ACCURACY-AWARE PRIVACY MECHANISMS FOR DISTRIBUTED COMPUTATION

BY

SHRIPAD GADE

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2020

Urbana, Illinois

Doctoral Committee:

Assistant Professor Subhonmesh Bose, Chair and Co-Director of Research
Adjunct Professor Nitin Vaidya, Co-Director of Research
Professor Rayadurgam Srikant
Professor Sayan Mitra
Assistant Professor Ji Liu, Stony Brook University

ABSTRACT

Distributed computing systems involve a network of devices or agents that use locally stored private information to solve a common problem. Distributed algorithms fundamentally require communication between devices leaving the system vulnerable to "privacy attacks" perpetrated by adversarial agents. In this dissertation, we focus on designing privacy-preserving distributed algorithms for – (a) solving distributed optimization problems, (b) computing equilibrium of network aggregate games, and (c) solving a distributed system of linear equations. Specifically, we propose a privacy definition for distributed computation – "non-identifiability", that allow us to simultaneously guarantee privacy and the accuracy of the computed solution. This definition involves showing that information observed by the adversary is compatible with several distributed computing problems and the associated ambiguity provides privacy.

- **Distributed Optimization:** We propose the Function Sharing strategy that involves using correlated random functions to obfuscate private objective functions followed by using a standard distributed optimization algorithm. We characterize a tight graph connectivity condition for proving privacy via non-identifiability of local objective functions. We also prove correctness of our algorithm and show that we can achieve privacy and accuracy simultaneously.
- **Network Aggregate Games:** We design a distributed Nash equilibrium computation algorithm for network aggregate games. Our algorithm uses locally balanced correlated random perturbations to hide information shared with neighbors for aggregate estimation. This step is followed by descent along the negative gradient of the local cost function. We show that if the graph of non-adversarial agents is connected and non-bipartite, then our algorithm keeps private local cost information non-identifiable while asymptotically converging to the accurate Nash equilibrium.
- **Average Consensus and System of Linear Equations:** Finally, we design a finite-time algorithm for solving the average consensus problem over directed graphs with information-theoretic privacy. We use this algorithm to solve a distributed system of linear equations in finite-time while protecting the privacy of local equations. We characterize computation, communication, memory and iteration cost of our algorithm and characterize graph conditions for guaranteeing information-theoretic privacy of local data.

To my family, for their unconditional love and support.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor Prof. Nitin Vaidya. He has been a source of constant guidance and support and this work would have been impossible without him. As my academic mentor, he has taught me how to do research, how to write papers and many more things than I credit him for. I am also very thankful for his time, energy and enthusiasm. He was always open to discuss ideas and explore them. I owe him my deepest gratitude and consider myself extremely lucky to be advised by him.

I am also very grateful to Prof. Subhonmesh Bose for being my co-advisor and collaborator. He has been very generous with his time and attention, eager to help and frank in his feedback. I have greatly benefited from his technical astuteness and guidance. I am also very grateful to Prof. Ji Liu for being an amazing collaborator and mentor. His in-depth understanding of consensus based methods was critical to some of the work in this dissertation. I would also like to thank Prof. Sayan Mitra, Prof. R. Srikant and Prof. Ji Liu for serving on my committee and providing feedback.

I found unbelievably talented colleagues at CSL. I am grateful all the technical discussions, coffee chats and their camaraderie. I would like to specifically thank Philip Pare, Thinh Doan, Amir Taghvaei, Arun Raman, Jin Kim, Amber Srivastava, Harsh Gupta, Siddhartha Satpathi, and Surya Sankagiri for their friendship. My colleagues in the DISC group, Lili Su, Daniel Xiang and Samir Khan, have been incredible groupmates and I am grateful for our many technical discussions. I am also thankful to Prof. Bose's research group, Avinash Madavan, Mariola Ndrio, Anna Winnicki and Boya Hou for making me feel one of them. I am also grateful to my UIUC friends Abhilash Harpale, Saptarshi Bandyopadhyay, Yashwanth Nakka, Ameya Patil, Ritwika Ghosh, Mansi Kumar, Amish Goel, Mohit Goyal, and Anurendra Kumar; and my IITB friends Raunak Borker, Anand Pratap Singh, and Prasad Bandarkar for being my support system. The time spent with you helped me get through the various difficulties of graduate student life.

I am also grateful to the various UIUC administrators who have helped me during the last few years. They made my life at UIUC so much easier. Special thanks to Jameson Laura Smith and Laurie Fischer (ECE), Carol Wisniewski and Angie Ellis (CSL), and Staci McDannel (AE) who have always been eager to help. I thank Jan Progen for carefully editing this dissertation.

More importantly, I would like to thank my parents and my sister, Gayatri. Their unconditional love and encouragement has been crucial as I navigated graduate school. They give me courage to persevere and strive for the best. This dissertation would have been impossible without them.

Finally, I am grateful to National Science Foundation, Aerospace Engineering and Electrical and Computer Engineering departments at UIUC for financially supporting me during graduate school.

TABLE OF CONTENTS

LIST OF SYMBOLS viii

CHAPTER 1 INTRODUCTION 1
1.1 Prior Work on Privacy in Distributed Computation 3
1.2 Summary of Contributions 4

CHAPTER 2 PRIVATE OPTIMIZATION OVER NETWORKS 6
2.1 Introduction 6
2.2 Problem Setup: Assumptions, Adversary Model and Privacy Definition 11
2.3 Function Sharing Algorithm for Directed Networks 13
2.4 Function Sharing Algorithm for Undirected Graphs 31
2.5 Conclusions 48

CHAPTER 3 PRIVATE OPTIMIZATION FOR FEDERATED LEARNING 49
3.1 Introduction 49
3.2 Problem Formulation 50
3.3 POLAR-SGD Algorithm 52
3.4 Main Results and Discussion 57
3.5 Analysis and Discussion 58
3.6 Experimental Validation 73
3.7 Conclusion 74

CHAPTER 4 PRIVATE EQUILIBRIUM COMPUTATION ALGORITHM FOR NETWORK AGGREGATE GAMES 75
4.1 Introduction 75
4.2 Equilibrium Computation in Aggregate Games and the Lack of Privacy 76
4.3 Our Algorithm and Its Properties 81
4.4 A Numerical Experiment 83
4.5 Privacy Analysis and Proof of Theorem 7 85
4.6 Convergence Analysis and Correctness 93
4.7 Conclusions 97
4.8 Proof of Lemma 14 97

CHAPTER 5 PRIVATE AND FINITE-TIME ALGORITHM FOR SOLVING A DISTRIBUTED SYSTEM OF LINEAR EQUATIONS 101
5.1 Introduction 101
5.2 Problem Formulation 104
5.3 TITAN - Private Average Consensus 106
5.4 Private Solver for System of Linear Equations 110
5.5 Numerical Experiments 112
5.6 Analysis and Proofs 113
5.7 Conclusion 119
5.8 Proof of Remark 3 120

CHAPTER 6 SUMMARY AND FUTURE RESEARCH DIRECTIONS 121
6.1 Dissertation Summary 121
6.2 Future Research Directions 122

REFERENCES 123

LIST OF SYMBOLS

n	Number of agents or nodes in the graph
τ	Number of corrupted/adversarial agents
$\mathcal{G}\backslash\mathfrak{G}$	Communication graph topology
$\mathcal{V}$	Set of agents or nodes in the graph
$\mathcal{E}$	Set of edges in the graph
$\mathcal{A}$	Set of corrupted/adversarial agents
$\mathcal{H}$	Set of non-corrupted/honest agents
$\mathcal{N}_i^{in}(k)$	In-neighbors of agent i at iteration k
$\mathcal{N}_i^{out}(k)$	Out-neighbors of agent i at iteration k
deg_k^i	Out degree of agent i at iteration k
$\mathcal{X}$	Decision or feasible set
d	Dimension of set $\mathcal{X}$ and $\mathcal{X} \subset \mathbb{R}^d$
$f_i(x)$	Objective function of agent i
L	Gradient bound of objective function $f_i(x)$
N	Lipschitz constant of gradient $\nabla f_i(x)$
$\mathcal{X}^*$	Set of optimizers
f^*	Optimal value of $f(x)$
$\phi(k,s)$	Transition matrix product from iterations s to k
$\mathcal{P}_{\mathcal{X}}\backslash\text{proj}_{\mathcal{X}}$	Projection over set $\mathcal{X}$
$\mathcal{L}$	Graph Laplacian matrix
B	Oriented Incidence matrix
$\mathcal{N}(0,\sigma^2)$	Gaussian random variable with mean 0 and variance σ^2
$\mathcal{U}[a,b)$	Uniform random variable over $[a,b)$
$\underline{\mu}(\mathcal{L})$	Second smallest eigenvalue of matrix $\mathcal{L}$
$\kappa(\mathcal{G})$	Vertex connectivity of graph $\mathcal{G}$

$\mathcal{F}$	Set of distributed computing problems
C	Number of Clients
S	Number of Parameter Servers
α_k	Learning step-size
$f_i(x^i, \overline{x})$	Payoff function for player i
$\phi(x)$	Gradient map
$\pi(\cdot)$	Permutation map
D	Graph Degree matrix
A	Graph Adjacency matrix
$\lceil \cdot \rceil$	Ceiling function
$\mod(\cdot, \cdot)$	Modulo operator
$\delta(\mathcal{G})$	Graph diameter of graph $\mathcal{G}$

CHAPTER 1

INTRODUCTION

Distributed computing involves design and study of algorithms that allows a network of computers or devices to solve a common problem. It often fundamentally depends on message passing over computer/device networks or shared memory for communication and coordination of state variables or decisions. The information exchange, while necessary for solving the common problem, may lead to adversarial devices learning about private information stored at the computers. Consider a scenario, where an adversary corrupts or takes control of a few devices in the network. Such an adversary may store and exploit the observed information to estimate private data associated with non-corrupt devices. In this work, we are interested in designing distributed algorithms that protect private data against such adversaries while solving the problem. Distributed computing systems have become pervasive in industrial and consumer Internet of Things (IoT) networks [1], autonomous vehicles and ride-sharing applications [2], and distributed learning systems for medical [3] and financial applications [4]. These systems increasingly operate on private and sensitive information such as personal preference, medical and financial data. Consequently, we need to defended distributed computing systems against privacy attacks by adversaries. In this dissertation, we consider three fundamental distributed computing problems – (a) Distributed Optimization, (b) Network Games and (c) Average Consensus – and focus on algorithms for solving them with provable privacy properties. We briefly introduce these problems and provide motivating examples for designing privacy preserving solutions.

Distributed optimization involves a network of n devices, where each device has access to a local objective function $f_i(x)$ and intends to collectively minimize the sum of local objective functions. Each device is a computational agent connected with other devices via a communication network. That is, agents are interested in solving the aggregate optimization problem

$$\min_{x \in \mathcal{X}} \sum_{i=1}^{n} f_i(x), \tag{1.1}$$

while any agent i has access to only local objective function $f_i(x)$.

Distributed machine learning has become a very popular information processing task in recent years with applications in almost all walks of life. In a distributed machine learning scenario, partitions of the dataset are stored with several different agents such as datacenters, personal computers or mobile devices

and the loss function computed over the dataset stored at any agent i is represented by objective function $f_i(x)$. Note, x denotes a vector of model parameters and $f_i(x)$ is loss computed by agent i for its dataset given parameters x. Agents solve a distributed optimization problem in order to collaboratively learn the most appropriate "model parameters". This collaboration depends on information exchange between agents, often via average consensus or gossip mechanisms. Interaction makes agents vulnerable to privacy attacks by adversarial agents. The datasets might contain healthcare records, location history, and credit card transactions and leakage of data or its statistics can cause serious privacy breaches [5, 6]. This prompts us to consider privacy as a critical requirement while designing distributed optimization algorithms.

Network aggregate games involve a system of n strategic and non-cooperative players, where each player i is endowed with a private cost function $f_i(x^i, \overline{x})$ that depends on both local action/decision x^i and aggregate action/decision $\overline{x} = \sum_{i=1}^{n} x^i$. Players' objective is to minimize their own local cost. That is, each player i tries to solve the following optimization problem,

$$\min_{x^i \in \mathcal{X}^i} f_i(x^i, \overline{x}), \quad \text{where } \overline{x} = \sum_{i=1}^{n} x^i. \tag{1.2}$$

As the local cost function $f_i(x^i, \overline{x})$ is dependent on both local action/decision x^i and aggregate action/decision $\overline{x}$, computing an equilibrium action requires communication between players. This information exchange between the players may lead to adversarial players learning about private cost functions of other players. These local cost functions depend on private and sensitive information necessitating design of distributed equilibrium computation algorithms with in-built privacy.

Cournot competition [7] is a popular example of network aggregate game. Consider n corporations competing to offer a commodity into the market, where the price of the product is inversely proportional to the total quantity of the commodity available in the market. Corporations then solve a competitive distributed optimization problem as described above in (1.2) with the cost function for player i defined as $f_i(x^i, \overline{x}) = c_i(x^i) - x^i(a - b\overline{x})$, where $c_i(\cdot)$ represents the manufacturing cost and $(a - b\overline{x})$ denotes the price of the commodity depending only on the aggregate product available in the market $\overline{x}$. The cost $c_i(\cdot)$ depends on private and sensitive information such as manufacturing processes, supply chains/logistics and protecting the privacy of $c_i(\cdot)$ is crucial. This requires us to design Nash equilibrium computation algorithms to solve (1.2) with provable privacy guarantees.

Finally, we introduce the average consensus problem and distributed system of linear equations. We consider a system of n agents, where each agent i has access to a private input x^i and the goal is for each agent to compute $(1/n)\sum_{i=1}^{n} x^i$ while protecting the private inputs x^i. Average consensus is a fundamental primitive and has been used as an inner loop in several algorithms such as, distributed optimization [8, 9], solving a system of linear equations [10, 11], solving network aggregate games [12, 13], and low rank matrix

completion [14–17]. We also consider a horizontally partitioned system stored over a network of agents similar to [10, 11]. Under this setup, each agent has access to a few linear equations, but not the entire system of linear equations required for computing a solution. We are interested in designing a distributed linear system solver that protects the privacy of local equations yet allows the agents to compute a solution. Linear equations are fundamental in electrical network analysis, sensor networking, supply chain and logistics, filtering and reinforcement learning.

1.1 Prior Work on Privacy in Distributed Computation

Focus on privacy issues in machine learning and data analytics has led to a surge of interest in design of privacy-preserving distributed algorithms. The techniques proposed in the literature can be classified into three groups – cryptographic methods, probabilistic methods such as differential privacy and transformation based methods. While we will discuss these methods in great detail in the following chapters, we point out some key deficits in literature and motivate the need of accuracy-aware privacy mechanisms.

Cryptographic methods exploit cryptographic primitives, such as symmetric key encryption and homomorphic encryption, to encrypt private information before sharing it with neighboring agents. Use of cryptographic protocols guarantees accuracy, however, the resulting high computational cost makes it impractical for high-dimensional optimization problems. We are interested in designing computationally lightweight algorithms. Moreover, cryptographic methods rely on computational hardness of certain math problems and bounded rationality assumptions on the adversarial nodes. In this work, we focus on methods that are independent of assumptions on computational capabilities of adversarial agents.

Differential privacy is a very popular notion of probabilistic privacy method [18, 19]. Differentially private algorithms involve use of carefully designed random noise such that the distributions of information observed by adversary are similar when the algorithm is run on similar private datasets. It is a strong probabilistic privacy definition with remarkable properties such as composition lemma, post-processing lemma and group privacy. The privacy is characterized by parameter ϵ. The smaller is the ϵ, the better is the privacy guarantee. Differential privacy, however, suffers from a fundamental accuracy – privacy trade-off. A superior privacy (smaller ϵ) leads to an inferior accuracy of the computation (larger errors). Some applications demand accuracy due to their function or as a matter of regulation and this trade-off precludes use of differential privacy mechanisms. In this work, we argue that by weakening the privacy definition and exploiting communication network topology, we can circumvent this trade-off. The resulting privacy definition dubbed "Privacy via Non-identifiability" allows us to achieve both accuracy and privacy simultaneously.

Transformation methods involve transforming private problem information, while, ensuring that the

solution is preserved. These methods by design ensure accuracy of computation. However, design of transformation methods is specific to a problem and difficult to generalize to other problems. In this dissertation, we exploit communication network topology to design generic transformation techniques.

We are interested in designing computationally lightweight, accuracy-aware and easy to generalize privacy preserving distributed algorithms.

1.2 Summary of Contributions

Privacy Preserving Optimization Using Correlated Noise

In Chapter 2, we present Function Sharing (FS) methodology for private distributed optimization over time-varying, directed and undirected graphs [20, 21]. The FS strategy involves two steps:

1. *Obfuscation Step*: The Obfuscation Step involves a secure exchange of perturbation (noise) functions with immediate neighbors. The perturbation functions are arbitrarily decided by an agent and behave like noise in the functional space. Next, the private objective function $f_i(x)$ is transformed by using perturbation functions and simple secure multiparty computing strategy to obtain new perturbed objective functions $\tilde{f}_i(x)$.

