\
&= (\mathbf{x}^{t+1} - \mathbf{x}^t)^\top \Big[\nabla^2 \mathbf{L}_1(\xi) - \frac{1}{m}\sum_{j\in[m]} \nabla^2 \mathbf{L}_j(\xi)\Big](\mathbf{x}^{t+1} - \mathbf{x}^*) \\
&= \frac{1}{2}\|\mathbf{x}^{t+1} - \mathbf{x}^*\|_\Sigma^2 - \frac{1}{2}\|\mathbf{x}^{t+1} - \mathbf{x}^{t+1}\|_\Sigma^2 + \frac{1}{2}\|\mathbf{x}^t - \mathbf{x}^{t+1}\|_\Sigma^2 - \frac{1}{2}\|\mathbf{x}^t - \mathbf{x}^*\|_\Sigma^2,
\end{aligned}$$

where $\Sigma := \nabla^2 \mathbf{L}_1(\xi) - \frac{1}{m}\sum_{j\in[m]} \nabla^2 \mathbf{L}_j(\xi)$. It follows that

$$\begin{aligned}
&\left(\nabla \mathbf{L}_1(\mathbf{x}^{t+1}) - \nabla \mathbf{L}_1(\mathbf{x}^t) + \Big(\frac{1}{m}\sum_{j\in[m]} \nabla \mathbf{L}_j(\mathbf{x}^t) - \frac{1}{m}\sum_{j\in[m]} \nabla \mathbf{L}_j(\mathbf{x}^{t+1})\Big)\right)^\top (\mathbf{x}^{t+1} - \mathbf{x}^*) \\
&= \frac{1}{2}\|\mathbf{x}^{t+1} - \mathbf{x}^*\|_\Sigma^2 + \frac{1}{2}\|\mathbf{x}^t - \mathbf{x}^{t+1}\|_\Sigma^2 - \frac{1}{2}\|\mathbf{x}^t - \mathbf{x}^{t+1} + \mathbf{x}^{t+1} - \mathbf{x}^*\|_\Sigma^2 \\
&\geq \frac{1}{2}\|\mathbf{x}^{t+1} - \mathbf{x}^*\|_\Sigma^2 + \frac{1}{2}\|\mathbf{x}^t - \mathbf{x}^{t+1}\|_\Sigma^2 \\
&- \frac{1}{2}(1+\delta_1)\|\mathbf{x}^{t+1} - \mathbf{x}^t\|_\Sigma^2 - \frac{1}{2}(1+1/\delta_1)\|\mathbf{x}^{t+1} - \mathbf{x}^*\|_\Sigma^2 \\
&\geq -\frac{1}{2}\delta_1\|\mathbf{x}^t - \mathbf{x}^{t+1}\|_\Sigma^2 - \frac{1}{2\delta_1}\|\mathbf{x}^{t+1} - \mathbf{x}^*\|_\Sigma^2,
\end{aligned} \tag{5.33}$$

for $\delta_1 > 0$. Note that

$$\begin{aligned}
&\frac{\rho}{2}\|\mathbf{x}^t - \mathbf{x}^*\|^2 = \frac{\rho}{2}\|\mathbf{x}^t - \mathbf{x}^{t+1} + \mathbf{x}^{t+1} - \mathbf{x}^*\|^2 \\
&\leq \frac{\rho}{2}(1+\delta_1)\|\mathbf{x}^t - \mathbf{x}^{t+1}\|^2 + \frac{\rho}{2}(1+1/\delta_1)\|\mathbf{x}^{t+1} - \mathbf{x}^*\|^2,
\end{aligned}$$

which implies that

$$\begin{aligned}
&-\frac{\rho}{2}\|\mathbf{x}^{t+1} - \mathbf{x}^*\|^2 + \frac{\rho}{2}\|\mathbf{x}^t - \mathbf{x}^*\|^2 \\
&\leq \frac{\rho}{2}(1+\delta_1)\|\mathbf{x}^t - \mathbf{x}^{t+1}\|^2 + \frac{\rho}{2}\frac{1}{\delta_1}\|\mathbf{x}^{t+1} - \mathbf{x}^*\|^2.
\end{aligned} \tag{5.34}$$

Combining (5.31), (5.32), (5.33), and (5.34), we can bound the optimality gap of the

function $\mathbf{F}$ by

$$\begin{aligned}\mathbf{F}(\mathbf{x}^{t+1},\mathbf{x}^t)-\mathbf{F}(\mathbf{x}^*,\mathbf{x}^*) \le & -\frac{\sigma^2}{2}\|\mathbf{x}^*-\mathbf{x}^{t+1}\|^2 \\ & +\delta_1 L\|\mathbf{x}^{t+1}-\mathbf{x}^t\|^2+\frac{L}{\delta_1}\|\mathbf{x}^{t+1}-\mathbf{x}^*\|^2 \\ & -\frac{\rho}{2}\|\mathbf{x}^{t+1}-\mathbf{x}^*\|^2+\frac{\rho}{2}\|\mathbf{x}^t-\mathbf{x}^*\|^2-(P5) \\ & \le -\frac{\sigma^2}{2}\|\mathbf{x}^*-\mathbf{x}^{t+1}\|^2+\frac{L}{\delta_1}\|\mathbf{x}^{t+1}-\mathbf{x}^*\|^2 \\ & +\frac{\rho}{2\delta_1}\|\mathbf{x}^{t+1}-\mathbf{x}^*\|^2+\delta_1 L\|\mathbf{x}^{t+1}-\mathbf{x}^t\|^2 \\ & +\frac{\rho}{2}(1+\delta_1)\|\mathbf{x}^t-\mathbf{x}^{t+1}\|^2-(P5). \end{aligned} \tag{5.35}$$

Now we bound (P5) on the RHS of (5.35), which is defined in (5.32). For some $\delta_1>0$ one has

$$\begin{aligned}(P5) &\ge -\frac{\delta_1}{2m}\sum_{j\in[m]}\|\nabla\mathbf{L}_j(\mathbf{x}^{t_j})-\nabla\mathbf{L}_j(\mathbf{x}^t)\|^2-\frac{1}{2\delta_1}\|\mathbf{x}^{t+1}-\mathbf{x}^*\|^2 \\ &\ge -\frac{\delta_1}{2m}L^2\sum_{j\in[m]}\|\mathbf{x}^{t_j}-\mathbf{x}^t\|^2-\frac{1}{2\delta_1}\|\mathbf{x}^{t+1}-\mathbf{x}^*\|^2.\end{aligned}$$

Therefore we have

$$\begin{aligned}\mathbf{F}(\mathbf{x}^{t+1},\mathbf{x}^t)-\mathbf{F}(\mathbf{x}^*,\mathbf{x}^*) \le & -\frac{\sigma^2}{2}\|\mathbf{x}^*-\mathbf{x}^{t+1}\|^2+\frac{L}{\delta_1}\|\mathbf{x}^{t+1}-\mathbf{x}^*\|^2 \\ & +\frac{\rho}{2\delta_1}\|\mathbf{x}^{t+1}-\mathbf{x}^*\|^2+\delta_1 L\|\mathbf{x}^{t+1}-\mathbf{x}^t\|^2 \\ & +\frac{\rho}{2}(1+\delta_1)\|\mathbf{x}^t-\mathbf{x}^{t+1}\|^2 \\ & +\frac{\delta_1}{2m}L^2\sum_{j\in[m]}\|\mathbf{x}^{t_j}-\mathbf{x}^t\|^2+\frac{1}{2\delta_1}\|\mathbf{x}^{t+1}-\mathbf{x}^*\|^2. \end{aligned} \tag{5.36}$$

Let $\frac{\sigma^2}{2}>\frac{L}{\delta_1}+\frac{\rho}{2\delta_1}+\frac{1}{2\delta_1}$, i.e., $\sigma^2>\frac{2L}{\delta_1}+\frac{\rho}{\delta_1}+\frac{1}{\delta_1}$, then it follows that

$$\begin{aligned}\mathbf{F}(\mathbf{x}^{t+1},\mathbf{x}^t)-\mathbf{F}(\mathbf{x}^*,\mathbf{x}^*) &\le \delta_1 L\|\mathbf{x}^{t+1}-\mathbf{x}^t\|^2\\ &+\big(-\frac{\sigma^2}{2}+\frac{L}{\delta_1}+\frac{\rho}{2\delta_1}+\frac{1}{2\delta_1}\big)\|\mathbf{x}^*-\mathbf{x}^{t+1}\|^2\\ &+\frac{\rho}{2}(1+\delta_1)\|\mathbf{x}^t-\mathbf{x}^{t+1}\|^2+\frac{\delta_1}{2m}L^2\sum_{j\in[m]}\|\mathbf{x}^{t_j}-\mathbf{x}^t\|^2\\ &\le \delta_1 L\|\mathbf{x}^{t+1}-\mathbf{x}^t\|^2+\frac{\rho}{2}(1+\delta_1)\|\mathbf{x}^t-\mathbf{x}^{t+1}\|^2\\ &+\frac{\delta_1}{2m}L^2\sum_{j\in[m]}\|\mathbf{x}^{t_j}-\mathbf{x}^t\|^2, \end{aligned}\tag{5.37}$$

from which we have

$$\begin{aligned}\frac{1}{\frac{\rho}{2}(1+\delta_1)+\delta_1}\big(\mathbf{F}(\mathbf{x}^{t+1},\mathbf{x}^t)-\mathbf{F}(\mathbf{x}^*,\mathbf{x}^*)\big) &\le \frac{\delta_1 L}{\frac{\rho}{2}(1+\delta_1)+\delta_1}\|\mathbf{x}^t-\mathbf{x}^{t+1}\|^2\\ &+\frac{\frac{\rho}{2}(1+\delta_1)}{\frac{\rho}{2}(1+\delta_1)+\delta_1}\|\mathbf{x}^t-\mathbf{x}^{t+1}\|^2\\ &+\frac{1}{\frac{\rho}{2}(1+\delta_1)+\delta_1}\frac{\delta_1}{2m}L^2\sum_{j\in[m]}\|\mathbf{x}^{t_j}-\mathbf{x}^t\|^2,\end{aligned}$$

therefore proving Lemma 2.3.4.