2. *Distributed Optimization Algorithm*: Agents run an appropriate Distributed Optimization Algorithm, such as Gradient-Push [9] for directed graphs and Distributed Gradient Descent [8] for undirected graphs, over new perturbed objective functions $\tilde{f}_i(x)$.

We prove deterministic convergence of the iterates to an optimum of $f(x)$ (true in every execution). We provide rigorous privacy analysis of the function sharing strategy and show local objective functions are non-identifiable. We show that the information observed by the adversary is compatible with a set of distributed optimization problems. We characterize graph conditions that are necessary and sufficient to prove privacy of local objective functions. We show that our algorithm is private and simultaneously converges to the correct optimum (accuracy).

In Chapter 3, we consider distributed learning in the parameter server framework also known as the federated learning architecture. Under this setup, we consider several parameter servers that communicate with each other and clients that each have private data and share ephemeral updates with the servers. We present POLAR-SGD, Private Optimization and Learning Algorithm - Stochastic Gradient Descent, that protects privacy of client data by obfuscating the updates transmitted by the clients. It is a synchronous protocol where clients use correlated additive and multiplicative perturbations to obfuscate the stochastic gradient. These obfuscated stochastic gradients are uploaded to multiple parameter servers, who then use consensus iteration and projected stochastic gradient descent to learn predictive models. We prove convergence of

POLAR-SGD under a few conditions on the perturbations. We also discuss privacy enhancement for polynomial optimization problems. While other perturbation based algorithms such as differential privacy incur accuracy loss due to privacy, we show that our algorithm solves the optimization problem accurately.

Privately Solving Network Aggregate Games Using Correlated Noise

In Chapter 4, we propose a distributed algorithm to privately compute the Nash equilibrium in aggregate games where players communicate over a fixed undirected network [22]. Our algorithm exploits correlated perturbation to obfuscate information shared over the network. We prove that our algorithm does not reveal private information of players to an honest-but-curious adversary who monitors several nodes in the network. In contrast with differential privacy based algorithms, our method does not sacrifice accuracy of equilibrium computation to provide privacy guarantees.

Private, Finite-Time Consensus and Linear System of Equations

In Chapter 5 we present TITAN, privaTe fInite Time Average coNsensus, algorithm for solving an average consensus problem over directed graphs, while protecting statistical privacy of private local data against an honest-but-curious adversary [23]. Our algorithm uses distributed recovery primitives to solve average consensus in finite-time that is dependent only on the total number of agents and the graph diameter. We show that TITAN provides information theoretic privacy of local inputs against an honest-but-curious adversary that corrupts at most τ nodes as long as the weak vertex-connectivity of the graph is at least $\tau + 1$. We exploit TITAN to solve a system of linear equations denoted as $Az = b$, which is horizontally partitioned (rows in A and b) and stored over a network of n devices connected in a fixed directed graph. Our proposed solution involves agents computing updates based on private data followed by executing TITAN to aggregate this information quickly and privately to converge to the solution of $Az = b$. Our solver converges to the least squares solution in finite rounds along with statistical privacy of local linear equations against an honest-but-curious adversary provided the graph has weak vertex-connectivity of at least $\tau + 1$. We perform numerical experiments to validate our claims and compare our solution to the state-of-the-art methods by comparing computation, communication and memory costs.

CHAPTER 2

PRIVATE OPTIMIZATION OVER NETWORKS

2.1 Introduction

Distributed optimization involves a network of n agents that collaboratively optimize a global objective function based solely on locally available information. For instance, consider a minimization problem with the objective function defined as $f(x) \triangleq \sum_{i=1}^{n} f_i(x)$, where an agent i has access to only the local objective function $f_i(x)$. The goal is compute the optimum decision variable x^* that minimizes $f(x)$, i.e.

$$x^* \in \arg\min_{x \in \mathcal{X}} \sum_{i=1}^{n} f_i(x). \tag{2.1}$$

Distributed optimization setup has found applications in machine learning [24–27], resource allocation [28], scheduling [29, 30], robotics and smart grid optimization [31]. Distributed optimization paradigm offers us considerable benefits such as:

- Communication efficiency: As the size of model parameters or decision variable x is far smaller than the actual dataset [32, 33], sharing x over a network incurs smaller communication costs as compared to sharing the entire dataset.
- Scalability: Adding more agents or participants does not significantly change local communication, computation or storage costs [33].
- Applicability to geo-distributed datasets: Datasets that are scattered over multiple data-farms or geographically separated machines necessitate use of distributed algorithms [26, 34].

As an example, consider a distributed machine learning scenario, where partitions of the dataset are stored among several agents and the local objective function $f_i(x)$ at agent i may be a loss function computed over the dataset stored at agent i. The agents are interested in learning "model parameters" or decision variable "x" that minimizes the aggregate loss. In several applications, the optimization is performed over private and sensitive user information. The datasets might include healthcare records, location history, and credit card transactions, information that is deemed private and sensitive. Leakage of such data or its statistics can lead to privacy breaches.

Distributed optimization algorithms, however, require information exchange among agents in between successive local gradient based updates to ensure correctness. This makes them vulnerable to privacy attacks by adversarial agents and prompts us to consider privacy as a critical requirement while designing distributed optimization algorithms [5,35]. In our prior work [36], we show that by corrupting a small fraction of agents in the network, an adversary may learn the private (polynomial) objective functions of non-corrupted agents up to a constant ambiguity. In this chapter, we present a distributed optimization algorithm that protects agents' private objective function against an adversary that corrupts a bounded fraction of agents in the network, while ensuring accuracy of computed solution (2.1).

2.1.1 Related Work

Consensus based distributed optimization algorithms have become a popular choice for solving (2.1). These iterative algorithms typically involve two steps: (1) information fusion and (2) local descent [37]. In the first step, each agent fuses its own state estimate with the information (state estimates) received from neighbors via convex averaging step. The second step involves a descent step using local gradient information. Information fusion step ensures that the state estimates converge to a common value. It also ensures that the local gradient based updates effectively drive the state estimates to the optimum of global optimization problem. Many distributed optimization algorithms have appeared in the literature in recent years, including sub-gradient descent [8,38], dual averaging [39], incremental algorithms [30,40], accelerated gradient [41–43], and ADMM [42]. Solutions to distributed optimization of convex functions have been proposed for myriad scenarios involving directed graphs [9, 44], communication failures and losses [45], asynchronous communication models [46–48], and fault tolerance [49]. Distributed optimization and learning algorithms operate on increasingly private and sensitive data, consequently, research focus has shifted towards privacy requirements. A few privacy preserving distributed optimization algorithms have been proposed in recent years. In the next few paragraphs, we summarize some of these solutions. Privacy-preserving distributed optimization algorithms can broadly be categorized into the following three groups.

Cryptographic Methods: These methods rely on hardness assumptions of certain math problems and bounded computational capability assumptions on adversary to guarantee privacy. It implies that an adversary cannot recover private information from the ciphertext generated by cryptographically private method in a reasonable time frame. Cryptographic methods have been proposed to privately solve linear programming problems [50] and general data mining problems [51]. More recently, partially homomorphic encryption based methods have been explored aggressively [52–54]. Partial homomorphic encryption is an encryption/decryption mechanism that allows computations to be performed on ciphertext (encrypted input) and the resulting value when decrypted gives us the answer as if the computation (addition and multiplication) had been performed on the un-encrypted input. However, cryptographic methods often

are computationally expensive to implement [55] and are not suitable for high-dimensional optimization problems such as machine learning.

Differentially Private and Probabilistically Private Methods: One of the popular probabilistic privacy mechanisms is differential privacy. It has been used extensively in industry in recent years[1,2,3] [19,56,57]. Differential privacy involves use of specialized random perturbations to mask query result, in order to minimize the probability of uncovering of specific records in a database based on query output. More specifically, $(\epsilon, \delta)-$differential private algorithms guarantee that the ratio of distributions of observations for two adjacent databases, differing in only one entry, is upper bounded by e^{ϵ} and fails with a small probability density of δ. Differentially private distributed optimization methods have been extensively explored in recent years [35, 56, 58–62]. Specifically, [35, 56, 62] study differential privacy with regards to machine learning applications (neural networks). Prior work by researchers in [58, 60, 61] study differential privacy of local objective functions in a distributed optimization problem over peer-to-peer network, while [59] explores differential privacy in distributed optimization over client-server framework. Differential private methods, however, suffer from a fundamental trade-off between privacy margin achieved (parameter ϵ) and the accuracy ($\sim \mathcal{O}(1/\epsilon^2)$) of the computed solution [59, 61]. In this chapter, we present a weaker privacy definition for networked optimization problems, that in addition to providing reasonable privacy will circumvent this fundamental trade-off. Researchers have also explored other probabilistic notions of privacy such as (α, β) data privacy proposed in [63, 64].

Transformation based Methods: Transformation based methods are noncryptographic techniques that involve converting a given optimization problem into a new problem via algebraic transformations such that the solution of the new problem is the same as the solution of the old problem [65, 66]. This enables agents to conceal private data effectively while the quality of solution is preserved. Transformation approaches in literature, however, cater only to a relatively small class of problems and are hard to generalize. In our prior work [27], discussed in Chapter 3, we study private distributed optimization in client-server framework with multiple coordinating servers. The clients send obfuscated (noisy) gradient updates to the servers while the servers perform gradient descent over noisy updates followed by a secure consensus step. The obfuscated updates sent to multiple servers each appear to have originated from a transformed objective function such that the transformed functions add up to the private objective function. In this approach, multi-server architecture is leveraged to develop an easy-to-generalize transformation technique.

[1] https://machinelearning.apple.com/research/learning-with-privacy-at-scale
[2] https://opensource.googleblog.com/2020/06/expanding-our-differential-privacy.html
[3] https://ai.facebook.com/blog/introducing-opacus-a-high-speed-library-for-training-pytorch-models-with-differential-privacy/

2.1.2 Our Contributions

The Function Sharing Approach: We propose "Function Sharing" (FS) methodology to perform privacy preserving distributed optimization with guaranteed accuracy. The FS strategy uses correlated random perturbations to obfuscate local objective functions followed by agents executing a standard, non-private distributed optimization algorithm. We present a sketch of our approach below. The FS strategy involves:

1. *Obfuscation Step*: The Obfuscation Step involves a secure exchange of perturbation (noise) functions with immediate neighbors. The perturbation functions are arbitrarily decided by an agent and behave like random noise in the functional space. Next, agents use perturbation functions and use a "zero-sum" secret sharing protocol to generate obfuscation functions. Finally, we add obfuscation functions to the private objective function $f_i(x)$ and get new perturbed objective functions $\tilde{f}_i(x)$.

2. *Distributed Optimization Algorithm*: Agents run a standard, non-private distributed optimization algorithm – such as Gradient-Push, GP, [9] or Distributed Gradient Descent, DGD, [8] – over new perturbed objective functions, $\tilde{f}_i(x)$.

In this chapter, we present Function Sharing - Gradient Push Algorithm or FS-GP for private distributed optimization over time-varying directed graphs and Function Shring - Distributed Gradient Descent or FS-DGD for distributed optimization over undirected graphs. We present correctness, convergence and privacy analysis for both algorithms.

Correctness and Convergence Analysis: We present the correctness and convergence analysis for FS-GP and FS-DGD. We show that algorithm iterates converge to the exact optimum of the global optimization problem (2.1) in each execution (deterministically).

Observe that the perturbed objective functions $\tilde{f}_i(x)$ may be non-convex as the agents may add non-convex obfuscation functions in the obfuscation step. However, we show later in the chapter that our unique design of the obfuscation step leads to the aggregate of perturbed objective functions being exactly $f(x)$, a convex function, i.e. $\sum_{i=1}^{n} \tilde{f}_i(x) = f(x)$. It leads us to a distributed optimization problem where agents seek to minimize a convex aggregate function while only accessing potentially non-convex functions. We call this problem minimizing a convex sum of non-convex functions [67]. This problem may be of general interest as discussed in [68]. Kvaternik and Pavel in [68] solved a similar problem albeit with additional assumption on $f(x)$ being strongly convex and underlying graph being fixed and undirected. We solve this problem in the fairly general scenario: we require the aggregate $f(x)$ to be (only) convex and we consider undirected and time-varying directed graphs. We prove that Gradient-Push algorithm [9] and Distributed Gradient Descent [8] solves this problem over directed and undirected graphs respectively in order to show convergence of FS-GP and FS-DGD algorithms.

Moreover, we show that the finite-time convergence rate for FS algorithms scales gracefully with the strength of the perturbation (noise) functions [20, 69]. We characterize the finite-time convergence rate

of FS-GP and FS-DGD to be $\mathcal{O}(\log(T)/\sqrt{T})$, similar to the rates for original GP algorithm [9] and DGD algorithm [8].

Privacy Definition and Analysis: We propose *Privacy via Non-identifiability*, a practical notion of privacy for distributed computing. It involves proving that there exists a set of distributed optimization problems that all lead to the same observed execution from adversary's perspective. We provide rigorous privacy analysis of the FS-GP and FS-DGD and guarantee Privacy via Non-identifiability of local objective functions. We characterize necessary and sufficient graph connectivity conditions to prove that our algorithm satisfies this notion of privacy and protects local objective functions of non-adversarial agents. We show that our algorithms, FS-GP and FS-DGD, are private and simultaneously converge to the correct optimum (accuracy).

Functional perturbations are also used in [60] for privacy preserving distributed optimization. They examine adding differential private noise in functional space to their private objective functions resulting in perturbed objective functions. The noise functions are not correlated (as opposed to our case) and hence do not converge to the exact optimum in a deterministic sense.

Gupta in [70] further explored statistical privacy properties of Function Sharing strategy when the perturbation (noise) functions are drawn from Gaussian distribution. In [70], Gupta discussed an algorithm for protecting the privacy of affine parts of objective functions. In our joint work with Gupta [21], we generalize this analysis to protect higher order components of non-adversarial agents objective functions when solving distributed optimization over undirected graphs.

Similar to our approach, correlated perturbations are used by Liu *et al.* in [71, 72], however their method of generating correlated noise is different. They use correlated noise to hide decision variable (state) from adversaries in distributed computation at each step. We instead add perturbations to the objective function which is a pre-processing operation performed only once. Our approach hence incurs lower computational and communication overhead.

2.1.3 Organization

This chapter is organized as follows. In Section 2.2, we present the problem formulation, adversary model and privacy definitions. We introduce Function Sharing methodology in Section 2.3. We discuss FS-GP algorithm for private optimization over directed graphs in Section 2.3.1, present main results in Section 2.3.2 and provide proofs thereafter. Finally, we present FS-DGD algorithm for private optimization over undirected graphs in Section 2.4.1 followed by its main results in Section 2.4.2 and proofs.

2.2 Problem Setup: Assumptions, Adversary Model and Privacy Definition

We consider a synchronous system consisting of n agents (nodes) connected using a time-varying network of communication links. The communication links are always reliable. Each agent i has access to its own private objective function $f_i(x)$. The agents are interested in collaboratively minimizing the aggregate function $f(x)$.

$$\underset{x\in\mathbb{R}^d}{\text{minimize}}\ f(x) \triangleq \sum_{i=1}^{n} f_i(x). \tag{2.2}$$

The private objective functions $f_i : \mathbb{R}^d \to \mathbb{R}$ are convex. Let $[n] = \{1, 2, \ldots, n\}$. We optimize over $d-$dimensional space $\mathbb{R}^d$. We assume that the objective function gradients are bounded (with bound $L > 0$), i.e. for each $i \in [n]$,

$$\|\nabla f_i(x)\| \leq L. \tag{2.3}$$

The objective function gradients are Lipschitz continuous with constant $N > 0$, i.e.

$$\|\nabla f_i(x) - \nabla f_i(y)\| \leq N\|x - y\|,$$

for all $i \in [n]$ and $x \neq y$. Let f^* denote the optimal value of $f(x)$ and let $\mathcal{X}^*$ denote the set of all optima of $f(x)$, i.e.

$$f^* = \inf_x f(x) \text{ and } \mathcal{X}^* = \{x \mid x \in \mathbb{R}^d, f(x) = f^*\}.$$

We assume that the set of optimum $\mathcal{X}^*$ is nonempty. Note $\|.\|$ is Euclidean norm for vectors and the Frobenius norm for matrices.

Let the set of agents be denoted by $\mathcal{V}$; thus, $|\mathcal{V}| = n$. Define $\mathcal{E}_k$ as a set of directed edges corresponding to the communication links in the network at iteration k,

$$\mathcal{E}_k = \big\{(u, v) : u \in \mathcal{V} \text{ sends message to } v \in \mathcal{V}, \text{ at iteration } k\big\}.$$

The communication network is represented using a graph $\mathcal{G}_k = (\mathcal{V}, \mathcal{E}_k)$ at iteration k. If $\mathcal{G}_k$ is a directed graph, then $\mathcal{E}_k$ is a set of directed edges. Correspondingly, if $\mathcal{G}_k$ is undirected graph, then edge $(i, j) \in \mathcal{E}_k$ implies edge $(j, i) \in \mathcal{E}_k$ and $\mathcal{E}_k$ becomes a set of undirected edges.