5.1.4 Proof of Corollary 2.3.1 and Corollary 2.3.2

We first prove Corollary 2.3.1. Following the proof steps in Lemma 2.3.1, we have

$$\begin{aligned}&\frac{\rho}{2}\|\mathbf{x}^{t+1}-\mathbf{x}^t\|^2+h(\mathbf{x}^{t+1})-\frac{\rho}{2}\|\mathbf{x}^t-\mathbf{x}^{t-1}\|^2-h(\mathbf{x}^t)\\ &\le -\Big\langle\nabla\mathbf{L}_1(\mathbf{x}^{t+1})+\frac{1}{m}\sum_{j=1}^m\nabla\mathbf{L}_j(\mathbf{x}^{t_j})-\nabla\mathbf{L}_1(\mathbf{x}^{t_1})-\epsilon^t,\\ &\Delta^{(t)}\Big\rangle-\frac{\gamma(\rho)}{2}\big\|\Delta^{(t)}\big\|^2-\frac{\rho}{2}\big\|\Delta^{(t-1)}\big\|^2.\end{aligned}$$

In the second step, for the descent of function $\mathbf{F}$, similar to Lemma 2.3.2 for any $\delta>0$ it holds that

$$\mathbf{F}(\mathbf{x}^{t+1},\mathbf{x}^t)-\mathbf{F}(\mathbf{x}^t,\mathbf{x}^{t-1})\leq\left(\frac{3L}{2}-\frac{\gamma(\rho)}{2}+L\delta\tau\right)\left\|\Delta^{(t)}\right\|^2-\frac{\rho}{2}\left\|\Delta^{(t-1)}\right\|^2 +\frac{L}{\delta}\sum_{k=1}^{\tau}\left\|\Delta^{(t-k)}\right\|^2+\langle\epsilon^t,\Delta^{(t)}\rangle. \tag{5.38}$$

Now following the similar path, the first conclusion of Theorem 2.3.1 can be proved. Summing up (5.38) over t yields

$$\begin{aligned}\mathbf{F}(\mathbf{x}^{T+1},\mathbf{x}^T)-\mathbf{F}(\mathbf{x}^1,\mathbf{x}^0)&=\mathbf{F}(\mathbf{x}^{T+1},\mathbf{x}^T)-\mathbf{F}(\mathbf{x}^T,\mathbf{x}^{T-1})+\mathbf{F}(\mathbf{x}^T,\mathbf{x}^{T-1})\\&\quad-\mathbf{F}(\mathbf{x}^{T-1},\mathbf{x}^{T-2})+\cdots\cdots+\mathbf{F}(\mathbf{x}^2,\mathbf{x}^1)-\mathbf{F}(\mathbf{x}^1,\mathbf{x}^0)\\&\leq\sum_{t=1}^{T}\Big(\frac{3L}{2}-\frac{\gamma(\rho)}{2}+L\delta\tau\Big)\left\|\Delta^{(t)}\right\|^2\\&\quad+\sum_{t=1}^{T}\Big(\frac{L\tau}{\delta}-\frac{\rho}{2}\Big)\left\|\Delta^{(t-1)}\right\|^2+\sum_{t=1}^{T}\langle\epsilon^t,\Delta^{(t)}\rangle,\end{aligned}$$

which further implies that

$$\begin{aligned}\mathbf{F}(\mathbf{x}^{T+1},\mathbf{x}^T)-\mathbf{F}(\mathbf{x}^1,\mathbf{x}^0)&\leq\sum_{t=1}^{T}\left(\frac{3L}{2}-\frac{\gamma(\rho)}{2}+L\delta\tau\right)\left\|\Delta^{(t)}\right\|^2\\&\quad+\sum_{t=1}^{T}\left(\frac{L\tau}{\delta}-\frac{\rho}{2}\right)\left\|\Delta^{(t-1)}\right\|^2+\frac{1}{2}\sum_{t=1}^{T}(\|\epsilon^t\|^2+\left\|\Delta^{(t)}\right\|^2)\\&\leq\sum_{t=1}^{T}\left(\frac{3L}{2}-\frac{\gamma(\rho)}{2}+L\delta\tau+\frac{1}{2}\right)\left\|\Delta^{(t)}\right\|^2\\&\quad+\sum_{t=1}^{T}\left(\frac{L\tau}{\delta}-\frac{\rho}{2}+\frac{1}{2}c_1\right)\left\|\Delta^{(t-1)}\right\|^2,\end{aligned}$$

where in the last inequality we use Assumption 2.3.4. This inequality bounds the cumulative progress of function $\mathbf{F}$ by the weighted summation of the norm of $\Delta^{(t)}$ for $t=1,\ldots,T$. To simplify the above inequality, we define $\widetilde{c}:=\min\big\{\frac{\gamma(\rho)}{2}-\frac{3L}{2}-L\delta\tau-$

$\frac{1}{2}, \frac{\rho}{2} - \frac{L\tau}{\delta} - \frac{1}{2}c_1\Big\}$. Assume that

$$\gamma(\rho) > 3L + 2L\delta\tau + 1 \quad \text{and} \quad \rho > \frac{2L\tau}{\delta} + c_1, \tag{5.39}$$

then we have $\widetilde{c} > 0$. Therefore

$$\mathbf{F}(\mathbf{x}^{T+1}, \mathbf{x}^T) - \mathbf{F}(\mathbf{x}^1, \mathbf{x}^0) \leq -\widetilde{c}\sum_{t=0}^{T} \left\|\Delta^{(t)}\right\|^2. \tag{5.40}$$

Note that by Lemma 2.3.3 the LHS of (5.40) is bounded from below. It follows that

$$\left\|\Delta^{(t)}\right\| \to 0, \qquad t \to \infty.$$

We now establish the second conclusion of Theorem 2.3.1. From (2.15) we know that

$$\mathbf{x}^{t+1} = \mathbf{Prox}_h\Big[\mathbf{x}^{t+1} - \Big(\nabla \mathbf{L}_1(\mathbf{x}^{t+1}) + \frac{1}{m}\sum_{j\in[m]} \nabla \mathbf{L}_j(\mathbf{x}^{t_j}) - \nabla \mathbf{L}_1(\mathbf{x}^{t_1}) + \rho\left(\mathbf{x}^{t+1} - \mathbf{x}^t\right) + \epsilon^t\Big)\Big].$$

This equation implies that

$$\begin{aligned}
&\left\|\mathbf{x}^t - \mathbf{Prox}_h\left(\mathbf{x}^t - \frac{1}{m}\sum_{j\in[m]} \nabla \mathbf{L}_j(\mathbf{x}^t)\right)\right\| \\
&\leq \left\|\mathbf{x}^t - \mathbf{x}^{t+1} + \mathbf{x}^{t+1} - \mathbf{Prox}_h\left(\mathbf{x}^t - \frac{1}{m}\sum_{j\in[m]} \nabla \mathbf{L}_j(\mathbf{x}^t)\right)\right\| \\
&\leq \left\|\mathbf{x}^t - \mathbf{x}^{t+1}\right\| \\
&+ \left\|\mathbf{Prox}_h\Big[\mathbf{x}^{t+1} - \Big(\nabla \mathbf{L}_1(\mathbf{x}^{t+1}) + \frac{1}{m}\sum_{j\in[m]} \nabla \mathbf{L}_j(\mathbf{x}^{t_j}) \right. \right. \\
&\left. \left. - \nabla \mathbf{L}_1(\mathbf{x}^{t_1}) + \rho\big(\mathbf{x}^{t+1} - \mathbf{x}^t\big) - \epsilon^t\Big)\Big] - \mathbf{Prox}_h\left(\mathbf{x}^t - \frac{1}{m}\sum_{j\in[m]} \nabla \mathbf{L}_j(\mathbf{x}^t)\right)\right\|
\end{aligned}$$

$$
\begin{aligned}
&\overset{(\widetilde{a})}{\leq} \Bigg\| (1+\rho)(\mathbf{x}^{t+1}-\mathbf{x}^t) - \epsilon^t + \frac{1}{m}\sum_{j\in[m]} \nabla \mathbf{L}_j(\mathbf{x}^t) \\
&- \frac{1}{m}\sum_{j\in[m]} \nabla \mathbf{L}_j(\mathbf{x}^{t_j}) - \big(\nabla \mathbf{L}_1(\mathbf{x}^{t+1}) - \nabla \mathbf{L}_1(\mathbf{x}^{t_1})\big) \Bigg\| + \big\|\Delta^{(t)}\big\| \\
&\leq (2+\rho)\big\|\Delta^{(t)}\big\| + 2L\sum_{k=0}^{\tau} \big\|\Delta^{(t-k)}\big\| + c_1^{\frac{1}{2}} \big\|\Delta^{(t-1)}\big\| \\
&\longrightarrow 0, \qquad t\to\infty.
\end{aligned} \tag{5.41}
$$

Note that here inequality $(\widetilde{a})$ holds because of the nonexpansiveness of the operator $\mathbf{Prox}_h$. The last inequality follows from Assumption 2.3.1. Therefore as in the proof of Theorem 2.3.1, the second result holds. The rest of the analysis is the same as that of Theorem 2.3.1. Specifically, we can bound the norm of the proximal gradient by the cumulative progress of function $\mathbf{F}$:

$$
\sum_{t=0}^{T} \Big\| \widetilde{\nabla} \mathbf{L}\left(\mathbf{x}^t\right) \Big\|^2 \leq \frac{\widetilde{\mu}}{\widetilde{c}} \left(\mathbf{F}(\mathbf{x}^1, \mathbf{x}^0) - \mathbf{F}(\mathbf{x}^{T+1}, \mathbf{x}^T) \right),
$$

where $\widetilde{\mu} := 3\left((2+\rho)^2 + 4L^2\tau + c_1\right)$.
Recall that $T(\epsilon) := \min\Big\{ t \mid \Big\| \widetilde{\nabla} \mathbf{L}(\mathbf{x}^t) \Big\|^2 \leq \epsilon, t \geq 0 \Big\}$. Thus it follows that

$$
\epsilon \leq \frac{C \cdot \big(\mathbf{F}(\mathbf{x}^1, \mathbf{x}^0) - \underline{\mathbf{F}} \big)}{T(\epsilon)},
$$

where $C := \frac{\widetilde{\mu}}{\widetilde{c}} > 0$. The conclusions in Corollary 2.3.1 have been proved.