We define the neighborhood sets for a node i. The out-neighborhood set, $\mathcal{N}_i^{out}(k)$, is the set of all nodes that directly receive messages from node i (at iteration k). The in-neighborhood set, $\mathcal{N}_i^{in}(k)$ is the set of all nodes that send messages to node i. We include the node i in its own in-neighborhood and out-neighborhood

sets. Specifically,

$$\mathcal{N}_i^{out}(k) = \{j \mid (i,j) \in \mathcal{E}_k\} \cup \{i\} \quad \text{and} \quad \mathcal{N}_i^{in}(k) = \{j \mid (j,i) \in \mathcal{E}_k\} \cup \{i\}.$$

As $(i,i) \notin \mathcal{E}_k$, we include it in the neighborhood sets individually. The out degree of a node i at iteration k is defined as $deg_k^i = |\mathcal{N}_i^{out}(k)|$. We assume that every node i knows its out degree deg_k^i at iteration k. For undirected graphs, the in-neighborhood and out-neighborhood sets are the same, i.e. $\mathcal{N}_i^{out} = \mathcal{N}_i^{in}$ and similarly the in-degree and out-degree are the same.

We will assume connectivity conditions of the graph $\mathcal{G}_0$ (considered for obfuscation step at iteration 0) that are necessary and sufficient for guaranteeing privacy in Sections 2.3.2 and 2.4.2. We will impose additional connectivity conditions on $\mathcal{G}_k$ for convergence analysis in Sections 2.3.2 and 2.4.2.

2.2.1 The Adversary Model

In this work we consider honest-but-curious adversary that is fairly common in the privacy literature [58–60]. Honest-but-curious adversaries are passive adversaries that capture, store and exploit observations to uncover information private to other nodes. However, adversaries follow the protocols as prescribed. In this work, we consider an honest-but-curious adversary, denoted by A, that can corrupt at most τ agents in the network. Adversaries have access to all the states of the system, the graph topology and messages directly shared with the corrupted agents. We will denote the set of corrupted nodes as $\mathcal{A}$ and the set of non-corrupted nodes as $\mathcal{H} = \mathcal{V} \setminus \mathcal{A}$. As discussed above, $|\mathcal{A}| \leq \tau$. We consider privacy in synchronous setting which is often more difficult than asynchronous setting where the unpredictability of messages may make it difficult for adversary to break privacy.

2.2.2 Privacy Definition

We begin by describing a distributed optimization problem as a collection of local objective functions. Hence, any set of n appropriate local objective functions defines a distributed optimization problem. Let $\mathcal{F}$ denote a set of distributed optimization problems, each characterized by a collection of n objective functions that satisfy – (a) convexity, (b) bounded gradients, (c) Lipschitz continuous gradients and (d) add to $f(x)$. We write $\mathcal{F}$ as

$$\mathcal{F} = \Big\{\{g_1(x), \ldots, g_n(x)\} | g_i(x) \text{ is convex and gradients } \nabla g_i(x) \text{ are bounded and Lipschitz continuous, and } \sum_{i=1}^{n} g_i(x) = f(x)\Big\}.$$

Each element of set $\mathcal{F}$, say $\{g_1(x), \ldots, g_n(x)\}$ corresponds to an instance of distributed optimization problem, where the $g_i(x)$ is the local objective functions for each agent i and the aggregate objective function is $f(x)$.

Intuitively, we define privacy as the inability of the adversary to exactly guess the private objective function given the adversary's observation of the algorithm execution. In our scenario, we claim that the observations made by the adversary are compatible with each instance of distributed optimization problems in set $\mathcal{F}$.

Definition 1 (Privacy via Non-identifiability)**.** *If adversarial observations are same under two separate executions, one with private objective functions $f_i(x)$ and the other with private objective functions $g_i(x) \neq f_i(x)$ for at least one $i \in [n]$, then we state adversary's observations are compatible with a problem instance $\{g_1(x), \ldots, g_n(x)\} \in \mathcal{F}$. A distributed optimization algorithm satisfies privacy via non-identifiability, if the information observed by the adversaries is compatible with any $\{g_i(x) | i \in [n]\} \in \mathcal{F}$.*

2.3 Function Sharing Algorithm for Directed Networks

In this section, we will present the Function Sharing methodology. We will also present Function Sharing - Gradient Push algorithm (FS-GP) for privacy-preserving distributed optimization on directed graphs, along with key results and proofs.

The Function Sharing strategy is inspired by the so called Secure-Sum protocol proposed in [73] for private aggregate computation over fully connected networks and generalized for private average consensus over incomplete undirected graphs in [70]. These protocols involve using additive perturbations to hide private information before running an distributed aggregation or averaging protocol. The perturbations are carefully designed to add to zero over the network and this guarantees correctness of secure-sum protocol. Distributed optimization problems also exhibit such an additive sub-structure and this motivates us to exploit secure-sum type protocols for securing private information in distributed optimization.

The Function Sharing strategy, as alluded to in Section 2.1.2, is a two-step protocol. First, we perform a pre-processing step that transforms private objective functions $f_i(x)$ to new perturbed objective functions $\tilde{f}_i(x)$ (Obfuscation Step). This is followed by appropriate distributed optimization algorithm run by agents where each agent uses its new perturbed objective function $\tilde{f}_i(x)$ as the objective function (Distributed Optimization). We will detail each of the steps below.

We first present the obfuscation step that transforms $f_i(x)$ to $\tilde{f}_i(x)$. Initially, perturbation functions are shared with neighboring agents in a secure manner possibly using encrypted messages or physical layer encryption. Note that this is the only step that requires additional security measures. More precisely, each agent i sends perturbation function $R_{ij}(x)$ to each out-neighbor $j \in \mathcal{N}_i^{out}$ in a secure fashion.[4] Consequently, agent i also receives perturbation function $R_{li}(x)$ from all in-neighbors $l \in \mathcal{N}_i^{in}$. The perturbation functions are arbitrary and randomly picked functions satisfying gradient boundedness and Lipschitz continuous

[4]The obfuscation step is a pre-processing step that happens at $k = 0$. For brevity we will drop the time index from the notation for neighborhood sets $\mathcal{N}_i^{in}$ and $\mathcal{N}_i^{out}$, when discussing the neighborhood sets for obfuscation.

gradients.

Next, each agent i computes perturbed objective function $\tilde{f}_i(x)$ by taking the private objective function $f_i(x)$, adding all the received perturbation functions $(R_{li}(x), l \in \mathcal{N}_i^{in})$ and subtracting all the transmitted perturbation functions $(R_{ij}(x), j \in \mathcal{N}_i^{out})$. That is for each $i \in [n]$, we have

$$\tilde{f}_i(x) = f_i(x) + \sum_{l \in \mathcal{N}_i^{in}} R_{li}(x) - \sum_{j \in \mathcal{N}_i^{out}} R_{ij}(x). \tag{2.4}$$

This point onward (in the execution of the algorithm), each agent i uses the perturbed objective function $\tilde{f}_i(x)$ in lieu of its private objective function $f_i(x)$.

The obfuscation step is followed by agents running appropriate and well known distributed optimization algorithms such as Gradient-Push [9] and Distributed Gradient Descent [8]. These algorithms use only the gradients of perturbed objective functions $\tilde{f}_i(x)$ to update local iterates followed by interleaved consensus based averaging. We discuss specific algorithms in the next subsection.

Recall that we use perturbed objective functions in lieu of private objective functions and as such need to relate perturbed objective functions to the original problem (2.1). We make the following key observations that help establish this link and aid us in convergence analysis. First we show that, the aggregate objective function $f(x)$ is invariant under the obfuscation step. Specifically,

$$\sum_{i=1}^{n} \tilde{f}_i(x) = \sum_{i=1}^{n} f_i(x) + \sum_{i=1}^{n} \left(\sum_{l \in \mathcal{N}_i^{in}} R_{li}(x) - \sum_{j \in \mathcal{N}_i^{out}} R_{ij}(x) \right) = \sum_{i=1}^{n} f_i(x) \triangleq f(x). \tag{2.5}$$

Observe that each perturbation function $R_{ij}(x)$ is added to the objective function by an agent (in this case j) and subtracted from the objective function by another agent (in this case i). It ensures that while computing sum of all perturbed objective functions $\tilde{f}_i(x)$, the effect of perturbation (noise) functions get canceled and the sum equals the sum of private objective functions $f(x)$. We use this property to establish convergence. The aggregate invariance property is key to ensure that Function Sharing based protocols guarantee privacy and correctness (accuracy) simultaneously.

The perturbed objective function $\tilde{f}_i(x)$ may be non-convex and the aggregate invariance property above only informs us that $\sum_{i=1}^{n} \tilde{f}_i(x) = f(x)$. Hence, the distributed optimization algorithm employed in Function Sharing methodology (the second step) needs to distributedly minimize $f(x)$ where each agent i has access to possibly non-convex function $\tilde{f}_i(x)$. We call this problem *minimizing a convex sum of non-convex functions* [67]. Conventional convergence analysis for GP and DGD algorithms however does not work for this scenario as they crucially depend on each local function being convex. Here we cannot do that, but there is hope, in that the aggregate objective function is convex and this structure can be exploited for convergence.

Kvaternik and Pavel [68] addressed this problem albeit with stronger assumptions such as the aggregate function being strongly convex and the graph being fixed and undirected. We consider reasonably weak assumptions and require $f(x)$ to be only convex and allow the graph to be time-varying and directed.[5] Moreover, we show how this problem unintentionally leads to privacy in distributed optimization.

We observe, proving convergence of privatized (via obfuscation step) distributed optimization algorithm is tantamount to the original distributed optimization algorithm being able to minimize a convex sum of non-convex functions. We illustrate this while proving that FS-GP and FS-DGD (specific algorithms described later) solves problem (2.1) and provides privacy via non-identifiability.

2.3.1 Function Sharing Algorithm for Directed Graph Scenario

We first consider the scenario where agents are connected in a time-varying, directed graphs. We propose Function Sharing - Gradient Push (FS-GP, Algorithm 1) protocol for privacy preserving distributed optimization over time-varying, directed networks. Our algorithm, FS-GP, essentially involves two steps. In the first step, we perform obfuscation step to transform private objective functions, $f_i(x)$, to perturbed objective functions, $\tilde{f}_i(x)$ as described in (2.4). In the second step, we run Gradient-Push protocol, GP, proposed in [9].

In line 1 (Algorithm 1) each node i sends function $R_{ij}(x)$ to out-neighbors $j \in \mathcal{N}_i^{out}$ in a secure manner. The noise functions $R_{ij}(x)$ are arbitrary functions satisfying gradient boundedness and Lipschitz continuous gradients. Next (line 2, Algorithm 1), each node i obfuscates its private objective function $f_i(x)$ by adding all received noise functions $R_{ji}(x)$ ($j \in \mathcal{N}_i^{in}$) and subtracting all transmitted noise functions $R_{il}(x)$ ($l \in \mathcal{N}_i^{out}$) as defined in (2.4). This leads us to perturbed objective function, $\tilde{f}_i(x)$, stored at agent i ($\forall i \in [n]$).

This is followed by each agent running the GP algorithm [9]. Each agent i stores states $w_k^i, x_k^i, z_k^i \in \mathbb{R}^d$ and scalar variable y_k^i. The decision variable is initialized to a random value and the scalar variable is initialized to 1 (i.e. $y_0^i = 1$) for each agent i (line 3, Algorithm 1).

Next each agent i receives x_k^j/deg_k^j and y_k^j/deg_k^j that is broadcast by neighbors $j \in \mathcal{N}_i^{in}(k)$. Agent i updates w_k^i by adding x_k^j/deg_k^j and updates y_k^i by adding y_k^j/deg_k^j received from in-neighbors (lines 5-6, Algorithm 1). The next two steps do not require any communication with neighbors. Agents first performs a scaling operation to compute z_{k+1}^i (line 7, Algorithm 1). This scaled version of state z_k^i converges to a common point reaching average consensus. This is followed by a local gradient descent operation (line 8, Algorithm 1) that uses the gradient of perturbed objective function computed at the scaled state z_k^i. Note that the step sizes α_k, $k \geq 1$, form a non-increasing sequence such that $\sum_{k=1}^{\infty} \alpha_k = \infty$ and $\sum_{k=1}^{\infty} \alpha_k^2 < \infty$.

Prior results show that for convex objective functions, the scaled variables z_{k+1}^i gets pushed towards each other while the local gradient based update forces iterate z_{k+1}^i toward the optimum of aggregate function $f(x)$. In Section 2.3.4, we will show that z_{k+1}^i converges to minimizer of $f(x)$ while only accessing non-convex

[5] We separately consider the scenario where we allow the communication network to be an undirected graph. We report results in our technical report [67].

Algorithm 1 Function Sharing - Gradient Push, FS-GP

Input: Each node i has access to $f_i(x)$.
Result: Iterates converge to $x^* \in \arg\min_{x\in\mathbb{R}^d} \sum_{i=1}^n f_i(x)$

$\Diamond$ **Obfuscation Step**

Each node i sends function $R_{ij}(x)$ to out neighbors $j \in \mathcal{N}_i^{out}$

Each node i perturbs its objective function to get new objective function $\tilde{f}_i(x)$

$$\tilde{f}_i(x) = f_i(x) + \sum_{j\in\mathcal{N}_i^{in}} R_{ji}(x) - \sum_{l\in\mathcal{N}_i^{out}} R_{il}(x)$$

$\Diamond$ **Gradient-Push Algorithm** is run at each node.

Initialize: $x_0^i \in \mathcal{X}$ and $y_0^i = 1$ for each node i

for Iteration number $k = 0, 1, 2, \ldots$ **do**

  Update: $w_{k+1}^i = \sum_{j\in\mathcal{N}_i^{in}(k)} \frac{x_k^j}{deg_k^j}$

  Update: $y_{k+1}^i = \sum_{j\in\mathcal{N}_i^{in}(k)} \frac{y_k^j}{deg_k^j}$

  Update: $z_{k+1}^i = \frac{w_{k+1}^i}{y_{k+1}^i}$

  Update: $x_{k+1}^i = w_{k+1}^i - \alpha_{k+1}\nabla\tilde{f}_i(z_{k+1}^i)$

end for

functions $\tilde{f}_i(x)$.

2.3.2 Main Results and Discussion

We present the following three types of results – correctness, privacy via non-identifiability, and convergence rate in this subsection followed by some discussion.

Correctness Results: Our correctness result guarantees that Algorithm 1 iterates converge to the optimizer of $f(x)$ provided that assumptions in Section 2.2 are satisfied. We also require that the directed graph is B-strongly connected, implying that for some integer $B > 0$, the graph $(\mathcal{V}, \cup_{k=tB}^{(t+1)B-1}\mathcal{E}_k)$ is strongly connected for every t.

Theorem 1 (Correctness)**.** *Consider a distributed optimization problem satisfying assumptions from Section 2.2. Assume that the graph $\mathcal{G}_k$ is B-strongly connected. Iterates z_k^j for FS-GP (Algorithm 1) converge to an optimum $x^* \in \mathcal{X}^*$ asymptotically.*

Theorem 1 guarantees that the sequence of iterates generated by algorithm FS-GP converge to the optimum. It outlines the fact that, although agents use perturbed objective functions, the agent states reach the correct optimum. This can be credited to both, the push-sum based information fusion that averages information over the network and the fact that aggregate objective function is invariant under the obfuscation step (2.5). Theorem 1 guarantees that privacy does not impact the correctness of the algorithm (accuracy).

Privacy Results: We consider a set $\mathcal{C}$ of allowed objective functions that is closed under addition. We select noise/perturbation functions $R_{ij}(x)$ also belong to the set of allowed functions. This ensures the perturbed objective functions $\tilde{f}_i(x) \in \mathcal{C}$. The set of all feasible problems, $\mathcal{F}$, is expressed as,

$$\mathcal{F} = \Big\{ \{g_i(x), \forall i \in [n]\} \;\Big|\; g_i(x) \text{ is convex, gradients } \nabla g_i(x) \text{ are bounded and Lipschitz continuous and} \sum_{i=1}^{n} g_i(x) = f(x) \Big\}.$$

The obfuscation step ensures adversaries observe the obfuscated objective functions $\tilde{f}_i(x)$ and not the private functions $f_i(x)$. Definition 1 states that an algorithm is private, if the adversarial observations are compatible for all instances of optimization problem in set $\mathcal{F}$.

Theorem 2. *(Privacy via Non-identifiability) Let the communication graph at time zero be denoted as $\mathcal{G}_0$. For FS-GP (Algorithm 1), the following statements hold.*

(P1) For a non-corrupted node i, $|\mathcal{N}_i^{in}(0) \cup \mathcal{N}_i^{out}(0) \setminus \{i\}| \geq \tau + 1$ is necessary and sufficient for privacy of $f_i(x)$ as per Definition 1. In other words, the adversary cannot learn function $f_i(x)$.