Now we prove Corollary 2.3.2. Similar to Lemma 2.3.4, the optimality gap of function $\mathbf{F}$ can be bounded as the following:

$$
\begin{aligned}
&\frac{1}{\frac{\rho}{2}(1+\delta_1)+\delta_1} \left(\mathbf{F}(\mathbf{x}^{t+1}, \mathbf{x}^t) - \mathbf{F}(\mathbf{x}^*, \mathbf{x}^*) \right) \\
&\leq \frac{\delta_1 L}{\frac{\rho}{2}(1+\delta_1)+\delta_1} \|\mathbf{x}^t - \mathbf{x}^{t+1}\|^2 + \frac{\frac{\rho}{2}(1+\delta_1)}{\frac{\rho}{2}(1+\delta_1)+\delta_1} \|\mathbf{x}^t - \mathbf{x}^{t+1}\|^2 \\
&+ \frac{1}{\frac{\rho}{2}(1+\delta_1)+\delta_1} \frac{\delta_1}{2m} L^2 \sum_{j\in[m]} \|\mathbf{x}^{t_j} - \mathbf{x}^t\|^2 + \big(\frac{1}{2}\|\epsilon^t\|^2 + \frac{1}{2}\big\|\Delta^{(t)}\big\|^2\big) \frac{1}{\frac{\rho}{2}(1+\delta_1)+\delta_1}.
\end{aligned} \tag{5.42}
$$

Following the steps of Lemma 2.3.2, we have

$$\mathbf{F}(\mathbf{x}^{t+1},\mathbf{x}^{t})-\mathbf{F}(\mathbf{x}^{t},\mathbf{x}^{t-1})\leq\left(\frac{3L}{2}-\frac{\gamma(\rho)}{2}+L\delta\tau\right)\left\|\Delta^{(t)}\right\|^{2}-\frac{\rho}{2}\left\|\Delta^{(t-1)}\right\|^{2}+\frac{L}{\delta}\sum_{k=1}^{\tau}\left\|\Delta^{(t-k)}\right\|^{2}+\langle\epsilon^{t},\Delta^{(t)}\rangle. \tag{5.43}$$

Summing (5.42) and (5.43) and then applying the Assumption 2.3.4 leads to

$$\begin{aligned}\left(1+\frac{1}{\frac{\rho}{2}(1+\delta_1)+\delta_1}\right)\widetilde{\Delta}^{(t+1)}\leq\;&\widetilde{\Delta}^{(t)}-\frac{\rho-c_1}{2}\left\|\Delta^{(t-1)}\right\|^{2}+\frac{L}{\delta}\sum_{k=1}^{\tau}\left\|\Delta^{(t-k)}\right\|^{2}\\&+\Big[\frac{\delta_1 L+\frac{\rho}{2}(1+\delta_1)+\frac{1}{2}}{\frac{\rho}{2}(1+\delta_1)+\delta_1}+\frac{3L}{2}-\frac{\rho-1}{2}+L\delta\tau\Big]\left\|\Delta^{(t)}\right\|^{2}\\&+\frac{1}{\frac{\rho}{2}(1+\delta_1)+\delta_1}\frac{\delta_1}{2m}L^{2}\sum_{j\in[m]}\|\mathbf{x}^{t_j}-\mathbf{x}^{t}\|^{2}\\&+\frac{c_1}{\frac{\rho}{2}(1+\delta_1)+\delta_1}\left\|\Delta^{(t-1)}\right\|^{2}.\end{aligned}$$

By a recursive argument similar to the proof of Theorem 2.3.2, one can prove that if $\rho>0$ satisfies that

$$\frac{\delta_1 L+\frac{\rho}{2}(1+\delta_1)+\frac{1}{2}}{\frac{\rho}{2}(1+\delta_1)+\delta_1}+\frac{3L}{2}-\frac{\rho-1}{2}+L\delta\tau<0 \tag{5.44}$$

and

$$\begin{aligned}&\frac{\delta_1 L+\frac{\rho}{2}(1+\delta_1)+\frac{1}{2}}{\frac{\rho}{2}(1+\delta_1)+\delta_1}+\frac{3L}{2}-\frac{\rho-1}{2}+L\delta\tau-\frac{(\rho-c_1)\eta}{2}\\&+\left(\frac{L}{\delta}+\frac{\frac{\delta_1}{2}L^{2}\tau}{\frac{\rho}{2}(1+\delta_1)+\delta_1}\right)\frac{\eta^{\tau}-1}{\eta-1}+\frac{c_1\eta}{\frac{\rho}{2}(1+\delta_1)+\delta_1}<0,\end{aligned} \tag{5.45}$$

then we have

$$0\leq\widetilde{\Delta}^{(t+1)}\leq\frac{1}{\eta^{t}}\widetilde{\Delta}^{(1)},$$

the corollary is proved.

5.2 Analysis Details for Chapter 3

5.2.1 Proof of Lemma 3.3.3

First, we define

$$\mathbf{J}_p(\underline{\boldsymbol{\theta}}^k, \underline{\mathbf{w}}^k) := \big[J_{1,p}(\boldsymbol{\theta}_1^k, \mathbf{w}_1^k), \cdots, J_{N,p}(\boldsymbol{\theta}_N^k, \mathbf{w}_N^k)\big].$$

Then the update of $\mathbf{s}_i^k$ and $\boldsymbol{\theta}_i^k$ in (3.14) and (3.16) for $i \in \{1, \cdots, N\}$ can be written in a compact form:

$$\underline{\mathbf{s}}^k = \underline{\mathbf{s}}^{k-1}\mathbf{C}_1^\top + \frac{1}{M}\big[\nabla_{\underline{\boldsymbol{\theta}}}\mathbf{J}_{p_k}(\underline{\boldsymbol{\theta}}^k, \underline{\mathbf{w}}^k) - \nabla_{\underline{\boldsymbol{\theta}}}\mathbf{J}_{p_k}(\underline{\boldsymbol{\theta}}^{\tau_{p_k}^{k-1}}, \underline{\mathbf{w}}^{\tau_{p_k}^{k-1}})\big]\mathbf{C}_2^\top \tag{5.46}$$

$$\underline{\boldsymbol{\theta}}^{k+1} = \underline{\boldsymbol{\theta}}^k\mathbf{R}_1^\top - \gamma_1\underline{\mathbf{s}}^k\mathbf{R}_2^\top. \tag{5.47}$$

Moreover, we define the weighted average of $\underline{\boldsymbol{\theta}}^{k+1}$ over columns by $\bar{\boldsymbol{\theta}}^{k+1} := \frac{1}{N}\underline{\boldsymbol{\theta}}^{k+1}\mathbf{u}$. Then by the definition of $\mathbf{u}$ we have

$$\begin{aligned}\bar{\boldsymbol{\theta}}^{k+1} &:= \frac{1}{N}(\underline{\boldsymbol{\theta}}^k\mathbf{R}_1^\top - \gamma_1\underline{\mathbf{s}}^k\mathbf{R}_2^\top)\mathbf{u}\\ &= \bar{\boldsymbol{\theta}}^k - \frac{\gamma_1}{N}\underline{\mathbf{s}}^k\mathbf{R}_2^\top\mathbf{u}\\ &= \bar{\boldsymbol{\theta}}^k - \frac{\gamma_1}{N}\underline{\mathbf{s}}^k\mathbf{u},\end{aligned} \tag{5.48}$$

where the last two equalities use Assumption 3.3.3. From (5.48) and (5.47) we have

$$\begin{aligned}\underline{\boldsymbol{\theta}}^{k+1} - \bar{\boldsymbol{\theta}}^{k+1}\mathbf{1}^\top &= (\underline{\boldsymbol{\theta}}^k\mathbf{R}_1^\top - \gamma_1\underline{\mathbf{s}}^k\mathbf{R}_2^\top) - \big(\bar{\boldsymbol{\theta}}^k - \frac{\gamma_1}{N}\underline{\mathbf{s}}^k\mathbf{u}\big)\mathbf{1}^\top\\ &= (\underline{\boldsymbol{\theta}}^k - \bar{\boldsymbol{\theta}}^k\mathbf{1}^\top)\mathbf{R}_1^\top - \gamma_1\underline{\mathbf{s}}^k\big(\mathbf{R}_2 - \frac{\mathbf{1}\mathbf{u}^\top}{N}\big)^\top\\ &= (\underline{\boldsymbol{\theta}}^k - \bar{\boldsymbol{\theta}}^k\mathbf{1}^\top)\big(\mathbf{R}_1 - \frac{\mathbf{1}\mathbf{u}^\top}{N}\big)^\top\\ &\quad - \gamma_1(\underline{\mathbf{s}}^k - \bar{\mathbf{s}}^k\mathbf{1}^\top)\big(\mathbf{R}_2 - \frac{\mathbf{1}\mathbf{u}^\top}{N}\big)^\top.\end{aligned} \tag{5.49}$$

Similarly, the average of $\underline{\mathbf{s}}^{k+1}$ is defined as $\bar{\mathbf{s}}^{k+1} := \frac{1}{N}\underline{\mathbf{s}}^{k+1}\mathbf{1}$. Thus we have

$$\begin{aligned}
\underline{\mathbf{s}}^{k+1} - \bar{\mathbf{s}}^{k+1}\mathbf{v}^\top &= \big(\frac{1}{M}\big[\nabla_{\underline{\boldsymbol{\theta}}}\mathbf{J}_{p_{k+1}}(\underline{\boldsymbol{\theta}}^{k+1}, \underline{\mathbf{w}}^{k+1}) - \nabla_{\underline{\boldsymbol{\theta}}}\mathbf{J}_{p_{k+1}}(\underline{\boldsymbol{\theta}}^{\tau^k_{p_{k+1}}}, \underline{\mathbf{w}}^{\tau^k_{p_{k+1}}})\big]\big) \\
&\quad \cdot \big(\mathbf{C}_2 - \frac{\mathbf{v}\mathbf{1}^\top}{N}\big)^\top + \underline{\mathbf{s}}^k\mathbf{C}_1^\top - \bar{\mathbf{s}}^k\mathbf{v}^\top \\
&= \big(\frac{1}{M}\big[\nabla_{\underline{\boldsymbol{\theta}}}\mathbf{J}_{p_{k+1}}(\underline{\boldsymbol{\theta}}^{k+1}, \underline{\mathbf{w}}^{k+1}) - \nabla_{\underline{\boldsymbol{\theta}}}\mathbf{J}_{p_{k+1}}(\underline{\boldsymbol{\theta}}^{\tau^k_{p_{k+1}}}, \underline{\mathbf{w}}^{\tau^k_{p_{k+1}}})\big]\big) \\
&\quad \cdot \big(\mathbf{C}_2 - \frac{\mathbf{v}\mathbf{1}^\top}{N}\big)^\top + (\underline{\mathbf{s}}^k - \bar{\mathbf{s}}^k\mathbf{v}^\top)\big(\mathbf{C}_1 - \frac{\mathbf{v}\mathbf{1}^\top}{N}\big)^\top. \qquad (5.50)
\end{aligned}$$

Recall that we define σ_{R_i} as the spectral radii of $(R_i - \mathbf{1}\mathbf{u}^\top/N)$, for $i = 1, 2$. Then by (5.49), we have

$$\begin{aligned}
\|\underline{\boldsymbol{\theta}}^{k+1} - \bar{\boldsymbol{\theta}}^{k+1}\mathbf{1}^\top\| &\le \sigma_{\mathbf{R}_1}\|\underline{\boldsymbol{\theta}}^k - \bar{\boldsymbol{\theta}}^k\mathbf{1}^\top\| + \gamma_1\sigma_{\mathbf{R}_2}\|\underline{\mathbf{s}}^k - \bar{\mathbf{s}}^k\mathbf{1}^\top\| \\
&\le \sigma_{\mathbf{R}_1}\|\underline{\boldsymbol{\theta}}^k - \bar{\boldsymbol{\theta}}^k\mathbf{1}^\top\| + \gamma_1\sigma_{\mathbf{R}_2}\|\underline{\mathbf{s}}^k - \bar{\mathbf{s}}^k\mathbf{v}^\top\| + \gamma_1\sigma_{\mathbf{R}_2}\|\ \mathbf{v} - \mathbf{1}\|\|\bar{\mathbf{s}}^k\| \\
&\le \sigma_{\mathbf{R}_1}\|\underline{\boldsymbol{\theta}}^k - \bar{\boldsymbol{\theta}}^k\mathbf{1}^\top\| + \gamma_1\sigma_{\mathbf{R}_2}\|\underline{\mathbf{s}}^k - \bar{\mathbf{s}}^k\mathbf{v}^\top\| + \gamma_1\sigma_{\mathbf{R}_2}\|\ \mathbf{v} - \mathbf{1}\| \\
&\quad \cdot \Big\|\bar{\mathbf{s}}^k - \frac{1}{NM}\sum_{i=1}^N\sum_{p=1}^M \nabla_{\boldsymbol{\theta}} J_{i,p}(\bar{\boldsymbol{\theta}}^{\tau_p^k}, \mathbf{w}_i^{\tau_p^k})\Big\| + \gamma_1\sigma_{\mathbf{R}_2}\|\ \mathbf{v} - \mathbf{1}\| \\
&\quad \cdot \Big\|\frac{1}{NM}\sum_{i=1}^N\sum_{p=1}^M \nabla_{\boldsymbol{\theta}} J_{i,p}(\bar{\boldsymbol{\theta}}^{\tau_p^k}, \mathbf{w}_i^{\tau_p^k}) - \frac{1}{NM}\sum_{i=1}^N\sum_{p=1}^M \nabla_{\boldsymbol{\theta}} J_{i,p}(\boldsymbol{\theta}^*, \mathbf{w}_i^*)\Big\| \\
&\le \Big(\sigma_{\mathbf{R}_1} + \gamma\sigma_{\mathbf{R}_2}\|\mathbf{v} - \mathbf{1}\|\frac{\rho}{\sqrt{N}}\Big) \max_{(k-M)_+ \le s \le k}\|\underline{\boldsymbol{\theta}}^s - \bar{\boldsymbol{\theta}}^s\mathbf{1}^\top\| \\
&\quad + \gamma\sigma_{\mathbf{R}_2}\|\underline{\mathbf{s}}^k - \bar{\mathbf{s}}^k\mathbf{v}^\top\| + \gamma\sigma_{\mathbf{R}_2}\|\ \mathbf{v} - \mathbf{1}\|\rho \max_{(k-M)_+ \le s \le k}\|\bar{\boldsymbol{\theta}}^s - \boldsymbol{\theta}^*\| \\
&\quad + \gamma\sigma_{\mathbf{R}_2}\|\ \mathbf{v} - \mathbf{1}\|\bar{A} \cdot \max_{(k-M)_+ \le s \le k}\frac{1}{N}\sum_{i=1}^N\|\mathbf{w}_i^s - \mathbf{w}_i^*\|. \qquad (5.51)
\end{aligned}$$

Similarly, we define σ_{C_i} as the spectral radii of $(C_i - \mathbf{v}\mathbf{1}^\top/N)$, for $i = 1, 2$. Then it follows from this definition and (5.50) that

$$
\begin{aligned}
\|\underline{\mathbf{s}}^{k+1} - \bar{\mathbf{s}}^{k+1}\mathbf{v}^\top\| &\leq \sigma_{\mathbf{C}_1}\|\underline{\mathbf{s}}^{k} - \bar{\mathbf{s}}^{k}\mathbf{v}^\top\| \\
&\quad + \sigma_{\mathbf{C}_2}\frac{1}{M}\big(\rho\|\underline{\boldsymbol{\theta}}^{k+1} - \underline{\boldsymbol{\theta}}^{\tau^k_{p_{k+1}}}\| + \bar{A}\|\underline{\mathbf{w}}^{k+1} - \underline{\mathbf{w}}^{\tau^k_{p_{k+1}}}\|\big) \\
&\leq \sigma_{\mathbf{C}_1}\|\underline{\mathbf{s}}^{k} - \bar{\mathbf{s}}^{k}\mathbf{v}^\top\| \\
&\quad + \sigma_{\mathbf{C}_2}\frac{1}{M}\sum_{l=\tau^k_{p_{k+1}}}^{k}\big(\rho\|\underline{\boldsymbol{\theta}}^{l+1} - \underline{\boldsymbol{\theta}}^{l}\| + \bar{A}\|\underline{\mathbf{w}}^{l+1} - \underline{\mathbf{w}}^{l}\|\big) \\
&\leq \sigma_{\mathbf{C}_1}\|\underline{\mathbf{s}}^{k} - \bar{\mathbf{s}}^{k}\mathbf{v}^\top\| \\
&\quad + \sigma_{\mathbf{C}_2}\frac{1}{M}\sum_{l=(k-M)_+}^{k}\big(\rho\|(\underline{\boldsymbol{\theta}}^{l}(\mathbf{R}_1 - \mathbf{I})^\top) - \gamma_1\underline{\mathbf{s}}^{l}\mathbf{R}_2^\top\| + \gamma_2\bar{A}\|\underline{\mathbf{d}}^{l}\|\big),
\end{aligned}
$$

where the last inequality uses the update equations (3.16)-(3.17) of the primal dual variables. Thus we have

$$
\begin{aligned}
&\|\underline{\mathbf{s}}^{k+1} - \bar{\mathbf{s}}^{k+1}\mathbf{v}^\top\| \\
&\leq \sigma_{\mathbf{C}_1}\|\underline{\mathbf{s}}^{k} - \bar{\mathbf{s}}^{k}\mathbf{v}^\top\| \\
&+ \sigma_{\mathbf{C}_2}\frac{1}{M}\big(\rho\sum_{l=(k-M)_+}^{k}\|(\underline{\boldsymbol{\theta}}^{l} - \bar{\boldsymbol{\theta}}^{l}\mathbf{1}^\top)(\mathbf{R}_1 - \mathbf{I})^\top\| + \gamma\|\underline{\mathbf{s}}^{l}\mathbf{R}_2^\top\| + \gamma_2\bar{A}\|\underline{\mathbf{d}}^{l}\|\big) \\
&\overset{(a)}{\leq} \sigma_{\mathbf{C}_1}\|\underline{\mathbf{s}}^{k} - \bar{\mathbf{s}}^{k}\mathbf{v}^\top\| + \sigma_{\mathbf{C}_2}\frac{1}{M}\Big(\rho\sum_{l=(k-M)_+}^{k}\|(\underline{\boldsymbol{\theta}}^{l} - \bar{\boldsymbol{\theta}}^{l}\mathbf{1}^\top)(\mathbf{R}_1 - \mathbf{I})^\top\| \\
&+ \gamma\rho\|(\underline{\mathbf{s}}^{l} - \bar{\mathbf{s}}^{l}\mathbf{v}^\top)\mathbf{R}_2^\top\| + \gamma\rho\|\mathbf{R}_2\mathbf{v}\|\|\bar{\mathbf{s}}^{l}\| + \gamma_2\bar{A}\|\underline{\mathbf{d}}^{l} - \mathbf{d}^*\|\Big) \\
&\overset{(b)}{\leq} \sigma_{\mathbf{C}_1}\|\underline{\mathbf{s}}^{k} - \bar{\mathbf{s}}^{k}\mathbf{v}^\top\| + \sigma_{\mathbf{C}_2}\frac{1}{M}\Big(\rho\sum_{l=(k-M)_+}^{k}\big(\|(\underline{\boldsymbol{\theta}}^{l} - \bar{\boldsymbol{\theta}}^{l}\mathbf{1}^\top)(\mathbf{R}_1 - \mathbf{I})^\top\| \\
&+ \gamma\rho\|(\underline{\mathbf{s}}^{l} - \bar{\mathbf{s}}^{l}\mathbf{v}^\top)\mathbf{R}_2^\top\| + \gamma\rho\|\mathbf{R}_2\mathbf{v}\|\|\bar{\mathbf{s}}^{l} - \mathbf{s}^*\| + \gamma_2\bar{A}^2\frac{1}{M}\sum_{p=1}^{M}\|\underline{\boldsymbol{\theta}}^{\tau^l_p} - \boldsymbol{\theta}^*\mathbf{1}^\top\| \\
&+ \gamma_2\bar{A}\bar{D}\frac{1}{M}\sum_{p=1}^{M}\|\underline{\mathbf{w}}^{\tau^l_p} - \underline{\mathbf{w}}^*\|\big)\Big),
\end{aligned}
$$

where in the last inequality we use the Lipschitz property of the dual gradient surrogate

$\underline{\mathbf{d}}^l$ with respect to ($w.r.t.$) the primal and dual variables. Note that inequalities (a) and (b) follow from the optimality condition of the problem (3.12) with respect to $\mathbf{w}_i$ for $i \in [N]$:

$$\mathbf{d}^* := \begin{bmatrix} \frac{1}{M}\sum_{i=1}^{M} \nabla_{\mathbf{w}_1} J_{1,p}(\boldsymbol{\theta}^*, \mathbf{w}_1^*) \\ \vdots \\ \frac{1}{M}\sum_{i=1}^{M} \nabla_{\mathbf{w}_N} J_{N,p}(\boldsymbol{\theta}^*, \mathbf{w}_N^*) \end{bmatrix} = \mathbf{0},$$

and also the optimality condition with respect to $\boldsymbol{\theta}$:

$$\mathbf{s}^* := \frac{1}{NM}\sum_{i=1}^{N}\sum_{p=1}^{M} \nabla_{\boldsymbol{\theta}} J_{i,p}(\boldsymbol{\theta}^*, \mathbf{w}_i^*) = 0.$$