(P2) Assume V is a strict subset of non-corrupt nodes and $|\mathcal{A}| \leq \tau$, then

$$\Big| \big(\cup_{i \in V} \mathcal{N}_i^{in}(0)\big) \cup \big(\cup_{i \in V} \mathcal{N}_i^{out}(0)\big) \setminus V \Big| \geq \tau + 1, \quad \forall\, V \subset \mathcal{H},$$

is necessary and sufficient for privacy of $\sum_{i \in V} f_i(x)$ as per Definition 1. In other words, the adversary cannot learn $\sum_{i \in V} f_i(x)$.

The graph connectivity conditions in Theorem 2 is intuitive. It essentially requires that any agent or group of agents (that needs to be protected against adversaries) needs to have at least $\tau + 1$ neighbors. This allows τ of the neighbors to be adversaries and still possess a non-adversarial neighbor. The inability of adversary to figure out perturbation (noise) function corresponding to communication link connecting the group to its non-adversarial neighbor provides privacy.

Theorem 2 characterizes the necessary and sufficient conditions for privacy via non-identifiability. However, even if the graph conditions are satisfied there are two important privacy limitations that need to be discussed. We cannot protect the privacy of aggregate objective function $f(x)$. This is a fundamental limitation due to the correctness property of our algorithm. Adversary may estimate $f(x)$ by observing progress of the algorithm. They also have access to their own private objective functions $\sum_{i \in \mathcal{A}} f_i(x)$. Using these two quantities, adversary can estimate the aggregate objective function of all non-adversarial nodes, $\sum_{i \in \mathcal{H}} f_i(x)$. Observe that Theorem 2 allows for privacy-preservation of cumulative objective functions of a set of nodes, however, such a set has to be a strict subset of non-adversarial nodes.

Finite-Time Convergence Rate: The FS-GP algorithm provides privacy while guaranteeing accuracy, however, it also degrades the convergence rate. For a learning rate of $\alpha_k = 1/\sqrt{k}$, in the following result, we bound the finite time convergence rate of the function value at a running average $\widehat{z}_k^j$.

Theorem 3. *Let estimates $\{z_k^j\}$ be generated by FS-GP respectively with $\alpha_k = 1/\sqrt{k}$. For each $j \in \mathcal{V}$, let $\widehat{z}_{T+1}^j = \frac{\sum_{k=0}^{T} \alpha_{k+1} z_{k+1}^j}{\sum_{k=0}^{T} \alpha_{k+1}}$. Then,*

$$f(\widehat{z}_{T+1}^j) - f(x^*) \leq \mathcal{O}\left(\tilde{L}^2 \frac{\log(T+1)}{\sqrt{T+1}}\right).$$

Theorem 3 presents the finite time convergence rate. It provides a bound on the sub-optimality of function value at $\widehat{z}_T^j$. The rate $\mathcal{O}(\log(T)/\sqrt{T})$ is similar to that of original distributed GP [9]. However, $\tilde{L}^2$ which is the gradient bound over perturbed objective functions $\tilde{f}_i(x)$ appears instead of L^2 (as seen in GP). The perturbed objective functions have a larger gradient bound $\tilde{L}$ due to the noise functions that are added to the private objective functions. This creates a slowdown in finite-time convergence and represents the price of privacy in distributed optimization.

2.3.3 Proof of Theorem 2

Recall that function sharing based algorithm involves an obfuscation step that transforms private objective functions $f_i(x)$ to perturbed objective functions $\tilde{f}_i(x)$ using noise functions $R_{ij}(x)$ as seen in (2.4). The selection of noise functions $R_{ij}(x)$ has an impact on how effectively $f_i(x)$ can be hidden. As an example consider that the objective function is a quadratic function and all noise functions are affine. Under this scenario the affine noise functions cannot hide the coefficients of quadratic terms of the private objective functions and result in privacy breach. Hence, we require that both $f_i(x)$ and $\tilde{f}_i(x)$ both appear similar.

Formally, let us consider that the set of allowed private objective functions $\mathcal{C}$ and addition operator ("+") forms an additive group [74]. Now, we pick the noise functions $R_{ij}(x)$ from the set of allowed private objective function $\mathcal{C}$. This ensures that the perturbed private objective functions $\tilde{f}_i(x)$, resulting from addition between $f_i(x)$ and several $R_{ij}(x)$ also belong to the set of allowed functions $\mathcal{C}$. This ensures that $f_i(x)$ and $\tilde{f}_i(x)$ appear similar as discussed above.

We prove privacy via non-identifiability for Function Sharing approach or FS-GP algorithm (Theorem 2). We use contradiction to prove necessity of graph connectivity conditions. On the other hand, we provide a constructive method to show multiple instances of distributed optimization problem in $\mathcal{F}$ would lead to the exact same algorithm execution under the connectivity condition on $\mathcal{G}_0$. This proves the sufficiency of connectivity conditions on $\mathcal{G}_0$ for privacy via non-identifiability.

Proof. **Necessity (P2):** Recall that $\mathcal{A}$ represents the set of corrupted agents, $\mathcal{H}$ represents the set of non-corrupted agents and $V \subset \mathcal{H}$. We will prove the necessity of condition in P2 using contradiction. We assume that, $\left| \left(\cup_{i \in V} \mathcal{N}_i^{in}(0)\right) \cup \left(\cup_{i \in V} \mathcal{N}_i^{out}(0)\right) \setminus V \right| \leq \tau$.

First observe that $\left(\cup_{i\in V}\mathcal{N}_i^{in}(0)\right)$ is the set of all in-neighbors of nodes in set V, and $(\cup_{i\in V}\mathcal{N}_i^{out}(0))$ is the set of all out-neighbors of nodes in set V.

The set $\left(\left(\cup_{i\in V}\mathcal{N}_i^{in}(0)\right) \cup (\cup_{i\in V}\mathcal{N}_i^{out}(0)) \setminus V\right)$ includes all nodes that communicate with nodes in V. Our assumption states that there are at most τ nodes in this set.

Consider that each node in $\left(\left(\cup_{i\in V}\mathcal{N}_i^{in}(0)\right) \cup (\cup_{i\in V}\mathcal{N}_i^{out}(0)) \setminus V\right)$ is an adversary. This is plausible scenario since we allow at most τ adversaries. In this scenario, the adversaries have access to all the perturbation functions $R_{ij}(x)$ and $R_{ji}(s)$ where $i \in V$ and $j \notin V$. Under the adversary model, we assume that the adversary can estimate $\sum_{i\in V}\tilde{f}_i(x)$ by observing the execution of the algorithm. Consider the cumulative objective, $\sum_{i\in V} f_i(x)$,

$$\begin{aligned}\sum_{i\in V}\tilde{f}_i(x) &= \sum_{i\in V}\left(f_i(x) + \sum_{j\in\mathcal{N}_i^{in}} R_{ji}(x) - \sum_{l\in\mathcal{N}_i^{out}} R_{il}(x)\right)\\ &= \sum_{i\in V} f_i(x) + \sum_{i\in V}\left(\sum_{j\in\mathcal{N}_i^{in}\setminus V} R_{ji}(x) - \sum_{l\in\mathcal{N}_i^{out}\setminus V} R_{il}(x)\right). \end{aligned} \tag{2.6}$$

Observe that in (2.6), the adversary has access to both the perturbed objective functions $\sum_{i\in V}\tilde{f}_i(x)$ and the perturbation functions shared by the nodes in set V. Hence, the adversaries can estimate $\sum_{i\in V} f_i(x)$. The privacy of group cumulative objective function $\sum_{i\in V} f_i(x)$ is breached. This gives us the contradiction. Hence $\left|\left(\cup_{i\in V}\mathcal{N}_i^{in}(0)\right) \cup (\cup_{i\in V}\mathcal{N}_i^{out}(0)) \setminus V\right| \geq \tau+1$ is necessary.

Sufficiency (P2): We use a constructive approach to prove the sufficiency of the condition. We will show that provided the condition is satisfied, there exist several instances of problem that can result in the exact same observed execution.

First, observe that the condition $\left|\left(\cup_{i\in V}\mathcal{N}_i^{in}(0)\right) \cup (\cup_{i\in V}\mathcal{N}_i^{out}(0)) \setminus V\right| \geq \tau+1$ (for all $V \subset \mathcal{H}$) ensures that any strict subset of the good (non-adversarial) nodes have more than $\tau+1$ neighbors (both in-neighbors and out-neighbors combined). Hence, V will have at least 1 non-adversarial neighbor (since we allow at most τ adversaries). This ensures that the graph obtained by deleting adversarial nodes and incident edges is weakly connected.

Under the adversary model, we conservatively assume that adversaries can estimate $\tilde{f}_i(x)$ for all $i \in [n]$. Adversaries also have access to the private objective function of all adversarial nodes $f_i(x)$ for $i \in \mathcal{A}$ and the perturbation functions received and transmitted by adversaries $R_{ij}(x)$ when $i \in \mathcal{A}$ or $j \in \mathcal{A}$. The adversaries are also aware of the function obfuscation step performed by nodes (2.4).

Consider two instances of distributed optimization problem, P_1 and P_2, with private objective functions $f_i(x)(i \in [n])$ and $f_i^0(x)(i \in [n])$ respectively, such that, $\sum_{i=1}^n f_i(x) = \sum_{i=1}^n f_i^0(x)$. Specifically, P_1, $P_2 \in \mathcal{F}$. We have two sets of noise functions $R_{ij}(x)(\forall(i,j) \in \mathcal{E}_0)$ for P_1 and $G_{ij}(x)(\forall(i,j) \in \mathcal{E}_0)$ for P_2, such that

they lead to the exact same execution. Implying,

$$\begin{aligned}\tilde{f}_i(x) &\triangleq f_i(x) + \sum_{j \in \mathcal{N}_i^{in}} R_{ji}(x) - \sum_{l \in \mathcal{N}_i^{out}} R_{il}(x) \\ &\triangleq f_i^0(x) + \sum_{j \in \mathcal{N}_i^{in}} G_{ji}(x) - \sum_{l \in \mathcal{N}_i^{out}} G_{il}(x). \end{aligned} \tag{2.7}$$

Next, we elaborate a method to compute the noise functions $G_{ij}(x)$ given $\tilde{f}_i(x)$, $f_i^0(x)$ and $R_{ij}(x)$, such that, P_1 and P_2 will have same execution.

Note that since adversaries observe $R_{ij}(x)$ whenever i or $j \in \mathcal{A}$, and hence, $G_{ij}(x) = R_{ij}(x)$ if $i \in \mathcal{A}$, $j \in \mathcal{A}$ or both $i, j \in \mathcal{A}$.

1. First we consider the graph with the adversarial nodes and incident edges removed. Let us call this graph $\mathcal{G}_{\mathcal{H}}$, with vertex set $\mathcal{H}$ and edge set $\mathcal{E}_{\mathcal{H}}$.

2. Characterize the weakly connected graph $\mathcal{G}_{\mathcal{H}}$.

   (a) We call a graph $\mathcal{Q} = (\mathcal{V}_{\mathcal{Q}}, \mathcal{E}_{\mathcal{Q}})$ a spanning pseudo-tree of $\mathcal{G}_{\mathcal{H}}$, if it satisfies:

   - $\mathcal{V}_{\mathcal{Q}} = \mathcal{H}$.
   - $|\mathcal{E}_{\mathcal{Q}}| = |\mathcal{V}_{\mathcal{Q}}| - 1$ and $\mathcal{E}_{\mathcal{Q}} \subseteq \mathcal{E}_{\mathcal{H}}$.
   - $(\mathcal{V}_{\mathcal{Q}}, \mathcal{E}_{\mathcal{Q}} \cup -\mathcal{E}_{\mathcal{Q}})$ is an undirected tree graph.

   Note $-\mathcal{E}_{\mathcal{Q}}$ is same set of edges as $\mathcal{E}_{\mathcal{Q}}$ although with directions reversed and $\mathcal{Q}$ is a directed graph. An alternate way to define $\mathcal{Q}$ can be to select $\mathcal{E}_{\mathcal{Q}}$ be the set of directed edges in $\mathcal{G}_{\mathcal{H}}$ such that (i) at most one directed edge is chosen between any pair of node, and (ii) when their directions are ignored, they form a spanning tree of $\mathcal{G}_{\mathcal{H}}$.

   (b) By our assumption, the graph connectivity condition ensures that $\mathcal{G}_{\mathcal{H}}$ is a weakly connected graph. Hence, $(\mathcal{H}, \mathcal{E}_{\mathcal{H}} \cup -\mathcal{E}_{\mathcal{H}})$ is a connected undirected graph. We know every connected undirected graph $(\mathcal{H}, \mathcal{E}_{\mathcal{H}} \cup -\mathcal{E}_{\mathcal{H}})$ has a spanning tree. Identify the edges of that tree $\mathcal{E}_{tree}$. We identify the edges $\mathcal{E}_{\mathcal{Q}}$ to be subset of $\mathcal{E}_{tree} \cap \mathcal{E}_{\mathcal{H}}$. This follows the fact that we are interested in $\mathcal{Q}$ being a directed graph.

   (c) Identify a spanning pseudo-tree $\mathcal{Q}$, as defined above in (a), of $\mathcal{G}_{\mathcal{H}}$.

3. Design $G_{ij}(x)$ corresponding to the edges from $\mathcal{G}_{\mathcal{H}}$.

   (a) Design $G_{ij}(x)$ corresponding to edges in $\mathcal{G}_{\mathcal{H}}$ that are not in $\mathcal{E}_{\mathcal{Q}}$.

   - Select these $G_{ij}(x)$ to be arbitrary selected noise functions similar to the objective functions. They can be randomly picked functions with bounded and Lipschitz gradients.

   (b) Design $G_{ij}(x)$ corresponding to edges $\mathcal{E}_{\mathcal{Q}}$ of pseudo-tree $\mathcal{Q}$.

   - Observe that $|\mathcal{E}_{\mathcal{Q}}| = |\mathcal{V}_{\mathcal{Q}}| - 1$. In the event that $\mathcal{Q}$ has only two nodes, one node is the leaf node and the other is root node.

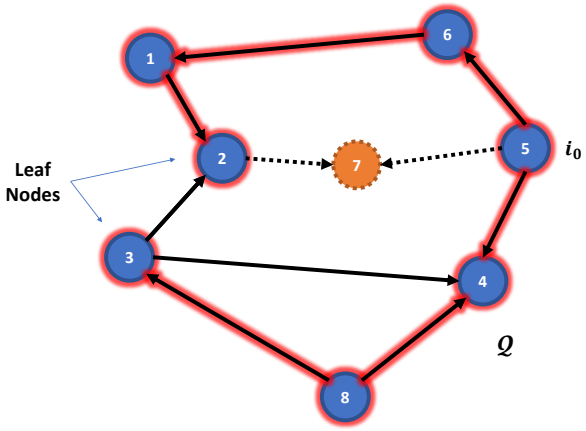


Figure 2.1: An example of construction used in proof. Consider a directed graph $\mathcal{G}_0$ with $n = 8$ nodes and $\tau = 1$ adversary (node 7). Step 1: Residual graph ($\mathcal{H}$) is depicted in bold (the dotted edges and node are adversarial and deleted from $\mathcal{G}_0$). Step 2: Construct spanning pseudo-tree $\mathcal{Q} = (\mathcal{V}_\mathcal{Q}, \mathcal{E}_\mathcal{Q})$ where $\mathcal{V}_\mathcal{Q} = \{1, 2, 3, 4, 5, 6, 8\}$, $\mathcal{E}_\mathcal{Q} = \{(8, 4), (5, 4), (5, 6), (6, 1), (1, 2), (8, 3)\}$) with root node $i_0 = 5$ and leaf nodes $\{2, 3\}$. Step 3 (a): Randomly pick $G_{34}(x)$, $G_{32}(x)$. Step 3 (b): Compute $G_{ij}(x)$ corresponding to the spanning pseudo-tree $\mathcal{Q}$ as per Step 3 (b).

- We start with the leaf nodes of our spanning pseudo-tree.
- For each leaf node, the noise function corresponding to only one incident edge is undecided. We decide the noise function for this link by solving (2.4) for the leaf node. Specifically consider i to be the leaf node and j be its parent with a link connecting j to i.

$$G_{ji}(x) = \tilde{f}_i(x) - f_i(x) - \sum_{l \in \mathcal{N}_i^{in} \setminus \{j\}} G_{li}(x) + \sum_{j \in \mathcal{N}_i^{out}} G_{ij}(x).$$

  The expression above leads to an exact $G_{ji}(x)$. This follows from the fact that $\sum_{l \in \mathcal{N}_i^{in} \setminus \{j\}} G_{li}(x)$ and $\sum_{j \in \mathcal{N}_i^{out}} G_{ij}(x)$ can be computed as noise functions $G_{li}(x)$ $(l \neq j)$ and $G_{ij}(x)$ is decided for each j.
- Once we have performed the above mentioned process for each leaf node, we move on to the parent nodes for each of the leaf nodes. Observe that these parent nodes also have only one incident edge that does not have an assigned noise function. We repeat the above step, i.e. decide on the noise function by solving (2.4) for the parent node.
- We keep following the procedure, recursively, until we reach the root node of the spanning pseudo-tree. At the root node, there are no more noise functions that are left to be computed. So we need to show that the noise functions selected in prior steps satisfy (2.4) at the root node.