For any given matrix Q, recall that $\|Q\|_S$ denotes its spectral norm. Therefore it follows that

$$\begin{aligned}
\|\underline{\mathbf{s}}^{k+1} - \bar{\mathbf{s}}^{k+1}\mathbf{v}^\top\| \leq\ & (\sigma_{\mathbf{C}_1} + \sigma_{\mathbf{C}_2}\rho\gamma\|\mathbf{R}_2\|_S) \max_{(k-M)_+ \leq l \leq k} \|\underline{\mathbf{s}}^l - \bar{\mathbf{s}}^l\mathbf{v}^\top\| \\
& + \sigma_{\mathbf{C}_2}\frac{\rho}{M} \sum_{(k-M)_+ \leq s \leq k} \|\mathbf{R}_1 - \mathbf{I}\|_S \|\underline{\boldsymbol{\theta}}^l - \bar{\boldsymbol{\theta}}^l\mathbf{1}^\top\| \\
& + \sigma_{\mathbf{C}_2}\frac{\bar{A}^2}{M} \sum_{(k-M)_+ \leq l \leq k} \gamma_2 \max_{(l-M)_+ \leq s \leq l} \|\underline{\boldsymbol{\theta}}^s - \boldsymbol{\theta}^*\mathbf{1}^\top\| \\
& + \sigma_{\mathbf{C}_2}\frac{\bar{A}\bar{D}}{M} \sum_{(k-M)_+ \leq l \leq k} \gamma_2 \max_{(l-M)_+ \leq s \leq l} \|\underline{\mathbf{w}}^s - \underline{\mathbf{w}}^*\| \\
& + \sigma_{\mathbf{C}_2}\frac{\rho}{M} \sum_{(k-M)_+ \leq l \leq k} \gamma \|\mathbf{R}_2\mathbf{v}\| \Big(\frac{\rho}{\sqrt{N}} \|\underline{\boldsymbol{\theta}}^s - \bar{\boldsymbol{\theta}}^s\mathbf{1}^\top\| \\
& + \rho \max_{(l-M)_+ \leq s \leq l} \|\bar{\boldsymbol{\theta}}^s - \boldsymbol{\theta}^*\| + \frac{\bar{A}}{N} \max_{(l-M)_+ \leq s \leq l} \sum_{i=1}^{N} \|\mathbf{w}_i^s - \mathbf{w}_i^*\| \Big),
\end{aligned}$$

Further bounding the RHS of the above inequality, we have

$$
\begin{aligned}
\|\underline{\mathbf{s}}^{k+1} - \bar{\mathbf{s}}^{k+1}\mathbf{v}^\top\| \leq\ & (\sigma_{\mathbf{C}_1} + \sigma_{\mathbf{C}_2}\rho\gamma\|\mathbf{R}_2\|_S) \max_{(k-M)_+\leq l\leq k} \|\underline{\mathbf{s}}^l - \bar{\mathbf{s}}^l\mathbf{v}^\top\| \\
& + \big(\sigma_{\mathbf{C}_2}\rho\|\mathbf{R}_1 - \mathbf{I}\|_S + \sigma_{\mathbf{C}_2}\rho^2\gamma\|\mathbf{R}_2\mathbf{v}\|\frac{1}{\sqrt{N}}\big) \\
& \cdot \max_{(k-2M)_+\leq s\leq k} \|\underline{\boldsymbol{\theta}}^s - \bar{\boldsymbol{\theta}}^s\mathbf{1}^\top\| \\
& + \sigma_{\mathbf{C}_2}\bar{A}^2\gamma_2 \max_{(k-2M)_+\leq s\leq k} \|\underline{\boldsymbol{\theta}}^s - \boldsymbol{\theta}^*\mathbf{1}^\top\| \\
& + \sigma_{\mathbf{C}_2}\rho^2\gamma\|\mathbf{R}_2\mathbf{v}\| \max_{(k-2M)_+\leq s\leq k} \|\bar{\boldsymbol{\theta}}^s - \boldsymbol{\theta}^*\| \\
& + \big(\sigma_{\mathbf{C}_2}\bar{A}\bar{D}\gamma_2 + \sigma_{\mathbf{C}_2}\rho\bar{A}\gamma\|\mathbf{R}_2\mathbf{v}\|\frac{1}{\sqrt{N}}\big) \cdot \max_{(k-2M)_+\leq s\leq k} \|\underline{\mathbf{w}}^s - \underline{\mathbf{w}}^*\| \\
& \overset{(c)}{\leq} (\sigma_{\mathbf{C}_1} + \sigma_{\mathbf{C}_2}\rho\gamma\|\mathbf{R}_2\|_S) \max_{(k-M)_+\leq l\leq k} \|\underline{\mathbf{s}}^l - \bar{\mathbf{s}}^l\mathbf{v}^\top\| \\
& + \big(\sigma_{\mathbf{C}_2}\rho\|\mathbf{R}_1 - \mathbf{I}\|_S + \sigma_{\mathbf{C}_2}\rho^2\gamma\|\mathbf{R}_2\mathbf{v}\|\frac{1}{\sqrt{N}} + \sigma_{\mathbf{C}_2}\bar{A}^2\gamma_2\big) \\
& \cdot \max_{(k-2M)_+\leq s\leq k} \|\underline{\boldsymbol{\theta}}^s - \bar{\boldsymbol{\theta}}^s\mathbf{1}^\top\| \\
& + \big(\sigma_{\mathbf{C}_2}\bar{A}^2\gamma_2\sqrt{N} + \sigma_{\mathbf{C}_2}\rho^2\gamma\|\mathbf{R}_2\mathbf{v}\|\big) \cdot \max_{(k-2M)_+\leq s\leq k} \|\bar{\boldsymbol{\theta}}^s - \boldsymbol{\theta}^*\| \\
& + \big(\sigma_{\mathbf{C}_2}\bar{A}\bar{D}\gamma_2 + \sigma_{\mathbf{C}_2}\rho\bar{A}\gamma\|\mathbf{R}_2\mathbf{v}\|\frac{1}{\sqrt{N}}\big) \\
& \cdot \max_{(k-2M)_+\leq s\leq k} \|\underline{\mathbf{w}}^s - \underline{\mathbf{w}}^*\|, \qquad (5.52)
\end{aligned}
$$

where we use the triangle inequality in (c).

5.2.2 Proof of Theorem 3.3.1

For any $\beta > 0$, by the optimality condition, the primal-dual optimal solution to the optimal problem (3.12), $(\boldsymbol{\theta}^*, \{\mathbf{w}_i^*\}_{i=1}^N)$, satisfies

$$\mathbf{G} \begin{bmatrix} \boldsymbol{\theta}^* \\ \frac{r'}{\sqrt{\beta N}}\mathbf{w}_1^* \\ \vdots \\ \frac{r'}{\sqrt{\beta N}}\mathbf{w}_N^* \end{bmatrix} = - \begin{bmatrix} 0 \\ \sqrt{\frac{\beta}{N}}\widehat{\boldsymbol{b}}_1 \\ \vdots \\ \sqrt{\frac{\beta}{N}}\widehat{\boldsymbol{b}}_N \end{bmatrix}, \tag{5.53}$$

where we define the constant $r' := \frac{\mathbf{u}^\top \mathbf{v}}{N}$ and

$$\mathbf{G} = \begin{bmatrix} \rho r' \mathbf{I} & \sqrt{\frac{\beta}{N}}\widehat{\mathbf{A}}^\top & \cdots & \sqrt{\frac{\beta}{N}}\widehat{\mathbf{A}}^\top \\ -\sqrt{\frac{\beta}{N}}\widehat{\mathbf{A}} & \frac{\beta}{r'}\widehat{\mathbf{D}} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ -\sqrt{\frac{\beta}{N}}\widehat{\mathbf{A}} & \mathbf{0} & \cdots & \frac{\beta}{r'}\widehat{\mathbf{D}} \end{bmatrix}. \tag{5.54}$$

For $p \in \{1, \cdots, M\}$, $\mathbf{G}_p$ is defined as

$$\mathbf{G}_p = \begin{bmatrix} \rho r' \mathbf{I} & \sqrt{\frac{\beta}{N}}\mathbf{A}_p^\top & \cdots & \sqrt{\frac{\beta}{N}}\mathbf{A}_p^\top \\ -\sqrt{\frac{\beta}{N}}\mathbf{A}_p & \frac{\beta}{r'}\mathbf{D}_p & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ -\sqrt{\frac{\beta}{N}}\mathbf{A}_p & \mathbf{0} & \cdots & \frac{\beta}{r'}\mathbf{D}_p \end{bmatrix}. \tag{5.55}$$

By definition we know that $\mathbf{G} = \frac{1}{M}\sum_{p=1}^M \mathbf{G}_p$.

Define $\bar{\boldsymbol{\theta}}^k := \frac{1}{N}\mathbf{u}^\top \underline{\boldsymbol{\theta}}^k$ as the weighted average of the parameters at iteration k.

Furthermore, define

$$h_{\boldsymbol{\theta}}(k) := \rho\bar{\boldsymbol{\theta}}^k + \frac{1}{N}\sum_{i=1}^{N}\widehat{\mathbf{A}}^\top \mathbf{w}_i^k \tag{5.56}$$

$$\widehat{g}_{\boldsymbol{\theta}}(k) := \frac{1}{N}\underline{\mathbf{s}}^k\mathbf{u}$$

$$h_{\mathbf{w}_i}(k) := \widehat{\mathbf{A}}\bar{\boldsymbol{\theta}}^k - \widehat{\mathbf{D}}\mathbf{w}_i^k - \widehat{\boldsymbol{b}}_i \tag{5.57}$$

$$g_{\mathbf{w}_i}(k) := \frac{1}{M}\sum_{p=1}^{M}(\mathbf{A}_p\boldsymbol{\theta}_i^{\tau_p^k} - \mathbf{D}_p\mathbf{w}_i^{\tau_p^k} - \boldsymbol{b}_{p,i}),$$

where $h_{\boldsymbol{\theta}}(k)$ and $h_{\mathbf{w}_i}(k)$ represent the batch gradients *w.r.t.* $\boldsymbol{\theta}$ and $\mathbf{w}_i$ at $(\bar{\boldsymbol{\theta}}^k, \mathbf{w}_i^k)$. It can be checked that $\bar{\boldsymbol{\theta}}^{k+1} = \bar{\boldsymbol{\theta}}^k - \gamma_1\widehat{g}_{\boldsymbol{\theta}}(k)$ and $\mathbf{w}_i^{k+1} = \mathbf{w}_i^k - \gamma_2 g_{\mathbf{w}_i}(k)$ for all $k \geq 1$. That is, the primal-dual variables $\bar{\boldsymbol{\theta}}^{k+1}$ and $\mathbf{w}_i^{k+1}$ are updated with $\widehat{g}_{\boldsymbol{\theta}}(k)$ and $g_{\mathbf{w}_i}(k)$. We also define $\underline{\mathbf{h}}(k)$, $\underline{\mathbf{g}}(k)$, and $\underline{\boldsymbol{v}}^k$ by

$$\underline{\mathbf{h}}(k) = \begin{bmatrix} r' h_{\boldsymbol{\theta}}(k) \\ -\sqrt{\frac{\beta}{N}} h_{\mathbf{w}_1}(k) \\ \vdots \\ -\sqrt{\frac{\beta}{N}} h_{\mathbf{w}_N}(k) \end{bmatrix}, \quad \underline{\mathbf{g}}(k) = \begin{bmatrix} \widehat{g}_{\boldsymbol{\theta}}(k) \\ -\sqrt{\frac{\beta}{N}} g_{\mathbf{w}_1}(k) \\ \vdots \\ -\sqrt{\frac{\beta}{N}} g_{\mathbf{w}_N}(k) \end{bmatrix}, \tag{5.58}$$

and

$$\underline{\boldsymbol{v}}^k = \begin{bmatrix} \bar{\boldsymbol{\theta}}^s - \boldsymbol{\theta}^* \\ \frac{r'}{\sqrt{\beta N}}(\mathbf{w}_1^k - \mathbf{w}_1^*) \\ \vdots \\ \frac{r'}{\sqrt{\beta N}}(\mathbf{w}_N^k - \mathbf{w}_N^*) \end{bmatrix}. \tag{5.59}$$