(c) We will now show that the noise functions decided by our construction method satisfy (2.4) at the root node. We begin by using the fact that (2.4) is satisfied at each node other than root node i_0.

Aggregating the expression for each node other than i_0 we get,

$$\sum_{i\neq i_0}\tilde{f}_i(x)=\sum_{i\neq i_0}f_i^0(x)+\sum_{i\neq i_0}\left(\sum_{l\in\mathcal{N}_i^{in}}G_{li}(x)-\sum_{j\in\mathcal{N}_i^{out}}G_{ij}(x)\right).$$

Next, we add and subtract the effective noise function at node i_0, i.e. $\tilde{f}_{i_0}-f_{i_0}$, on right-hand side of the above expression,

$$\sum_{i\neq i_0}\tilde{f}_i(x)=\sum_{i\neq i_0}f_i^0(x)+\underbrace{\sum_{i\neq i_0}\left(\sum_{l\in\mathcal{N}_i^{in}}G_{li}(x)-\sum_{j\in\mathcal{N}_i^{out}}G_{ij}(x)\right)+\left(\sum_{l\in\mathcal{N}_{i_0}^{in}}G_{li_0}(x)-\sum_{j\in\mathcal{N}_{i_0}^{out}}G_{i_0j}(x)\right)}_{=0}$$
$$-\left(\sum_{l\in\mathcal{N}_{i_0}^{in}}G_{li_0}(x)-\sum_{j\in\mathcal{N}_{i_0}^{out}}G_{i_0j}(x)\right).$$

Recall that the aggregate of effective noise function at all nodes is exactly zero, i.e. $\sum_i(\tilde{f}_i-f_i)=0$.

$$\begin{aligned}
\sum_{i\neq i_0}\tilde{f}_i(x)&=\sum_{i\neq i_0}f_i^0(x)-\left(\sum_{l\in\mathcal{N}_{i_0}^{in}}G_{li_0}(x)-\sum_{j\in\mathcal{N}_{i_0}^{out}}G_{i_0j}(x)\right)\\
&=\sum_{i\neq i_0}f_i^0(x)-\left(\sum_{l\in\mathcal{N}_{i_0}^{in}}G_{li_0}(x)-\sum_{j\in\mathcal{N}_{i_0}^{out}}G_{i_0j}(x)\right)+f_{i_0}^0(x)-f_{i_0}^0(x)\\
&=\underbrace{\sum f_i^0(x)}_{f(x)}-f_{i_0}^0(x)-\left(\sum_{l\in\mathcal{N}_{i_0}^{in}}G_{li_0}(x)-\sum_{j\in\mathcal{N}_{i_0}^{out}}G_{i_0j}(x)\right)\\
&=f(x)-f_{i_0}^0(x)-\left(\sum_{l\in\mathcal{N}_{i_0}^{in}}G_{li_0}(x)-\sum_{j\in\mathcal{N}_{i_0}^{out}}G_{i_0j}(x)\right)\\
&=\sum_{i=1}^{n}\tilde{f}_i(x)-f_{i_0}^0(x)-\left(\sum_{l\in\mathcal{N}_{i_0}^{in}}G_{li_0}(x)-\sum_{j\in\mathcal{N}_{i_0}^{out}}G_{i_0j}(x)\right)\\
-\tilde{f}_{i_0}(x)&=-f_{i_0}^0(x)-\left(\sum_{l\in\mathcal{N}_{i_0}^{in}}G_{li_0}(x)-\sum_{j\in\mathcal{N}_{i_0}^{out}}G_{i_0j}(x)\right).
\end{aligned}$$

This expression is exactly the obfuscation step applied at root node i_0 see (2.4). Consequently, the noise functions computed by our algorithm are consistent and feasible.

We present an example to illustrate the above construction in Figure 2.1. We present an example of a directed graph $\mathcal{G}_0$ with eight nodes and one adversary.

We have constructed a set of perturbation functions $G_{ij}(x)$ such that when we perturb private objective functions $f_i^0(x)$ using $G_{ij}(x)$ we will get $\tilde{f}_i(x)$, for each $i \in [n]$. This results in the execution for P_1 and P_2 being exactly the same. By selecting different $f_i^0(x)$, we can have several different distributed optimization problems result in identical execution. This ensures that observations by an adversary are insufficient to determine $f_i(x)$. This proves sufficiency of $\left|\left(\cup_{i\in V}\mathcal{N}_i^{in}(0)\right) \cup \left(\cup_{i\in V}\mathcal{N}_i^{out}(0)\right) \setminus V\right| \geq \tau + 1$ (for each $V \subset \mathcal{V}\backslash\mathcal{A}$) for privacy.

Necessity and Sufficiency (P1): The necessity and sufficiency proof for $P1$ are similar as above. The necessity can be easily proven by contradiction. We assume that a node i has fewer than $\tau+1$ neighbors. If the neighbors were adversarial then they would observe all the noise function shared by node i and shared to node i and break privacy. The sufficiency proof requires an additional observation. If τ nodes are corrupted by an adversary and removed from the graph it may fragment the graph into multiple components. So we need to use the constructive method similar to above for each component. □

2.3.4 Proof of Theorems 1 and 3

Recall, the obfuscation step (2.4) performed by agents transforms private objective functions to perturbed objective functions $\tilde{f}_i(x)$ for all $i \in [n]$. Distributed optimization algorithm is run by the nodes over the perturbed objective functions $\tilde{f}_i(x)$. Proving convergence of FS-GP essentially requires proving that GP algorithm can distributedly minimize $\sum_{i=1}^n \tilde{f}_i(x)$ where each agent i has access to $\tilde{f}_i(x)$.

First, we observe from Section 2.3 that the perturbed objective functions, $\tilde{f}_i(x)$, may be non-convex although their sum given by $\sum_{i=1}^n \tilde{f}_i(x)$ is convex. Recall that the private objective functions had bounded gradients (bound L) and Lipschitz continuous gradients (constant N). The noise functions $R_{ij}(x)$ exchanged between the nodes also have bounded and Lipschitz continuous gradients. This allows us to ensure that the perturbed objective functions have bounded gradients and Lipschitz continuous gradients albeit with different constants. That is $\|\nabla\tilde{f}_i(x)\| \leq \tilde{L}$ for each $x \in \mathbb{R}^d$ and $\|\nabla\tilde{f}_i(x) - \nabla\tilde{f}_i(x)\| \leq \tilde{N}\|x-y\|$ for each $x \neq y$.

The convergence analysis closely follows the convergence analysis of Gradient-Push in [9] and we borrow a lemma from [9]. Our convergence analysis involves three key steps.

- First, we borrow an elementary result on disagreement in perturbed push-sum protocol from [9]. It provides a bound on the disagreement between the state values (z_k^i) and the state averages ($\overline{x}_k$). It allows us to claim convergence of local iterates to a common value.
- Secondly, we construct the Iterate Lemma that relates the distance between the state average $\overline{x}_k = \sum_{i=1}^n x_k^i$ and the solution x^* that minimizes $f(x)$. As we will elaborate later, the key difference between our proofs and the proofs in [9] lies in the proof for Iterate Lemma. We exploit Lipschitz continuity of gradients to obtain an approximation of $\nabla\tilde{f}_i(z_k^i)$ in terms of $\nabla\tilde{f}_i(\overline{x}_k)$ and an error that decays with time at a fast enough rate. Hence sum of gradients can be approximated as sum of $\nabla\tilde{f}_i(\overline{x}_k)$ and error. Next, we exploit

the aggregate invariance under the obfuscation step (2.5), to express sum of the approximate gradient as the gradient of $f(x)$. This allows us to to construct an iterate lemma that is a non-negative almost supermartingale.

- Finally, we invoke a result on convergence of non-negative almost supermartingales by Robbins and Siegmund [75] and use the iterate lemma to prove convergence result (Theorem 1). We also characterize the finite-time convergence rate of function value using the iterate lemma (Theorem 3).

Disagreement Lemma

First we consider the matrix A_k that captures the weights used in update equation for w^i_{k+1} and y^i_{k+1} (Lines 5 and 6 in Algorithm 1). We can define the $[i,j]^{th}$ entry of matrix A_k as,

$$A_k[i,j] = \begin{cases} \frac{1}{deg^j_k} & j \in \mathcal{N}^{in}_i(k), \\ 0 & \text{otherwise.} \end{cases} \tag{2.8}$$

Observe that the sum of all entries in any column is 1. The term $(1/deg^j_k)$ appears for every i such that $j \in \mathcal{N}^{in}_i(k)$, i.e. deg^j_k times. All entries of A_k are non-negative and sum of all entries in any column is 1 implying A_k matrix is a column stochastic matrix. This is fairly common when dealing with consensus problem over directed graphs [9]. The product of transpose of the column stochastic matrices converges to a stochastic vector linearly (see Lemma 2 in [9]). It is used to prove the following key result borrowed from [9]. Lemma 1 presents bounds on the behavior of iterates of a perturbed push-sum algorithm. In the context of distributed optimization and our algorithm, the gradient based updates can be considered as the perturbations. We will use the claims in Lemma 1 to prove convergence of iterates at each agent to a common value. However, the lemma is for scalar variables. So we will reproduce the lemma first and discuss its applicability to our problem later. Consider perturbed push-sum protocol as presented in [9]. Consider, $a_k = [a^1_k, \dots a^n_k]^T$ where each a^i_k is a scalar. Similarly, $b_k = [b^1_k, \dots b^n_k]^T$, $c_k = [c^1_k, \dots c^n_k]^T$ and $d_k = [d^1_k, \dots d^n_k]^T$. Recall A_k is a column stochastic matrix as defined in (2.8). Finally consider $\epsilon_k = [\epsilon^1_k, \dots, \epsilon^n_k]$ where ϵ^i_k is a scalar perturbation added by agent i at iteration k. The perturbed push-sum protocol [9] can be written as:

$$\begin{aligned} a_{k+1} &= A_k b_k, \\ c_{k+1} &= A_k c_k, \\ d^i_{k+1} &= \frac{a^i_{k+1}}{c^i_{k+1}}, \quad \forall i \in [n], \\ b_{k+1} &= a_{k+1} + \epsilon_{k+1}. \end{aligned} \tag{2.9}$$

Lemma 1 (Lemma 1, [9])**.** *Consider the iterate c^i_k for each node i generated by* (2.9) *and the graph sequence*

$\mathcal{G}_k$ to be $B-$strongly connected, then:

(a) For all $k \geq 1$ we have,

$$|d_{k+1}^i - \overline{b}_k| \leq \frac{8}{\delta}\left(\lambda^k \|b_0\|_1 + \sum_{t=1}^{k} \lambda^{k-t}\|\epsilon_t\|_1\right),$$

where $\overline{b}_k = \frac{1}{n}\sum_{i=1}^n b_k^i$, $\delta \geq \frac{1}{n^{nB}}$ and $\lambda \leq \left(1 - \frac{1}{n^{nB}}\right)^{\frac{1}{B}}$.
(b) If $\lim_{k\to\infty} \epsilon_k^i = 0$ for all $i \in [n]$ then,

$$\lim_{k\to\infty} |d_{k+1}^i - \overline{b}_k| = 0, \text{ for all } i \in [n].$$

(c) If $\{\alpha_k\}$ is a non-increasing, positive scalar sequence, with $\sum_{t=1}^{\infty} \alpha_k |\epsilon_k^i| < \infty$ $\forall i$,

$$\sum_{t=0}^{\infty} \alpha_{k+1}|d_{k+1}^i - \overline{b}_k| < \infty, \text{ for all } i \in [n].$$

Now observe the similarities between perturbed push sum protocol and push-gradient algorithm in FS-GP (Algorithm 1). The perturbation ϵ represents gradient based update $\alpha_k \nabla \tilde{f}_i(x)$. We will elaborate this in the proof of Theorem 1 later in this subsection.

Iterate Lemma

We construct the iterate lemma that relates the distance of the average state to the optimum over time (iterations). As discussed before, the iterate lemma has structure similar to the result on the convergence of almost supermartingales [75]. We will reproduce the deterministic version of the almost supermartingale convergence result [8, 9, 20] first, followed by the iterate lemma.

Lemma 2 (Robbins-Siegmund, [75])**.** *Let $\{F_k\}$, $\{E_k\}, \{G_k\}$ and $\{H_k\}$, be non-negative, real sequences. Assume that $\sum_{k=0}^{\infty} F_k < \infty$, and $\sum_{k=0}^{\infty} H_k < \infty$ and*

$$E_{k+1} \leq (1 + F_k)E_k - G_k + H_k.$$

Then, the sequence $\{E_k\}$ converges to a non-negative real number and $\sum_{k=0}^{\infty} G_k < \infty$.

Lemma 3 (Iterate Lemma)**.** *Consider the distributed optimization problem presented in Section 2.2. We have for all $v \in \mathcal{X}$ and $k > 0$,*

$$\|\overline{x}_{k+1} - v\|^2 \leq \left(1 + \alpha_{k+1}\tilde{N}\left(\max_j \|\overline{x}_k - z_{k+1}^j\|\right)\right)\|\overline{x}_k - v\|^2 - 2\frac{\alpha_{k+1}}{n}\left(f(\overline{x}_k) - f(v)\right) + \alpha_{k+1}^2\tilde{L}^2 + \alpha_{k+1}\tilde{N}\left(\max_j \|\overline{x}_k - z_{k+1}^j\|\right).$$

Proof. Let us define $\tilde{x}_k^l$ to be the vector in $\mathbb{R}^n$ which stacks up the l^{th} entries of vectors x_k^i for each node i. That is, the j^{th} entry of $\tilde{x}_k^l$ is the l^{th} entry of vector x_k^j. Similarly $\tilde{g}_k^l$ is the vector in $\mathbb{R}^n$ which stacks up the l^{th} entries of vectors $\nabla \tilde{f}_j(z_k^j)$ for each node j. And the j^{th} entry of $\tilde{g}_k^l$ is the l^{th} entry of vector $\nabla \tilde{f}_j(z_k^j)$.

We follow the x_{k+1}^i update equation from gradient-push protocol (Line 8 in Algorithm 1). We can rewrite the equation using the above defined notation as follows

$$\tilde{x}_{k+1}^l = A_k \tilde{x}_k^l - \alpha_{k+1} \tilde{g}_{k+1}^l.$$

Note that A_k is a column stochastic matrix. Hence, $\mathbf{1}^T A_k = \mathbf{1}^T$. Using this in above equation, we get, for each $l = 1, 2, \dots d$,

$$\begin{aligned} \mathbf{1}^T \tilde{x}_{k+1}^l &= \mathbf{1}^T A_k \tilde{x}_k^l - \alpha_{k+1} \mathbf{1}^T \tilde{g}_{k+1}^l \\ \implies \frac{1}{n} \sum_{j=1}^n \tilde{x}_{k+1}^l[j] &= \frac{1}{n} \sum_{j=1}^n \tilde{x}_k^l[j] - \frac{\alpha_{k+1}}{n} \sum_{j=1}^n \tilde{g}_{k+1}^l[j]. \end{aligned}$$

Observe that, the l^{th} entry of $\overline{x}_k$ is $\frac{1}{n}\sum_{j=1}^n \tilde{x}_k^l[j]$, and the above expression is rewritten as,

$$\overline{x}_{k+1} = \overline{x}_k - \frac{\alpha_{k+1}}{n} \sum_{j=1}^n g_{k+1}^j = \overline{x}_k - \frac{\alpha_{k+1}}{n} \sum_{j=1}^n \nabla \tilde{f}_j(z_{k+1}^j). \tag{2.10}$$

Consider an arbitrary vector v. Next we use (2.10) to get the following relation,

$$\begin{aligned} \|\overline{x}_{k+1} - v\|^2 &= \|\overline{x}_k - \frac{\alpha_{k+1}}{n} \sum_{j=1}^n \nabla \tilde{f}_j(z_{k+1}^j) - v\|^2 \\ &\leq \|\overline{x}_k - v\|^2 + \frac{\alpha_{k+1}^2}{n^2} \|\sum_{j=1}^n \nabla \tilde{f}_j(z_{k+1}^j)\|^2 - 2\frac{\alpha_{k+1}}{n} (\overline{x}_k - v)^T \left(\sum_{j=1}^n \nabla \tilde{f}_j(z_{k+1}^j) \right) \\ &\leq \|\overline{x}_k - v\|^2 + \alpha_{k+1}^2 \tilde{L}^2 - 2\frac{\alpha_{k+1}}{n} (\overline{x}_k - v)^T \left(\sum_{j=1}^n \nabla \tilde{f}_j(z_{k+1}^j) \right), \end{aligned} \tag{2.11}$$

where we use $\|\nabla \tilde{f}_j(x)\| \leq \tilde{L}$ and $\|\sum_{j=1}^n \nabla \tilde{f}_j(z_{k+1}^j)\| \leq n\tilde{L}$.