Note (5.56) and (5.57) can be written as

$$\mathbf{G}\begin{bmatrix} \bar{\boldsymbol{\theta}}(k) \\ \frac{r'}{\sqrt{\beta N}}\mathbf{w}_1^k \\ \vdots \\ \frac{r'}{\sqrt{\beta N}}\mathbf{w}_N^k \end{bmatrix} = \begin{bmatrix} r' h_{\boldsymbol{\theta}}(k) \\ -\sqrt{\frac{\beta}{N}} h_{\mathbf{w}_1}(k) - \sqrt{\frac{\beta}{N}}\widehat{\boldsymbol{b}}_1 \\ \vdots \\ -\sqrt{\frac{\beta}{N}} h_{\mathbf{w}_N}(k) - \sqrt{\frac{\beta}{N}}\widehat{\boldsymbol{b}}_N \end{bmatrix}. \tag{5.60}$$

Combining (5.60) and (5.53) yields

$$\underline{\mathbf{h}}(k) = \mathbf{G}\underline{\boldsymbol{v}}^k. \tag{5.61}$$

By the analysis similar to [97], it can be shown that under Assumption 3.3.2 and with $\beta := \frac{8r'(\rho+\lambda_{\max}(\widehat{\mathbf{A}}^\top\widehat{\mathbf{D}}^{-1}\widehat{\mathbf{A}}))}{\lambda_{\min}(\widehat{\mathbf{D}})}$, $\mathbf{G}$ is full rank and has eigenvalues satisfying

$$\begin{aligned}\lambda_{\max}(\mathbf{G}) &\leq \Big|\frac{\lambda_{\max}(\widehat{\mathbf{D}})}{\lambda_{\min}(\widehat{\mathbf{D}})}\Big|\lambda_{\max}(\rho r'\mathbf{I} + r'\widehat{\mathbf{A}}^\top\widehat{\mathbf{D}}^{-1}\widehat{\mathbf{A}}) \\ \lambda_{\min}(\mathbf{G}) &\geq \frac{8r'}{9}\lambda_{\min}(\widehat{\mathbf{A}}^\top\widehat{\mathbf{D}}^{-1}\widehat{\mathbf{A}}).\end{aligned} \tag{5.62}$$

By the definition of r' and Lemma 3.3.2, we know that $r' \neq 0$. Furthermore, assume $\mathbf{G} := \mathbf{U}\Lambda\mathbf{U}^{-1}$ is the eigen-decomposition of $\mathbf{G}$, where Λ is the diagonal matrix of eigenvalues of $\mathbf{G}$; the columns of $\mathbf{U}$ are the eigenvectors. Then one can let $\mathbf{U}$ satisfy

$$\begin{aligned}\|\mathbf{U}\| &\leq 8r'\Big|\frac{\lambda_{\max}(\widehat{\mathbf{D}})}{\lambda_{\min}(\widehat{\mathbf{D}})}\Big|(\rho + \lambda_{\max}(\widehat{\mathbf{A}}^\top\widehat{\mathbf{D}}^{-1}\widehat{\mathbf{A}})) \\ \|\mathbf{U}^{-1}\| &\leq \frac{1}{\rho r' + r'\lambda_{\max}(\widehat{\mathbf{A}}^\top\widehat{\mathbf{D}}^{-1}\widehat{\mathbf{A}})}.\end{aligned} \tag{5.63}$$

Furthermore, we define the upper bounds of the spectral norms by

$$G := \|\mathbf{G}\|_S, \qquad \bar{G} = \max_{p=1,\cdots,M}\|\mathbf{G}_p\|_S, \tag{5.64}$$

$$\bar{A} = \max_{p=1,\cdots,M}\|\mathbf{A}_p\|_S, \qquad \bar{D} = \max_{p=1,\cdots,M}\|\mathbf{D}_p\|_S. \tag{5.65}$$

We also define the Lyapunov function as

$$\boldsymbol{\varepsilon}_c(k) := \frac{1}{N}\sum_{i=1}^{N}\|\boldsymbol{\theta}_i^k - \bar{\boldsymbol{\theta}}^k\|.$$

Next, recall that $\gamma_1 := \gamma$ and $\gamma_2 := \beta\gamma$. In the following we establish a bound on the optimality gap of the primal-dual variables, $\underline{\boldsymbol{v}}^k$. Note that $\underline{\boldsymbol{v}}^{k+1} = \underline{\boldsymbol{v}}^k - \gamma\underline{\mathbf{g}}(k)$. Thus we

have

$$\underline{\boldsymbol{v}}^{k+1} = (\mathbf{I} - \gamma \mathbf{G})\underline{\boldsymbol{v}}^{k} + \gamma(\underline{\mathbf{h}}(k) - \underline{\mathbf{g}}(k)). \tag{5.66}$$

Now we consider the difference $\underline{\mathbf{h}}(k) - \underline{\mathbf{g}}(k)$. Its first block can be written as

$$\begin{aligned}
[\underline{\mathbf{h}}(k) - \underline{\mathbf{g}}(k)]_1 &= r' h_{\boldsymbol{\theta}}(k) - \widehat{g}_{\boldsymbol{\theta}}(k) \\
&= -\frac{1}{N}(\underline{\mathbf{s}}^k - \bar{\mathbf{s}}^k \mathbf{v}^\top)\mathbf{u} - \frac{1}{N}\bar{\mathbf{s}}^k \mathbf{v}^\top \mathbf{u} + r' h_{\boldsymbol{\theta}}(k) \\
&= -\frac{1}{N}(\underline{\mathbf{s}}^k - \bar{\mathbf{s}}^k \mathbf{v}^\top)(\mathbf{u} - \mathbf{1}) + r'(h_{\boldsymbol{\theta}}(k) - \bar{\mathbf{s}}^k).
\end{aligned} \tag{5.67}$$

For any $i \in \{1, \cdots, N\}$, the $(i+1)$-th block is

$$\begin{aligned}
[\underline{\mathbf{h}}(k) - \underline{\mathbf{g}}(k)]_{i+1} &= \sqrt{\frac{\beta}{N}} \frac{1}{M} \sum_{p=1}^{M} \mathbf{A}_p \Big(\bar{\boldsymbol{\theta}}^k - \boldsymbol{\theta}_i^{\tau_p^k} \Big) + \mathbf{D}_p (\mathbf{w}_i^k - \mathbf{w}_i^{\tau_p^k}) \\
&= \sqrt{\frac{\beta}{N}} \frac{1}{M} \sum_{p=1}^{M} \mathbf{A}_p \Big(\bar{\boldsymbol{\theta}}^k - \bar{\boldsymbol{\theta}}^{\tau_p^k} \Big) + \mathbf{D}_p (\mathbf{w}_i^k - \mathbf{w}_i^{\tau_p^k}) \\
&\quad + \sqrt{\frac{\beta}{N}} \frac{1}{M} \sum_{p=1}^{M} \mathbf{A}_p \Big(\bar{\boldsymbol{\theta}}^{\tau_p^k} - \boldsymbol{\theta}_i^{\tau_p^k} \Big).
\end{aligned} \tag{5.68}$$

We construct the residual vector $\underline{\boldsymbol{\varepsilon}}_c(k)$ as the following: the first block of $\underline{\boldsymbol{\varepsilon}}_c(k)$ is $-\frac{1}{N}(\underline{\mathbf{s}}^k - \bar{\mathbf{s}}^k \mathbf{v}^\top)(\mathbf{u} - \mathbf{1}) + \frac{\rho r'}{NM} \sum_{i=1}^{N} \sum_{p=1}^{M} \big(\boldsymbol{\theta}_i^{\tau_p^k} - \bar{\boldsymbol{\theta}}^{\tau_p^k} \big)$ and the remaining blocks are given by $\sqrt{\frac{\beta}{N}} \frac{1}{M} \sum_{p=1}^{M} \mathbf{A}_p \Big(\boldsymbol{\theta}_i^{\tau_p^k} - \bar{\boldsymbol{\theta}}^{\tau_p^k} \Big)$, $i \in \{1, \cdots, N\}$. By (5.67), (5.68), and the definition of $\mathbf{G}_p$ in (5.55), we have the following simple form of $\underline{\mathbf{h}}(k) - \underline{\mathbf{g}}(k)$:

$$\underline{\mathbf{h}}(k) - \underline{\mathbf{g}}(k) - \underline{\boldsymbol{\varepsilon}}_c(k) = \frac{1}{M} \sum_{p=1}^{M} \mathbf{G}_p \big(\sum_{j=\tau_p^k}^{k-1} \Delta \mathbf{v}(j) \big), \tag{5.69}$$

where we define

$$\Delta \underline{\boldsymbol{v}}^j = \begin{bmatrix} \bar{\boldsymbol{\theta}}^{j+1} - \bar{\boldsymbol{\theta}}^j \\ \frac{r'}{\sqrt{\beta N}} (\mathbf{w}_1^{j+1} - \mathbf{w}_1^j) \\ \vdots \\ \frac{r'}{\sqrt{\beta N}} (\mathbf{w}_N^{j+1} - \mathbf{w}_N^j) \end{bmatrix}. \tag{5.70}$$

Note it holds that $\Delta\underline{\boldsymbol{v}}^j = \underline{\boldsymbol{v}}^{j+1} - \underline{\boldsymbol{v}}^j$. From (5.66), $\Delta\underline{\boldsymbol{v}}^j$ in (5.70) can also be written as

$$\Delta\underline{\boldsymbol{v}}^j = \gamma[\underline{\mathbf{h}}(j) - \underline{\mathbf{g}}(j)] - \gamma\underline{\mathbf{h}}(j). \tag{5.71}$$

Multiplying $\mathbf{U}^{-1}$ on both sides of (5.66) results in

$$\widehat{\underline{\boldsymbol{v}}}^{k+1} = (\mathbf{I} - \gamma\mathbf{G})\widehat{\underline{\boldsymbol{v}}}^k + \gamma\mathbf{U}^{-1}(\underline{\mathbf{h}}(k) - \underline{\mathbf{g}}(k)), \tag{5.72}$$

where $\widehat{\underline{\boldsymbol{v}}}^k := \mathbf{U}^{-1}\underline{\boldsymbol{v}}^k$. By combining (5.69), (5.71), and (5.72), we have

$$\begin{aligned}
\|\widehat{\underline{\boldsymbol{v}}}^{k+1}\| &\le \|\mathbf{I} - \gamma\Lambda\|\|\widehat{\underline{\boldsymbol{v}}}^k\| + \gamma\|\mathbf{U}^{-1}\|\|\underline{\mathbf{h}}(k) - \underline{\mathbf{g}}(k)\| \\
&\le \|\mathbf{I} - \gamma\Lambda\|\|\widehat{\underline{\boldsymbol{v}}}^k\| + \gamma\|\mathbf{U}^{-1}\|\Big[\|\underline{\boldsymbol{\varepsilon}}_c(k)\| + \frac{1}{M}\sum_{p=1}^{M}\|\mathbf{G}_p\sum_{j=\tau_p^k}^{k-1}\Delta\underline{\boldsymbol{v}}^j\|\Big] \\
&\le \|\mathbf{I} - \gamma\Lambda\|\|\widehat{\underline{\boldsymbol{v}}}^k\| + \gamma\|\mathbf{U}^{-1}\|\Big\{\|\underline{\boldsymbol{\varepsilon}}_c(k)\| \\
&+ \frac{\gamma\bar{G}}{M}\sum_{p=1}^{M}\sum_{j=\tau_p^k}^{k-1}\big[\|\underline{\mathbf{h}}(j)\| + \|\underline{\mathbf{h}}(j) - \underline{\mathbf{g}}(j)\|\big]\Big\},
\end{aligned} \tag{5.73}$$

where $\bar{G}$ is defined in (5.64). By further bounding the right-hand side of (5.73) we have

$$\begin{aligned}
\|\widehat{\underline{\boldsymbol{v}}}^{k+1}\| &\le \|\mathbf{I} - \gamma\Lambda\|\|\widehat{\underline{\boldsymbol{v}}}^k\| + \gamma\|\mathbf{U}^{-1}\|\Big\{\|\underline{\boldsymbol{\varepsilon}}_c(k)\| \\
&\quad + \gamma\bar{G}\sum_{j=(k-M)_+}^{k-1}\big[\|\underline{\mathbf{h}}(j)\| + \|\underline{\mathbf{h}}(j) - \underline{\mathbf{g}}(j)\|\big]\Big\} \\
&\le \|\mathbf{I} - \gamma\Lambda\|\|\widehat{\underline{\boldsymbol{v}}}^k\| + \gamma\|\mathbf{U}^{-1}\|\Big\{\|\underline{\boldsymbol{\varepsilon}}_c(k)\| \\
&\quad + \gamma\bar{G}\sum_{j=(k-M)_+}^{k-1}\Big[\|\underline{\boldsymbol{\varepsilon}}_c(j)\| + G\|\mathbf{U}\|\|\widehat{\underline{\boldsymbol{v}}}^j\| \\
&\quad + \bar{G}\|\mathbf{U}\|\cdot\sum_{j=(k-M)_+}^{k-1}(\|\widehat{\underline{\boldsymbol{v}}}^{l+1}\| + \|\widehat{\underline{\boldsymbol{v}}}^l\|)\Big]\Big\}.
\end{aligned} \tag{5.74}$$

Moreover, $\|\underline{\boldsymbol{\varepsilon}}_c(k)\|$ can be upper bounded by

$$\begin{aligned}\|\underline{\boldsymbol{\varepsilon}}_c(k)\| &\leq \frac{1}{M}\sum_{p=1}^{M}\Big[(\rho r' + \bar{A}\sqrt{\beta N})\big(\frac{1}{N}\sum_{i=1}^{N}\|\boldsymbol{\theta}_i^{\tau_p^k} - \bar{\boldsymbol{\theta}}^{\tau_p^k}\|\big)\Big] \\ &\quad + \frac{1}{N}\|\mathbf{u}-\mathbf{1}\|\|\underline{\mathbf{s}}^k - \bar{\mathbf{s}}^k\mathbf{v}^\top\| \\ &\leq (\rho r' + \bar{A}\sqrt{\beta N}) \max_{(k-M)_+\leq q\leq k}\boldsymbol{\varepsilon}_c(q) + \frac{1}{N}\|\mathbf{u}-\mathbf{1}\|\|\underline{\mathbf{s}}^k - \bar{\mathbf{s}}^k\mathbf{v}^\top\|.\end{aligned}$$

Thus, we can bound $\|\widehat{\underline{\boldsymbol{v}}}^{k+1}\|$ by

$$\begin{aligned}\|\widehat{\underline{\boldsymbol{v}}}^{k+1}\| &\leq \|\mathbf{I}-\gamma\Lambda\|\|\widehat{\underline{\boldsymbol{v}}}^{k}\| + C_1(\gamma)\max_{(k-2M)_+\leq q\leq k-1}\|\widehat{\underline{\boldsymbol{v}}}^{q}\| \\ &\quad + C_2(\gamma)\max_{(k-2M)_+\leq q\leq k}\boldsymbol{\varepsilon}_c(q) + C_3(\gamma)\|\underline{\mathbf{s}}^k - \bar{\mathbf{s}}^k\mathbf{v}^\top\|, \qquad (5.75)\end{aligned}$$

where constants $C_1(\gamma)$, $C_2(\gamma)$, and $C_3(\gamma)$ are defined as

$$\begin{aligned}C_1(\gamma) &= \gamma^2\|\mathbf{U}\|\|\mathbf{U}^{-1}\|\bar{G}M(G+2\bar{G}M), \\ C_2(\gamma) &= \gamma\|\mathbf{U}^{-1}\|(1+\gamma\bar{G}M)(\rho r' + \bar{A}\sqrt{\beta N}), \\ C_3(\gamma) &= \gamma\|\mathbf{U}^{-1}\|\frac{\|\mathbf{u}-\mathbf{1}\|}{N}(1+\gamma\bar{G}M). \qquad (5.76)\end{aligned}$$

Now combining (5.51), (5.52), and (5.75) yields

$$\begin{bmatrix}\|\underline{\boldsymbol{\theta}}^{k+1} - \bar{\boldsymbol{\theta}}^{k+1}\mathbf{1}^\top\| \\ \|\underline{\mathbf{s}}^{k+1} - \bar{\mathbf{s}}^{k+1}\mathbf{v}^\top\| \\ \|\widehat{\underline{\boldsymbol{v}}}^{k+1}\|\end{bmatrix} \leq Q(\gamma)\begin{bmatrix}\max_{(k-2M)_+\leq s\leq k}\|\underline{\boldsymbol{\theta}}^{s} - \bar{\boldsymbol{\theta}}^{s}\mathbf{1}^\top\| \\ \max_{(k-M)_+\leq s\leq k}\|\underline{\mathbf{s}}^{s} - \bar{\mathbf{s}}^{s}\mathbf{v}^\top\| \\ \max_{(k-2M)_+\leq s\leq k}\|\widehat{\underline{\boldsymbol{v}}}^{s}\|\end{bmatrix},$$

where matrix $Q(\gamma)$ is defined by

$$Q(\gamma) := \begin{bmatrix}\sigma_{\mathbf{R}_1} + \gamma\sigma_{\mathbf{R}_2}\|\mathbf{v}-\mathbf{1}\|\frac{\rho}{\sqrt{N}} & \gamma\sigma_{\mathbf{R}_2} & C_6(\gamma) \\ C_7(\gamma) & C_4(\gamma) & C_5(\gamma) \\ \frac{1}{\sqrt{N}}C_2(\gamma) & C_3(\gamma) & C_8(\gamma)\end{bmatrix},$$

and

$$C_4(\gamma) := \sigma_{\mathbf{C}_1} + \sigma_{\mathbf{C}_2}\rho\gamma\|\mathbf{R}_2\|_S$$

$$
\begin{aligned}
C_5(\gamma) &:= \sqrt{2}\big(\sigma_{\mathbf{C}_2} L^2 \gamma_2 \sqrt{N} + \sigma_{\mathbf{C}_2} L^2 \gamma \|\mathbf{R}_2 \mathbf{v}\|\big) \max\{1, \frac{\sqrt{\beta}}{r'}\} \\
C_6(\gamma) &:= \sqrt{2}\gamma \sigma_{\mathbf{R}_2} \| \mathbf{v} - \mathbf{1}\| L \max(1, \frac{\sqrt{\beta}}{r'}) \\
C_7(\gamma) &:= \sigma_{\mathbf{C}_2} \rho \|\mathbf{R}_1 - \mathbf{I}\|_S + \gamma(\sigma_{\mathbf{C}_2} \rho^2 \|\mathbf{R}_2 \mathbf{v}\| \frac{1}{\sqrt{N}} + \sigma_{\mathbf{C}_2} \bar{A}^2 \beta) \\
C_8(\gamma) &:= \|I - \gamma \Lambda\| + C_1(\gamma),
\end{aligned} \tag{5.77}
$$

with $L := \max\{\rho, \bar{A}, \bar{D}\}$. Note that the eigenvalues of $\mathbf{G}$ are bounded in (5.62). Thus by setting the step size γ to be small enough we can ensure $\|\mathbf{1} - \gamma\Lambda\| < 1$. Hence there exists some $\alpha > 0$ such that $\|\mathbf{1} - \gamma\Lambda\| = 1 - \gamma\alpha$. Moreover, the upper bound of $\|\mathbf{U}\|$ and $\|\mathbf{U}^{-1}\|$ are given in (5.63). Now define

$$
\begin{aligned}
a_1 &:= \sigma_{\mathbf{R}_2} \| \mathbf{v} - \mathbf{1}\| \frac{\rho}{\sqrt{N}} \\
a_2 &:= \sqrt{2}\sigma_{\mathbf{R}_2} \| \mathbf{v} - \mathbf{1}\| L \max(1, \sqrt{\beta}/r') \\
a_3 &:= \sigma_{\mathbf{C}_2} \rho \|\mathbf{R}_1 - \mathbf{I}\|_S \\
a_4 &:= \sigma_{\mathbf{C}_2} \rho^2 \|\mathbf{R}_2 \mathbf{v}\| \frac{1}{\sqrt{N}} + \sigma_{\mathbf{C}_2} \bar{A}^2 \beta \\
a_5 &:= \sigma_{\mathbf{C}_2} \rho \|\mathbf{R}_2\|_S \\
a_6 &:= \sqrt{2}\big(\sigma_{\mathbf{C}_2} L^2 \beta \sqrt{N} + \sigma_{\mathbf{C}_2} L^2 \|\mathbf{R}_2 \mathbf{v}\|\big) \max\{1, \sqrt{\beta}/r'\} \\
a_7 &:= \frac{1}{\sqrt{N}} \|\mathbf{U}^{-1}\| \big(\rho r' + \bar{A}\sqrt{\beta N}\big) \\
a_8 &:= \frac{1}{\sqrt{N}} \|\mathbf{U}^{-1}\| 2\bar{G}M \big(\rho r' + \bar{A}\sqrt{\beta N}\big) \\
a_9 &:= \|\mathbf{U}\| \|\mathbf{U}^{-1}\| \bar{G}M(G + 2\bar{G}M) \\
a_{10} &:= \frac{\|\mathbf{u} - \mathbf{1}\|}{N} \|\mathbf{U}^{-1}\| (1 + \bar{G}).
\end{aligned}
$$