Note that at this step, one would invoke gradient inequality for convex functions (see [9] or [8]). However, $\tilde{f}_j(x)$ is non-convex so we cannot use gradient inequality for convex functions directly. Recall the Lipschitz continuity of gradients. We approximate the gradient $\nabla \tilde{f}_j(z_{k+1}^j)$ as the gradient at average state $\nabla \tilde{f}_j(\overline{x}_k)$ plus an error.

$$\nabla \tilde{f}_j(z_{k+1}^j) = \nabla \tilde{f}_j(\overline{x}_k) + e_{k+1}^j, \tag{2.12}$$

where $\|e_{k+1}^j\| \leq \tilde{N} \|\overline{x}_k - z_{k+1}^j\|$ and $\tilde{N}$ is the Lipschitz constant of gradients $\nabla \tilde{f}_j(x)$. This follows from the

Lipschitz continuity of gradient. Next, we construct a bound on the last term from (2.11),

$$B = (\overline{x}_k - v)^T \left(\sum_{j=1}^{n} \nabla \tilde{f}_j(z_{k+1}^j) \right) = \sum_{j=1}^{n} (\overline{x}_k - v)^T \left(\nabla \tilde{f}_j(\overline{x}_k) + e_{k+1}^j \right)$$
$$= \underbrace{(\overline{x}_k - v)^T \left(\sum_{j=1}^{n} \nabla \tilde{f}_j(\overline{x}_k) \right)}_{B_1} + \underbrace{\sum_{j=1}^{n} (\overline{x}_k - v)^T \left(e_{k+1}^j \right)}_{B_2}. \tag{2.13}$$

Next, we individually construct bounds on B_1 and B_2. The first term B_1 can be rewritten using the aggregate invariance property 2.5 followed by using gradient inequality for convex functions to construct the bound

$$B_1 = (\overline{x}_k - v)^T \left(\sum_{j=1}^{n} \nabla \tilde{f}_j(\overline{x}_k) \right) = (\overline{x}_k - v)^T \left(\nabla f(\overline{x}_k) \right) \geq f(\overline{x}_k) - f(v). \tag{2.14}$$

The bound on B_2 is slightly tricky and requires some manipulation. We exploit Lipschitz continuity of gradients as discussed above in (2.12) and the fact that $2\|a\| \leq \|a\|^2 + 1$ for any a to get,

$$\begin{aligned}
-B_2 &= \sum_{j=1}^{n} (v - \overline{x}_k)^T \left(e_{k+1}^j \right) \\
&\leq \sum_{j=1}^{n} \|v - \overline{x}_k\| \|e_{k+1}^j\| \\
&\leq \sum_{j=1}^{n} \tilde{N} \|\overline{x}_k - v\| \|\overline{x}_k - z_{k+1}^j\| \\
&\leq n\tilde{N} \left(\max_j \|\overline{x}_k - z_{k+1}^j\| \right) \|\overline{x}_k - v\| \\
&\leq \frac{n}{2} \tilde{N} \left(\max_j \|\overline{x}_k - z_{k+1}^j\| \right) \left(1 + \|\overline{x}_k - v\|^2 \right).
\end{aligned} \tag{2.15}$$

Using the bounds presented in (2.14) and (2.15) in the iterate relation in (2.11), we get,

$$\begin{aligned}
\|\overline{x}_{k+1} - v\|^2 &\leq \|\overline{x}_k - v\|^2 + \alpha_{k+1}^2 \tilde{L}^2 - 2\frac{\alpha_{k+1}}{n} \left(f(\overline{x}_k) - f(v) \right) + \alpha_{k+1} \tilde{N} \left(\max_j \|\overline{x}_k - z_{k+1}^j\| \right) \left(1 + \|\overline{x}_k - v\|^2 \right) \\
&\leq \left[1 + \alpha_{k+1} \tilde{N} \left(\max_j \|\overline{x}_k - z_{k+1}^j\| \right) \right] \|\overline{x}_k - v\|^2 - 2\frac{\alpha_{k+1}}{n} \left(f(\overline{x}_k) - f(v) \right) \\
&\quad + \alpha_{k+1}^2 \tilde{L}^2 + \alpha_{k+1} \tilde{N} \left(\max_j \|\overline{x}_k - z_{k+1}^j\| \right).
\end{aligned}$$

□

Proof of Theorem 1

Proof. The proof of Theorem 1 has two parts. First, we prove that the iterates from every node converge to a common value, i.e. $\lim_{k\to\infty} \|z_{k+1}^j - \overline{x}_k\| = 0$. Second, we prove that the common value, $\overline{x}_k$, converges to an optimum $x^* \in \mathcal{X}$.

In what follows, we invoke Lemma 1 to show iterates form every node converge to the average value. Recall, Lemma 1 was for scalars and we intend to invoke it for each coordinate separately. Moreover, a_k represents a coordinate of w_k, b_k represents a coordinate of x_k, c_k represents y_k and d_k represents a coordinate of z_k.

Observe that the gradient updates decrease in magnitude as the number of iterations increases, i.e. $\lim_{k\to\infty} \alpha_k \nabla \tilde{f}_i(z_k^i) = 0$ for each i and for each coordinate, since $\lim_{k\to\infty} \alpha_k = 0$. This satisfies the sufficient condition in Lemma 1 (b). It guarantees that each coordinate of the iterate z_k^i eventually tracks the average $\overline{x}_k$ for that coordinate, i,e, $\lim_{k\to\infty} |z_{k+1}^i[p] - \overline{x}_k[p]| = 0$ for each $p = 1, \ldots, d$, where $z_{k+1}^i[p], \overline{x}_k[p]$ represents the p^{th} coordinate of local iterate z_k^i and aggregate $\overline{x}_k$. This also guarantees $\lim_{k\to\infty} \sum_{p=1}^{d} |z_{k+1}^i[p] - \overline{x}_k[p]| = \lim_{k\to\infty} \|z_{k+1}^i - \overline{x}_k\|_1 = 0$. For finite dimensional spaces, we have norm equivalence relations, i.e. if q lies in finite dimensional space $\mathbb{R}^d$ then $\|q\|_2 \leq \|q\|_1$. This directly gives us $\lim_{k\to\infty} \|z_{k+1}^i - \overline{x}_k\|_2 = 0$. This proves the first part of Theorem 1.

Before we move on to the proof of the second part we make another observation,

$$\sum_{k=1}^{\infty} \alpha_k \|\alpha_k \nabla \tilde{f}_i(z_k^i)\|_2 = \sum_{k=1}^{\infty} \alpha_k^2 \|\nabla \tilde{f}_i(z_k^i)\|_2 \leq \tilde{L} \sum_{k=1}^{\infty} \alpha_k^2 < \infty.$$

From the norm equivalence in finite dimensional spaces, we have for $q \in \mathbb{R}^d$, $\|q\|_1 \leq \sqrt{d}\ \|q\|_2$. Using this norm equivalence and the above expression we get, $\sum_{k=1}^{\infty} \alpha_k \|\alpha_k \nabla \tilde{f}_i(z_k^i)\|_1 < \infty$. From the definition of $\|\|_1$, this directly gives us $\sum_{k=1}^{\infty} \alpha_k |\alpha_k \nabla \tilde{f}_i(z_k^i)[p]| < \infty$ for each $p = 1, \ldots, d$. This is the sufficient condition in Lemma 1 (c). It implies that $\sum_{k=0}^{\infty} \alpha_k |z_{k+1}^i[p] - \overline{x}_k[p]| < \infty$ for each i and for each p. By adding this expression for each p we directly get $\sum_{k=0}^{\infty} \alpha_k \|z_{k+1}^i - \overline{x}_k\|_1 < \infty$. From norm equivalence we can state, $\sum_{k=0}^{\infty} \alpha_k \|z_{k+1}^i - \overline{x}_k\|_2 < \infty$ for each i. Consequently, we can write $\sum_{k=0}^{\infty} \alpha_k \max_i \|z_{k+1}^i - \overline{x}_k\|_2 < \infty$. Note that this term appears twice in the expression for Iterate Lemma.

The proof for second part is quite straightforward given Lemma 3 and the key observations made above. Consider Lemma 3 with $v = x^* \in \mathcal{X}^*$ and we get

$$\|\overline{x}_{k+1} - x^*\|^2 \leq \left[1 + \alpha_{k+1}\tilde{N}\left(\max_j \|\overline{x}_k - z_{k+1}^j\|\right)\right] \|\overline{x}_k - x^*\|^2 - 2\frac{\alpha_{k+1}}{n}\left(f(\overline{x}_k) - f(x^*)\right) + \alpha_{k+1}^2 \tilde{L}^2 + \alpha_{k+1}\tilde{N}\left(\max_j \|\overline{x}_k - z_{k+1}^j\|\right).$$

This has the exact same structure as the dissipation inequality in Lemma 2. Observe $\{f(\overline{x}_k) - f(x^*)\}$ is a scalar, non-negative and real sequence. First we use $\sum_{k=0}^{\infty} \alpha_k \max_i \|z_{k+1}^i - \overline{x}_k\| < \infty$ and $\alpha_{k+1} \leq \alpha_k$ to show that $\sum_{j=0}^{\infty} \alpha_{k+1} \max_j \|\overline{x}_k - z_{k+1}^j\| < \infty$. Moreover, we use these statements to show

$\sum_{j=0}^{\infty}\left(\alpha_{k+1}^2\tilde{L}^2+\tilde{N}\alpha_{k+1}\left(\max_j\|\overline{x}_k-z_{k+1}^j\|\right)\right)<\infty$ following $\sum_k\alpha_k^2<\infty$ and $\sum_{k=0}^{\infty}\alpha_k\max_i\|z_{k+1}^i-\overline{x}_k\|<\infty$. This satisfies the conditions in Lemma 2. We conclude, $\sum_{k=0}^{\infty}\alpha_k(f(\overline{x}_k)-f(x^*))<\infty$ and $\lim_{k\to\infty}\|\overline{x}_k-x^*\|$ exists and is finite. Since $\sum_{k=0}^{\infty}\alpha_k=\infty$ and the function $f(x)$ is convex and continuous; we conclude that $\liminf_{k\to\infty}f(\overline{x}_k)=f(x^*)$ and sequence $\{\overline{x}_k\}$ converges to the optimum $x^*\in\mathcal{X}^*$.

Combining both the statements we get that, each iterate z_k^i tracks $\overline{x}_k$ and the iterate average $\overline{x}_k$ asymptotically converges to the optimizer x^*, implying that the iterates z_k^i converge to the optimum asymptotically. □

Proof of Theorem 3

Proof. The proof of Theorem 3 follows from Lemma 3 and a few elementary results on sequences and series.

We begin our analysis by considering the time weighted average of state $\widehat{x}_T^j=\frac{\sum_{k=0}^T\alpha_{k+1}x_k^j}{\sum_{k=0}^T\alpha_{k+1}}$, along with the fact that $\overline{\widehat{x}}_T=\frac{1}{n}\sum_{j=1}^n\widehat{x}_T^j=\frac{\sum_{k=0}^T\alpha_{k+1}\overline{x}_k}{\sum_{k=0}^T\alpha_{k+1}}$ and the convexity of $f(x)$, we get,

$$\begin{aligned}f(\overline{\widehat{x}}_T)-f^*&=f\left(\frac{\sum_{k=0}^T\alpha_{k+1}\overline{x}_k}{\sum_{k=0}^T\alpha_{k+1}}\right)-f^*\\&\leq\frac{\sum_{k=0}^T\alpha_{k+1}f(\overline{x}_k)}{\sum_{k=0}^T\alpha_{k+1}}-f^*\\&=\frac{\sum_{k=0}^T\alpha_{k+1}\left(f(\overline{x}_k)-f^*\right)}{\sum_{k=0}^T\alpha_{k+1}}.\end{aligned}\tag{2.16}$$

Next, we recall that the dissipation relation in Lemma 3 is $\eta_{k+1}^2\leq(1+F_k)\eta_k^2-c\alpha_{k+1}(f(\overline{x}_k)-f(x^*))+H_k$, where $c=2/n$ and F_k,H_k are defined in Lemma 3. We can rearrange terms to get $\alpha_{k+1}(f(\overline{x}_k)-f(x^*))\leq(1/c)\left((1+F_k)\eta_k^2-\eta_{k+1}^2+H_k\right)$. We use it to bound (2.16) as follows

$$f(\overline{\widehat{x}}_T)-f^*\leq\frac{\sum_{k=0}^T(1/c)\left((1+F_k)\eta_k^2-\eta_{k+1}^2+H_k\right)}{\sum_{k=0}^T\alpha_{k+1}}.\tag{2.17}$$

Canceling the telescoping terms in (2.17) and ignoring the negative term in the upper bound, we get

$$f(\overline{\widehat{x}}_T)-f^*\leq(1/c)\left(\frac{\eta_0^2+\sum_{k=0}^T\left(F_k\eta_k^2+H_k\right)}{\sum_{k=0}^T\alpha_{k+1}}\right).\tag{2.18}$$

Note that η_0^2 depends only on the initialization the algorithm (x_0^j) and the optimizer (x^*). Given an initialization, we can state that there exists $D_0<\infty$ such that $\eta_0^2<D_0$. Theorem 1 we know that $\lim_{k\to\infty}\eta_k^2=0$ implying that there exists a constant D_1 such that $\eta_k^2<\epsilon$ for all $k\geq D_1$. Consequently, there exists $D_2\triangleq\max\{\eta_0^2,\eta_1^2,\ldots,\eta_{D_1}^2,\epsilon\}$ that satisfies $\eta_k^2\leq D_2<\infty$ (Theorem 2.3.2, [76]). If $\alpha_{k+1}=1/\sqrt{k+1}$, we have from comparison test, $\sum_{k=0}^T\alpha_{k+1}\geq\sqrt{T+1}$. This along with (2.18), we get,

$$f(\overline{\widehat{x}}_T)-f^*\leq(1/c)\frac{D_0+\sum_{k=0}^T\left(D_2F_k+H_k\right)}{\sqrt{T+1}}.\tag{2.19}$$

We first construct a bound on $\sum_{k=0}^{T} F_k$. That is, $\sum_{k=0}^{T} \tilde{N}\alpha_{k+1} \max_j \|\overline{x}_k - z_{k+1}^j\|$. Recall from equivalence of norms in finite dimensional spaces, $\|q\|_2 \leq \|q\|_1$ and $\|q\|_1 \leq \sqrt{d}\|q\|_2$ for $q \in \mathbb{R}^d$. We use the Lemma 1(a) to simplify the bound. Let $D = \max_j \max_p \|x_0^j[p]\|_1$ and we have

$$\begin{aligned}
\sum_{k=0}^{T} F_k &\leq \sum_{k=0}^{T} \tilde{N}\alpha_{k+1} \max_j \|\overline{x}_k - z_{k+1}^j\|_1 \\
&\overset{(a)}{\leq} \sum_{k=0}^{T} \tilde{N}\alpha_{k+1} \max_j \sum_{p=1}^{d} |\overline{x}_k[p] - z_{k+1}^j[p]| \\
&\overset{(b)}{\leq} \sum_{k=1}^{T} \tilde{N}\alpha_{k+1} \max_j \left[\sum_{p=1}^{d} \frac{8}{\delta}\left[D\lambda^k + \sum_{t=1}^{k} \lambda^{k-t}\alpha_t \|\nabla \tilde{f}_j(z_t^j)\|_1\right]\right] + \tilde{N}\alpha_1 \max_j \|\overline{x}_0 - z_1^j\|_1 \\
&\overset{(c)}{\leq} \sum_{k=1}^{T} \tilde{N}\alpha_{k+1} \max_j \left[\sum_{p=1}^{d} \frac{8}{\delta}\left[D\lambda^k + \sum_{t=1}^{k} \lambda^{k-t}\alpha_t \|\nabla \tilde{f}_j(z_t^j)\|_1\right]\right] + D_3 \\
&\overset{(d)}{\leq} \frac{8\tilde{N}d}{\delta}\left(D\sum_{k=1}^{T}\alpha_{k+1}\lambda^k + \sqrt{d}\,\tilde{L}\sum_{k=1}^{T}\alpha_{k+1}\sum_{t=1}^{k}\lambda^{k-t}\alpha_t\right) + D_3,
\end{aligned}$$

where (a) follows from the definition of $\|\cdot\|_1$, we get (b) by using Lemma 1 (a) and we get (c) by defining $D_3 = \tilde{N}\alpha_1 \max_j \|\overline{x}_0 - z_1^j\|_1$. Finally, we get (d) by using the observation that the perturbation is gradient based update and the fact that $\|\nabla \tilde{f}_j(z_t^j)\|_1 \leq \sqrt{d}\tilde{L}$. Recall, the proof of Theorem 1 in the prior section, we showed that if $\lambda < 1$ then the series $\sum_k \alpha_k\lambda^k$ converges for non-increasing α_k. Let us set $C_0 = Dd\sum_{k=1}^{T}\alpha_{k+1}\lambda^k$ and recall that $\alpha_{k+1} \leq \alpha_t$ for all $t \leq k+1$, to get

$$\sum_{k=0}^{T} F_k \leq \frac{8\tilde{N}}{\delta}\left(C_0 + d^{3/2}\tilde{L}\sum_{k=1}^{T}\sum_{t=1}^{k}\lambda^{k-t}\alpha_t^2\right) + D_3. \tag{2.20}$$

Note $\alpha_t^2 = 1/t$ following step-size definition.

We can rewrite $d^{3/2}\tilde{L}\sum_{k=1}^{T}\sum_{t=1}^{k}(\lambda^{k-t}/t) = d^{3/2}\tilde{L}\sum_{k=1}^{T}\left((\sum_{j=0}^{T-k}\lambda^j)/k\right) \leq d^{3/2}\tilde{L}\sum_{k=1}^{T}(1/(1-\lambda))/k \leq C_1\log(T+1) + C_1$ where $C_1 = d^{3/2}\tilde{L}/(1-\lambda)$. This gives us,

$$\sum_{k=0}^{T} F_k \leq \frac{8\tilde{N}}{\delta}\left(C_0 + C_1 + C_1\log(T+1)\right) + D_3.$$

Next, we move on to compute $\sum_{k=0}^{T} H_k$. Recall from Lemma 3 that $H_k = F_k + \alpha_{k+1}^2\tilde{L}^2$.

$$\begin{aligned}
\sum_{k=0}^{T} H_k &= \sum_{k=0}^{T}\left(\alpha_{k+1}^2\tilde{L}^2 + F_k\right) \\
&= \sum_{k=0}^{T} F_k + \sum_{k=0}^{T}\alpha_{k+1}^2\tilde{L}^2 = \sum_{k=0}^{T} F_k + \sum_{k=0}^{T}\frac{\tilde{L}^2}{k+1} \\
&\leq \left(\frac{8\tilde{N}}{\delta}\left(C_0 + C_1 + C_1\log(T+1)\right) + D_3\right) + \tilde{L}^2\log(T+1).
\end{aligned} \tag{2.21}$$

Together using (2.20) and (2.21) in (2.19) we get,

$$f(\widehat{\overline{x}}_T) - f^* \leq (1/c)\frac{D_0 + (1+D_2)\left(\frac{8\tilde{N}}{\delta}\left(C_0 + d^{3/2}\tilde{L}\sum_{k=1}^{T}\sum_{t=1}^{k}\lambda^{k-t}\alpha_t^2\right) + D_3\right) + \tilde{L}^2\log(T+1)}{\sqrt{T+1}}$$
$$\leq \mathcal{O}\left(\tilde{L}^2\frac{\log(T+1)}{\sqrt{T+1}}\right). \tag{2.22}$$

Let us define $\widehat{z}_{T+1}^j = \frac{\sum_{k=0}^{T}\alpha_{k+1}z_{k+1}^j}{\sum_{k=0}^{T}\alpha_{k+1}}$. Finally, we use (2.22) to get,

$$\begin{aligned}
f(\widehat{z}_{T+1}^j) - f^* &= f(\widehat{z}_T^j) - f(\widehat{\overline{x}}_T) + f(\widehat{\overline{x}}_T) - f^* \\
&\overset{(a)}{\leq} nL\Big\|\frac{\sum_{k=0}^{T}\alpha_{k+1}z_{k+1}^j}{\sum_{k=0}^{T}\alpha_{k+1}} - \frac{\sum_{k=0}^{T}\alpha_{k+1}\overline{x}_k}{\sum_{k=0}^{T}\alpha_{k+1}}\Big\| + \mathcal{O}\left(\tilde{L}^2\frac{\log(T+1)}{\sqrt{T+1}}\right) \\
&\overset{(b)}{\leq} nL\frac{\sum_{k=0}^{T}\alpha_{k+1}\|z_{k+1}^j - \overline{x}_k\|}{\sqrt{T+1}} + \mathcal{O}\left(\tilde{L}^2\frac{\log(T+1)}{\sqrt{T+1}}\right) \\
&\overset{(c)}{\leq} \mathcal{O}\left(\tilde{L}^2\frac{\log(T+1)}{\sqrt{T+1}}\right).
\end{aligned}$$

where (a) follows from (2.22) and the boundedness of objective function gradients, (b) follows from $\sum_{k=0}^{T}\alpha_{k+1} \geq \sqrt{T+1}$ and (c) follows from $\sum_{k=0}^{T}\alpha_{k+1}\|z_{k+1}^j - \overline{x}_k\| = \frac{1}{\tilde{N}}\sum_{k=0}^{T}F_k$ and (2.20). □

2.4 Function Sharing Algorithm for Undirected Graphs

We presented Function Sharing based strategy for privacy preserving distributed optimization over undirected graphs in [20]. We first present the algorithm followed by a review of some key results for FS-DGD (Algorithm 2) and proofs. Note that we consider a slight modification to the distributed optimization problem in this section. Agents are interested in collaboratively minimizing the aggregate function $f(x)$ over a convex, non-empty and compact set $\mathcal{X}$. That is agents solve,

$$\underset{x\in\mathcal{X}}{\text{minimize}} f(x) \triangleq \sum_{i=1}^{n} f_i(x), \tag{2.23}$$

where agent i accesses local objective function $f_i(x)$.

2.4.1 Function Sharing - Distributed Gradient Descent

We propose Function Sharing - Distributed Gradient Descent (FS-DGD, Algorithm 2) protocol for privacy preserving distributed optimization over undirected networks. FS-DGD essentially involves two steps. In the first step we perform obfuscation to transform private objective function to perturbed objective function. In the second step we run Distributed Gradient Descent protocol as described in [8].

Algorithm 2 Function Sharing - Distributed Gradient Descent, FS-DGD [20]

Input: Each node j has access to $f_j(x)$.
Result: Iterates converge to $x^* \in \arg\min_{x \in \mathcal{X}} \sum_{j=1}^{n} f_j(x)$

$\Diamond$ **Obfuscation Step**

1: Each node j sends function $R_{ji}(x)$ to neighbors $i \in \mathcal{N}_j$

2: Each node j perturbs its objective function to get new objective function $\tilde{f}_j(x)$

$$\tilde{f}_j(x) = f_j(x) + \sum_{i \in \mathcal{N}_j} R_{ij}(x) - \sum_{i \in \mathcal{N}_j} R_{ji}(x)$$

$\Diamond$ **Distributed Gradient Descent Algorithm** is run at each node.

3: Initialize: $x_1^j \in \mathcal{X}$ for each $j \in \mathcal{V}$.

4: **for** Iteration Number $k = 1, 2, \ldots$ **do**

5: Information Fusion: $v_k^j = \sum_{i \in \mathcal{N}_i} B_k[j,i] x_k^i$

6: Projected Gradient Descent:

$$x_{k+1}^j = \mathcal{P}_{\mathcal{X}} \left[v_k^j - \alpha_k \nabla \tilde{f}_j(v_k^j) \right]$$

7: **end for**

The obfuscation step (lines 1-2, Algorithm 2) is exactly the same as in directed graphs, if we view each undirected edge as consisting of two directed edges. Next, agents run Distributed Gradient Descent where each agent j uses $\tilde{f}_j(x)$ as the objective function. DGD uses a combination of consensus dynamics and local gradient descent to distributedly find a minimizer of $\sum_{j=1}^{n} \tilde{f}_j(x)$. As shown on line 5 (Algorithm 2), each agent performs a *consensus step* (also called *information fusion*), which involves computing a convex combination of the state estimates. The resulting convex combination is denoted by v_k^j. Matrix B_k used in this step is a doubly stochastic matrix [8], which can be constructed by the agents using previously proposed techniques, such as Metropolis weights [77].

Agent j performs *projected gradient descent* step (line 6, Algorithm 2) involving descent from v_k^j along the local objective function's gradient $\nabla \tilde{f}_j(v_k^j)$, followed by projection onto the feasible set $\mathcal{X}$. This step yields the new state estimate at agent j, namely, x_{k+1}^j. Note that this step differs from FS-GP algorithm presented in Section 2.3.1. While we perform optimization over $\mathbb{R}^d$ in the optimization over directed graphs case, we perform optimization over compact set $\mathcal{X}$ in the optimization over undirected graphs case. The step sizes α_k, are non-summable yet square summable, i.e. $\sum_k \alpha_k = \infty$ and $\sum_k \alpha_k^2 < \infty$.

Prior work [8] analyzes the convergence of DGD Algorithm for convex optimization problem, i.e. $f_i(x)$ is convex for each i. The result states that the agents' state estimate asymptotically reaches consensus on an optimum in $\mathcal{X}^*$. We prove convergence of FS-DGD in Section 2.4.4. We show DGD algorithm solves convex sum of non-convex functions [67] and this leads us to show that FS-DGD iterates converge to an optimizer in $\mathcal{X}^*$ [20, 69].

2.4.2 Main Results

We discuss correctness, privacy and finite-time rate results for FS-DGD algorithm. We consider the problem defined in Section 2.2 along with the gradient boundedness and Lipschitz continuous gradient assumptions. We assume that the graph $\mathcal{G}_k$ is strongly connected at each k.[6] We also assume that the decision set $\mathcal{X}$ is convex, compact and non-empty.

Theorem 4 (Correctness of FS-DGD, [20])**.** *Consider a distributed optimization problem and assumptions from Section 2.2. Assume that the graph $\mathcal{G}_k$ is strongly connected at each k. Iterates x_k^j for FS-DGD (Algorithm 2) converges to an optimum $x^* \in \mathcal{X}^*$ asymptotically.*

Theorem 4 guarantees that the sequence of iterates generated by FS-DGD converge to the optimum. Theorem 4 guarantees that privacy does not impact the correctness of the algorithm (accuracy).

We can show that FS-DGD preserves privacy of local objective functions as per Definition 1. If the local objective functions belong to a set closed under addition, then we can design set of distributed optimization problems similar to $\mathcal{F}$ in Section 2.3.2 such that the adversarial observations are compatible with all problems in set $\mathcal{F}$.

Theorem 5. *(Privacy via Non-identifiability of FS-DGD, [20]) Consider the communication graph at time zero, $\mathcal{G}_0$.*

(P1) Let i be a non-adversarial node, then minimum degree of $\mathcal{G}_0 \geq \tau + 1$ is necessary and sufficient for privacy of $f_i(x)$ as per Definition 1. In other words, the adversary cannot learn $f_i(x)$.

(P2) Assume that $\mathcal{I}$ is a strict subset of non-adversarial nodes, then vertex connectivity of $\mathcal{G}_0$, $\kappa(\mathcal{G}_0) \geq \tau+1$, is necessary and sufficient for privacy of nodes in $\mathcal{I}$ as per Definition 1. In other words, the adversary cannot learn $\sum_{i\in\mathcal{I}} f_i(x)$.

The privacy result in Theorem 5 for undirected graphs has a similar structure as Theorem 2. We require the graph to have a vertex connectivity of at least $\tau + 1$ for privacy. This condition is tight. Intuitively, privacy demands that each agent or group of agents have at least $\tau + 1$ neighbors, so that even if τ of them are adversarial there is significant ambiguity due to perturbation (noise) function shared between non-adversarial neighbor. However, as discussed in Section 2.3.2, the correctness property of our algorithm precludes any privacy protection afforded to aggregate objective function $f(x)$. We also forgo the privacy protections that could be provided to the group of all non-adversarial agents i.e. $\mathcal{V} \setminus \mathcal{A}$.

2.4.3 Proof of Theorem 5

Proof. **Necessity (P2):**

[6] This can be further relaxed to B-connectedness but we have not explicitly considered it here.

Let us assume that $\kappa(\mathcal{G}_0) \leq \tau$. Hence, if τ nodes are deleted (along with their edges) the graph becomes disconnected to form components I_1 and I_2. Let the adversarial nodes be denoted by the set $\{1, \ldots, \tau\}$. Now consider that τ nodes, that we just deleted, are corrupted by an adversary. Agents generate correlated noise function using obfuscation step. All the perturbation functions $S_{ji}(x)$ shared by nodes in I_1 (with agents outside I_1) are observed by the adversary. Under our adversary model, an adversary may estimate the true objective function easily by using

$$\sum_{l \in I_1} f_l(x) = \sum_{l \in I_1} \widehat{f}_l - \left(\sum_{j=1, i \in I_1}^{j=\tau} S_{ji}(x) - \sum_{j=1, i \in I_1}^{j=\tau} S_{ij}(x) \right).$$

This gives us contradiction. Hence, $\kappa(\mathcal{G}_0) > \tau$ is necessary.

Sufficiency (P2): We present a constructive method to show that given an execution and corresponding observations, any distributed optimization problem in $\mathcal{F}$ is compatible with the execution.

We conservatively assume that the adversary can observe the obfuscated functions, $\widehat{f}_i(x)$, the private objective functions of corrupted nodes $f_a(x)$ ($a \in \mathcal{A}$) and noise functions transmitted from and received by each of the corrupted agents are denoted by S_{aJ} and S_{Ka} ($J \in \mathcal{N}_a$ and K such that $a \in \mathcal{N}_K$, for all $a \in \mathcal{A}$). Since the corrupted nodes also follow the same protocol (Algorithm 2), the adversary is also aware of the fact that the private objective functions have been obfuscated by function sharing approach, i.e.

$$\widehat{f}_i(x) = f_i(x) + \sum_{k: i \in \mathcal{N}_k} S_{ki}(x) - \sum_{j \in \mathcal{N}_i} S_{ij}(x)$$

One can rewrite this transformation approach, using signed incidence matrix of bidirectional graph $\mathcal{G}$ [78, 79]

$$\widehat{\mathbf{f}} = \mathbf{f} + \mathbf{BS}, \tag{2.24}$$

where $\widehat{\mathbf{f}} = \left[\widehat{f}_1(x), \widehat{f}_1(x), \ldots, \widehat{f}_n(x)\right]^T$ is a $n \times 1$ vector of obfuscated functions $\widehat{f}_i(x)$ for $i = \{1, 2, \ldots, n\}$, and $\mathbf{f} = \left[f_1(x), f_1(x), \ldots, f_n(x)\right]^T$ is a $n \times 1$ vector of private (true) objective functions, $f_i(x)$. Moreover, $\mathbf{B}$ is the incidence matrix and $\mathbf{S}$ is the vector of noise functions $S_{ij}(x)$. Notice, $\mathbf{B} = \left[B_C, -B_C\right]$, where B_C (of dimension $n \times |\mathcal{E}|/2$) is the incidence matrix of a directed graph obtained by considering only one of the directions of every bidirectional edge in graph $\mathcal{G}$.[7] Each column of $\mathbf{B}$ represents a directed communication link between any two agents. Hence, any bidirectional edge between agents i and j is represented as two directed links, i to j, $(i, j) \in \mathcal{E}$ and j to i $(j, i) \in \mathcal{E}$ and corresponds to two columns in $\mathbf{B}$. $\mathbf{S}$ represents a $|\mathcal{E}| \times 1$ vector consisting of functions $S_{ij}(x)$. Each entry in vector $\mathbf{S}$, function $S_{ij}(x)$ corresponds to a column

[7] This represents an orientation of graph $\mathcal{G}$ [78].

of $\mathbf{B}$ which, in turn corresponds to link $(i,j) \in \mathcal{E}$; and similarly, function $S_{ji}(x)$ corresponds to a different column of $\mathbf{B}$ which, in turn corresponds to link $(j,i) \in \mathcal{E}$. Note that ℓ^{th} row of column vector $\mathbf{S}$ corresponds to ℓ^{th} column of incidence matrix $\mathbf{B}$.

We will show that, two different sets of true objective functions ($\mathbf{f}$ and $\mathbf{f}^o$) belonging to $\mathcal{F}$ and correspondingly two different set of arbitrary functions ($\mathbf{S}$ and $\mathbf{G}$), can lead to exactly same execution and observations for the adversary.[8] We want to show that both these cases can result in same obfuscated objective functions

$$\widehat{\mathbf{f}} = \mathbf{f} + \mathbf{BS} = \mathbf{f}^o + \mathbf{BG}. \tag{2.25}$$

We will show that given any set of private objective functions $\mathbf{f}^o$, suitably selecting arbitrary functions $G_{ij}(x)$ corresponding to links incident at non-adversarial agents, it is possible to make $\mathbf{f}^o$ indistinguishable from original private objective functions $\mathbf{f}$, solely based on the execution observed by the corrupted nodes. We do so by determining entries of $\mathbf{G}$, which are arbitrary functions that are dissimilar from $S_{ij}(x)$ when i and j are both non-adversarial. The design $\mathbf{G}$ such that the obfuscated objective functions $\widehat{\mathbf{f}}$ are the same for both situations.