Observe that when $\gamma < \frac{1}{M}$ we have

$$
\mathbf{Q} \leq \mathbf{Q}_1 := \begin{bmatrix}
\sigma_{\mathbf{R}_1} + a_1\gamma & \sigma_{\mathbf{R}_2}\gamma & a_2\gamma \\
a_3 + a_4\gamma & \sigma_{\mathbf{C}_1} + a_5\gamma & a_6\gamma \\
a_7\gamma + a_8\gamma^2 & a_{10}\gamma & (1 - \gamma\alpha + a_9\gamma^2)
\end{bmatrix},
$$

where the inequality means $\mathbf{Q}$ is entrywise less than $\mathbf{Q}_1$.

To this end, we consider

$$\mathbf{g}(\sigma) := |\sigma\mathbf{I} - \mathbf{Q}_1| = \begin{vmatrix} \sigma - \sigma_{\mathbf{R}_1} - a_1\gamma & -\sigma_{\mathbf{R}_2}\gamma & -a_2\gamma \\ -a_3 - a_4\gamma & \sigma - \sigma_{\mathbf{C}_1} - a_5\gamma & -a_6\gamma \\ -a_7\gamma - a_8\gamma^2 & -a_{10}\gamma & \sigma - (1 - \gamma\alpha + a_9\gamma^2) \end{vmatrix}.$$

Therefore

$$\begin{aligned} \mathbf{g}(\sigma) = {} & \big(\sigma - (1 - \gamma\alpha + \gamma^2 a_9)\big)\mathbf{g}_0(\sigma) \\ & - (a_7\gamma + a_8\gamma^2)\big(\sigma_{\mathbf{R}_2} a_6\gamma^2 + a_2\gamma(\sigma - \sigma_{\mathbf{C}_1} - a_5\gamma)\big) \\ & - a_{10}\gamma\big(a_2\gamma(a_3 + a_4\gamma) + a_6\gamma(\sigma - \sigma_{\mathbf{R}_1} - a_1\gamma)\big), \end{aligned}$$

where

$$\mathbf{g}_0(\sigma) := (\sigma - \sigma_{\mathbf{R}_1} - a_1\gamma)(\sigma - \sigma_{\mathbf{C}_1} - a_5\gamma) - \sigma_{\mathbf{R}_2}\gamma(a_3 + a_4\gamma).$$

Without loss of generality, assume $\sigma_{\mathbf{R}_1} \le \sigma_{\mathbf{C}_1}$. We define $\bar{\sigma} := \sigma_{\mathbf{C}_1} + (a_1 + a_5)\gamma + \sqrt{\sigma_{\mathbf{R}_2}\gamma(a_3 + a_4\gamma)}$. It can be checked that for all $\sigma \ge \bar{\sigma}$, we have

$$\mathbf{g}_0(\sigma) \ge (\sigma - \bar{\sigma})^2.$$

Now we define

$$\begin{aligned} \sigma^* := \max\Bigg\{ & \frac{\gamma\alpha}{4} + (1 - \gamma\alpha + \gamma^2 a_9), \bar{\sigma} + \frac{2a_2\gamma(a_7 + a_8)}{\alpha} + \frac{2a_6 a_{10}\gamma}{\alpha} + \\ & \sqrt{\gamma}\Bigg\{ \frac{\frac{a_2^2\gamma^2(a_7 + a_8\gamma)^2}{\alpha} + (a_7\gamma + a_8\gamma^2)\sigma_{\mathbf{R}_2} a_6\gamma}{\frac{\gamma\alpha}{4}} + \frac{a_2 a_{10}\gamma(a_3 + a_4\gamma)}{\frac{\gamma\alpha}{4}} \\ & + \frac{a_2\gamma(a_7 + a_8\gamma)\big(a_1\gamma + \sqrt{\sigma_{\mathbf{R}_2}\gamma(a_3 + a_4\gamma)}\big)}{\frac{\gamma\alpha}{4}} \\ & + \frac{a_6 a_{10}\gamma\big(\sigma_{\mathbf{C}_1} - \sigma_{\mathbf{R}_1} + a_5\gamma + \sqrt{\sigma_{\mathbf{R}_2}\gamma(a_3 + a_4\gamma)}\big)}{\frac{\gamma\alpha}{4}} \Bigg\}^{\frac{1}{2}} \Bigg\}. \end{aligned} \tag{5.78}$$

For all $\sigma \geq \sigma^*$, it holds that

$$\begin{aligned}
\mathbf{g}(\sigma) \geq & \big(\sigma - (1 - \gamma\alpha + \gamma^2 a_9)\big)(\sigma - \bar{\sigma})^2 \\
& - (a_7\gamma + a_8\gamma^2)\big(\sigma_{\mathbf{R}_2} a_6\gamma^2 + a_2\gamma(\sigma - \sigma_{\mathbf{C}_1} - a_5\gamma)\big) \\
& - a_{10}\gamma\big(a_2\gamma(a_3 + a_4\gamma) + a_6\gamma(\sigma - \sigma_{\mathbf{R}_1} - a_1\gamma)\big) \\
& \geq \frac{\gamma\alpha}{4}\big(\sigma - \bar{\sigma} - \frac{2a_2\gamma(a_7 + a_8\gamma)}{\alpha} - \frac{2a_6 a_{10}\gamma}{\alpha}\big)^2 \\
& - \frac{a_2^2\gamma^3(a_7 + a_8\gamma)^2}{\alpha} - a_2\gamma^2(a_7 + a_8\gamma)(\bar{\sigma} - \sigma_C - a_5\gamma) \\
& - (a_7\gamma + a_8\gamma^2)\sigma_{\mathbf{R}_2} a_6\gamma^2 - a_2 a_{10}\gamma^2(a_3 + a_4\gamma) \\
& - a_6 a_{10}\gamma^2\big(\bar{\sigma} - \sigma_{\mathbf{R}_1} - a_1\gamma\big) \\
& \geq 0.
\end{aligned}$$

Now from the Perron-Frobenius Theorem [152] one can conclude that $\rho(\mathbf{Q}) < \sigma^*$. Moreover, by Assumptions 3.3.3-3.3.4, one has $\max\{\sigma_{\mathbf{R}_1}, \sigma_{\mathbf{C}_1}\} < 1$ ([46]). As $\alpha > 0$, there exists a sufficiently small γ such that $\sigma^* < 1$, which implies $\rho(\mathbf{Q}) < 1$.

Next, let us consider the asymptotic rate when $M, N \gg 1$. Note that the proposed algorithm converges if $\sigma^* < 1$. Let us consider the first operand in the $\max\{\cdot\}$ of (5.78). The first operand will be less than 1 if $0 < \gamma < \frac{\alpha}{2a_9}$ since

$$\frac{\gamma\alpha}{4} + 1 - \gamma\alpha + \gamma^2 a_9 \leq 1 - \frac{\gamma\alpha}{4} < 1. \tag{5.79}$$

By the definition of a_9, this requires $\gamma = \mathcal{O}(1/M^2)$ if $M \gg 1$.

Note that we have $\sigma_{\mathbf{C}_1} = 1 - \frac{\kappa}{N}$ for some positive κ. Using this in the second operand in (5.78) yields

$$\begin{aligned}
& 1 - \frac{\kappa}{N} + (2a_1 + a_5)\gamma + \sqrt{\sigma_{\mathbf{R}_2}\gamma(a_3 + a_4\gamma)} \\
& + \frac{2a_2\gamma(a_7 + a_8\gamma)}{\alpha} + \frac{2a_6 a_{10}\gamma}{\alpha} \\
& + \sqrt{\gamma}\Big\{\frac{4}{\alpha}\Big[a_2^2\gamma(a_2 + a_8\gamma)^2 + (a_7\gamma + a_8\gamma^2)\sigma_{\mathbf{R}_2} a_6 \\
& + a_2(a_7 + a_8\gamma)\big(a_1\gamma + \sqrt{\sigma_{\mathbf{R}_2}\gamma(a_3 + a_4\gamma)}\big) \\
& + a_2 a_{10}(a_3 + a_4\gamma) + a_6 a_{10}\big(\bar{\sigma} - \sigma_{\mathbf{R}_1} - a_1\gamma\big)\Big]\Big\}^{\frac{1}{2}}.
\end{aligned} \tag{5.80}$$

Therefore, to guarantee $\sigma^* < 1$, it's sufficient to let γ satisfy (5.79) and that (5.80) < 1.

Now let us consider the asymptotic rate when $N, M \gg 1$. Observe that $a_1 = \Theta(1)$, $a_2 = \Theta(\sqrt{N})$, $a_3 = \mathcal{O}(1)$, $a_4 = \Theta(1)$, $a_5 = \Theta(1)$, $a_6 = \Theta(\sqrt{N})$, $a_7 = \Theta(1)$, $a_8 = \Theta(M)$, and $a_{10} = o(1)$. Therefore Eq. (5.80) can be approximated by

$$\begin{aligned} &1 - \frac{e}{N} + \Theta(\gamma) + \Theta(\sqrt{\gamma}) + \gamma\Theta(\sqrt{N}) + \Theta(\gamma\frac{1}{\sqrt{N}}) \\ &+ \sqrt{\gamma}\Theta\big(\{N^2\gamma + \sqrt{N\gamma}\}^{\frac{1}{2}}\big). \end{aligned} \tag{5.81}$$

To ensure (5.81) is less than $1 - \frac{e}{2N}$, it requires that $\gamma = \mathcal{O}\big(\frac{1}{N^2}\big)$. In summary, by setting $\gamma = \mathcal{O}\big(1/\max\{N^2, M^2\}\big)$ we have $\sigma^* \leq \max\{1 - \gamma\frac{\alpha}{4}, 1 - e/(2N)\} = 1 - \mathcal{O}(1/\max\{N^2, M^2\})$.