Since corrupted nodes observe arbitrary functions corresponding to edges incident to and from them, we set the arbitrary functions corresponding to edges incident on corrupted nodes as $G_{ka}(x) = S_{ka}(x)$ and arbitrary functions corresponding to edges incident away the corrupted nodes as $G_{aj}(x) = S_{aj}(x)$ (where $k : a \in \mathcal{N}_k$ and $j \in \mathcal{N}_a$, for all $a \in \mathcal{A}$). Now, we define $\tilde{\mathbf{G}}$ as the vector containing all elements of $\mathbf{G}$ except those corresponding to the edges incident to and from the corrupted nodes.[9] Similarly, we define $\tilde{\mathbf{B}}$ to be the new incidence matrix obtained after deleting all edges that are incident on the corrupted nodes (i.e. deleting columns corresponding to the links incident on corrupted nodes, from the old incidence matrix $\mathbf{B}$). We subtract $G_{aj}(x)$ and $G_{ka}(x)$ ($\forall\ a \in \mathcal{A}$) by subtracting them from $[\widehat{\mathbf{f}} - \mathbf{f^o}]$ in (2.25) to get effective function difference denoted by $[\widehat{\mathbf{f}} - \mathbf{f^o}]_{\text{eff}}$ as follows,

$$\begin{aligned}
[\widehat{\mathbf{f}} - \mathbf{f^o}] &= \mathbf{BG} = \mathbf{f} - \mathbf{f}^o + \mathbf{BS}, \\
[\widehat{\mathbf{f}} - \mathbf{f^o}]_{\text{eff}} &= [\widehat{\mathbf{f}} - \mathbf{f}^o] - \sum_{a \in \mathcal{A}} \left[\sum_{k : a \in \mathcal{N}_k} G_{ka}(x) - \sum_{j \in \mathcal{N}_a} G_{aj}(x) \right] = \tilde{\mathbf{B}}\tilde{\mathbf{G}},
\end{aligned} \tag{2.26}$$

where if b entries of $\mathbf{G}$ were fixed then $\tilde{\mathbf{G}}$ is a $(|\mathcal{E}| - b) \times 1$ vector and $\tilde{\mathbf{B}}$ is a matrix with dimension $n \times (|\mathcal{E}| - b)$.[10] The columns deleted from $\mathbf{B}$ correspond to the edges that are incident to and from the corrupted nodes. Hence, $\tilde{\mathbf{B}}$ represents the incidence of a graph with these edges deleted.

We know from the $\kappa(\mathcal{G}_0) > \tau$ of the graph, that $\tilde{\mathbf{B}}$ connects all the non-adversarial agents into a connected

[8] $\mathbf{f}$ and $\mathbf{f}^o$ are dissimilar and arbitrarily different.

[9] The only entries of $\mathbf{G}$, that are undecided at this stage are included in $\tilde{\mathbf{G}}$. These are functions $G_{i,j}(x)$ such that i, j are both non-adversarial.

[10] Total number of edges incident to and from corrupted nodes is b. We fixed them to be the same as corresponding entries from $\mathbf{S}$, since the adversary can observe them.

component.[11] Since, the remaining edges form a connected component, the edges can be split into two groups. A group with edges that form a spanning tree over the good nodes (agents) and all other edges in the other group (see Remark 1 and Figure 2.2). Let $\tilde{\mathbf{B}}_{\mathrm{ST}}$ represent the incidence matrix of the spanning tree and $\tilde{\mathbf{G}}_{\mathrm{ST}}$ represents the arbitrary functions corresponding to the edges of the spanning tree.[12] $\tilde{\mathbf{B}}_{\mathrm{EE}}$ represents the incidence matrix formed by all other edges and $\tilde{\mathbf{G}}_{\mathrm{EE}}$ represents the arbitrary functions related to all other edges. We get from (2.26),

$$[\widehat{\mathbf{f}} - \mathbf{f}^{\mathbf{o}}]_{\mathrm{eff}} = \begin{bmatrix} \tilde{\mathbf{B}}_{\mathrm{ST}} & \tilde{\mathbf{B}}_{\mathrm{EE}} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{G}}_{\mathrm{ST}} \\ \tilde{\mathbf{G}}_{\mathrm{EE}} \end{bmatrix} = \tilde{\mathbf{B}}_{\mathrm{ST}} \tilde{\mathbf{G}}_{\mathrm{ST}} + \tilde{\mathbf{B}}_{\mathrm{EE}} \tilde{\mathbf{G}}_{\mathrm{EE}}. \tag{2.27}$$

We now arbitrary assign functions to elements of $\tilde{\mathbf{G}}_{\mathrm{EE}}$ and then compute the arbitrary weights for $\tilde{\mathbf{G}}_{\mathrm{ST}}$. We know that the columns of $\tilde{\mathbf{B}}_{\mathrm{ST}}$ are linearly independent, since $\tilde{\mathbf{B}}_{\mathrm{ST}}$ is the incidence matrix of a spanning tree (cf. Lemma 2.5 in [80]). Hence, the left pseudoinverse of $\tilde{\mathbf{B}}_{\mathrm{ST}}$ exists; and $\tilde{\mathbf{B}}_{\mathrm{ST}}^{\dagger} \tilde{\mathbf{B}}_{\mathrm{ST}} = \mathbb{I}$, giving us the solution for $\tilde{\mathbf{G}}_{\mathrm{ST}}$, [13,14]

$$\tilde{\mathbf{G}}_{\mathrm{ST}} = \tilde{\mathbf{B}}_{\mathrm{ST}}^{\dagger} \left[[\widehat{\mathbf{f}} - \mathbf{f}^{\mathbf{o}}]_{\mathrm{eff}} - \tilde{\mathbf{B}}_{\mathrm{EE}} \tilde{\mathbf{G}}_{\mathrm{EE}} \right]. \tag{2.28}$$

Using the construction shown above, for any $\mathbf{f}^o$ we can construct $\mathbf{G}$ such that the execution as seen by corrupted nodes is exactly the same as the original problem where the objective is $\mathbf{f}$ and the arbitrary functions are $\mathbf{S}$. An honest-but-curious adversary cannot distinguish between two executions involving $\mathbf{f}^o$ and $\mathbf{f}$ leading to privacy claim.

Necessity (P1): The proof of necessity here (for P1) follows the proof of necessity for P2. We prove this statement by contradiction. Assume that a node i has degree τ and we have $|\mathcal{A}| = \tau$ adversaries.

Agents generate correlated noise function as shown in the obfuscation step. All the perturbation functions $S_{ji}(x)$ shared by agent i (with agents in neighborhood of i) are observed by the adversary. Hence, the true objective function is easily estimated by using,

$$f_i(x) = \widehat{f}_i - \left(\sum_{j=1}^{j=\tau} S_{ji}(x) - \sum_{j=1}^{j=\tau} S_{ij}(x) \right).$$

This gives us contradiction. Hence, degree $> \tau$ is necessary.

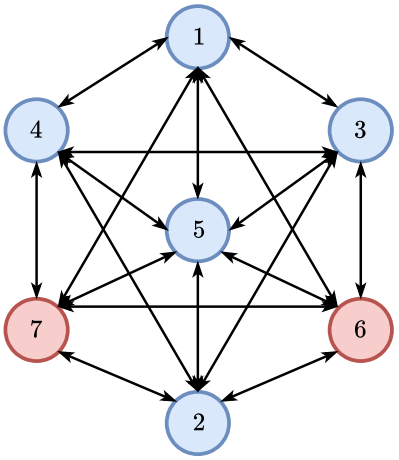
Sufficiency (P1): We can use a construction similar to the sufficiency proof for P2. Instead of considering all

[11] The adversarial nodes become disconnected due to the deletion of edges incident on corrupted nodes (previous step).

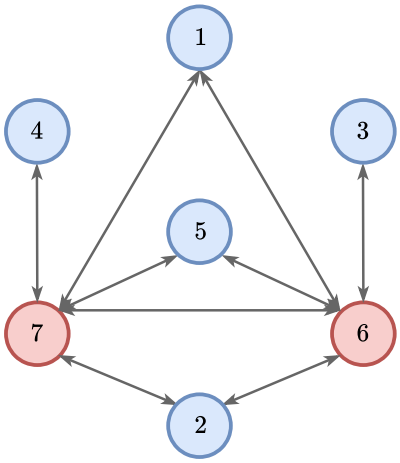
[12] Its columns correspond to the edges that form spanning tree.

[13] $A^{\dagger}$ represents the pseudoinverse of matrix A.

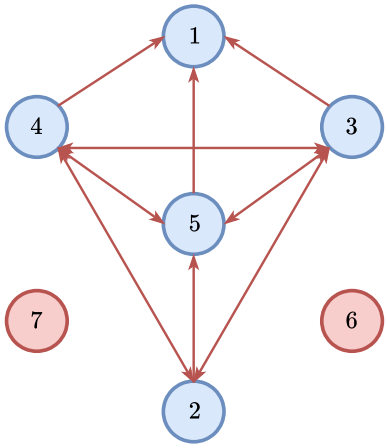
[14] An alternate way to look at this would be to see that $\tilde{\mathbf{B}}_{\mathrm{ST}}^{T} \tilde{\mathbf{B}}_{\mathrm{ST}}$ represents the edge Laplacian [81] of the spanning tree. The edge Laplacian of an acyclic graph is non-singular and this also proves that left-pseudoinverse of $\tilde{\mathbf{B}}_{\mathrm{ST}}$ exists.



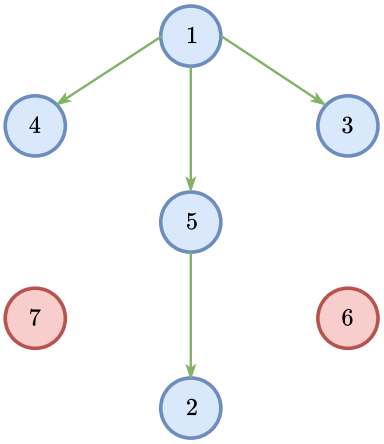
(a) Example



(b) Step-1. Fix $\mathbf{G}_{\mathscr{A},l}$ and $\mathbf{G}_{l,\mathscr{A}}$.



(c) Step-2. Arbitrarily select $\mathbf{G}_{\mathrm{EE}}$.



(d) Step-3. Solve for $\mathbf{G}_{\mathrm{ST}}$ using (2.28).

Figure 2.2: An example of construction used for proving Theorem 5. Network has $n = 7$ nodes and a adversary with $\tau = 2$ corrupted nodes. The graph satisfies $\kappa(\mathcal{G}) > 2$.

objective functions $\mathbf{f}$, we consider $f_i(x)$. Note that deleting adversarial agents may lead to fragmented graph components. We need to run the construction procedure mentioned above for each connected component.

□

In what follows, we summarize the construction used in our sufficiency proof. This method allows one to construct similar execution for many different sets of private objective functions.

Remark 1 (Method for Constructing $\mathbf{G}$)**.** *We present an example for the construction used in the above proof. Let us consider a system of $n = 7$ agents communicating under a topology with $\kappa(\mathcal{G}) > 2$ (see Figure 2.2a). An adversary with two corrupted nodes ($\mathcal{A} = \{6, 7\}$, $\tau = 2$) is a part of the system. We can divide the task of constructing $\mathbf{G}$ into three steps:*

1. *Fix G_{al} and G_{la} (links incident on corrupted nodes) to be the corresponding entries in $\boldsymbol{S}$.*

2. Arbitrarily select the noise/perturbation functions corresponding to non spanning tree edges ($\mathbf{G}_{\text{EE}}$).

3. Solve for the functions corresponding to the spanning tree ($\mathbf{G}_{\text{ST}}$) using (2.28).

We first follow Step 1 and fix $G_{ka} = S_{ka}$ and $G_{aj} = S_{aj}$ (where $k : a \in \mathcal{N}_k$ and $j \in \mathcal{N}_a$, for all $a \in \mathcal{A}$). Step 1 follows form the fact that the adversary observes S_{ka} and S_{aj}, and hence they need to be same in both executions. This is followed by substituting the known entries in $\mathbf{G}$ and subtract them from the left-hand side as shown in (2.26). *This corresponds to the deletion of all incoming and outgoing edges from the corrupted nodes. The incidence matrix of this new graph is denoted by $\tilde{\mathbf{B}}$. The edges in the new graph can be decomposed into two groups - a set containing edges that form a spanning tree and a set that contains all other edges. This is seen in Figure 2.2c where the red edges are all the remaining links (incidence matrix, $\tilde{\mathbf{B}}_{\text{EE}}$); and Figure 2.2d where the green edges form a spanning tree (incidence matrix, $\tilde{\mathbf{B}}_{\text{ST}}$) with Agent 1 as the root and all other non-adversarial agents as its leaves (agents 2, 3, 4, 5).*

2.4.4 Proof of Theorem 4

The proofs for FS-DGD follow similar structure as the proofs for FS-GP. We begin with disagreement lemma, followed by an iterate lemma and convergence result. Let ρ be the smallest non-zero entry of B_k, the doubly stochastic weight matrix used for consensus based averaging. We define transition matrix product, $\Phi(k, s)$, as the product of doubly stochastic weight matrices B_k, i.e. $\forall\ k \geq s > 0$

$$\Phi(k, s) = B_k B_{k-1} \dots B_{s+1} B_s.$$

We first note two important results from literature. The first result relates to convergence of non-negative sequences (Lemma 4) and the second result describes the linear convergence of transition matrix to $\frac{1}{n}\mathbf{1}\mathbf{1}^T$ (Lemma 5).

Lemma 4 (Lemma 3.1, [38]). *Let $\{\zeta_k\}$ be a non-negative scalar sequence. If $\sum_{k=0}^{\infty} \zeta_k < \infty$ and $0 < \beta < 1$, then $\sum_{k=0}^{\infty} \left(\sum_{j=0}^{k} \beta^{k-j} \zeta_j\right) < \infty$.*

Lemma 5 (Corollary 1, [82]). *Let the graph $\mathcal{G}$ be connected, then,*

1. $\lim_{k\to\infty} \Phi(k, s) = \frac{1}{n}\mathbf{1}\mathbf{1}^T$ for all $s > 0$.

2. $|\Phi(k, s)[i, j] - \frac{1}{n}| \leq \theta\beta^{k-s+1}$ for all $k \geq s > 0$, where $\theta = (1 - \frac{\rho}{4n^2})^{-2}$ and $\beta = (1 - \frac{\rho}{4n^2})$.

Lemma 6 (Disagreement Lemma, [20]). *Consider the FS-DGD algorithm, let $\mathcal{G}$ be connected, then for $\theta < 1$, $\beta < 1$ and doubly stochastic matrices B_k, we have,*

$$\max_{j\in\mathcal{V}} \|x_{k+1}^j - \overline{x}_{k+1}\| \leq n\theta\beta^k \max_{i\in\mathcal{V}} \|x_1^i\| + n\theta\tilde{L} \sum_{l=2}^{k} \beta^{k+1-l}\alpha_{l-1} + 2\alpha_k\tilde{L}.$$

Observe that the Disagreement Lemma (Lemma 1 in [20]) uses $L + \Delta$ instead of $\tilde{L}$. They are the same and elaborate the effect of noise function on the gradient bound for perturbed objective functions.

Proof. Define for all $j \in \mathcal{V}$ and all k,

$$z_{k+1}^j = x_{k+1}^j - \sum_{i=1}^n B_k[j,i] x_k^i \tag{2.29}$$

$$\implies x_{k+1}^j = z_{k+1}^j + \sum_{i=1}^n B_k[j,i] x_k^i. \tag{2.30}$$

We then unroll the iterations to get x_{k+1}^j as a function of z_{k+1}^j, and x_k^i and doubly stochastic weight matrix at current and previous iteration,

$$x_{k+1}^j = z_{k+1}^j + \sum_{i=1}^n \left[B_k[j,i] \left(z_k^i + \sum_{l=1}^n B_{k-1}[i,l] x_{k-1}^l \right) \right].$$

We perform the above mentioned unrolling successively and use the definition of transition matrix $\Phi(k,s)$,

$$x_{k+1}^j = z_{k+1}^j + \sum_{i=1}^n \Phi(k,1)[j,i] x_1^i + \sum_{l=2}^k \left[\sum_{i=1}^n \Phi(k,l)[j,i] z_l^i \right]. \tag{2.31}$$

Note that $\Phi(1,1) = B_1$. We verify the expression for $k = 1$, and we get the relationship $x_2^j = z_2^j + \sum_{i=1}^n \Phi(1,1)[j,i] x_1^i$.

We can write the relation for iterate average, $\overline{x}_k$, and use doubly stochastic nature of B_k to get,

$$\begin{aligned} \overline{x}_{k+1} = \frac{1}{n} \sum_{j=1}^n x_{k+1}^j &= \frac{1}{n} \sum_{j=1}^n \left(\sum_{i=1}^n B_k[j,i] x_k^i + z_{k+1}^j \right) = \frac{1}{n} \left(\sum_{i=1}^n \left(\sum_{j=1}^n B_k[j,i] \right) x_k^i + \sum_{j=1}^n z_{k+1}^j \right) \\ &= \overline{x}_k + \frac{1}{n} \sum_{j=1}^n z_{k+1}^j = \overline{x}_1 + \frac{1}{n} \sum_{l=2}^{k+1} \sum_{j=1}^n z_l^j. \end{aligned} \tag{2.32}$$

Using relations for $\overline{x}_{k+1}$ (2.32) and x_{k+1}^j (2.31) to get an expression for the disagreement. We further use the property of norm $\| \sum a \| \leq \sum \|a\|$, $\forall a$, to get,

$$\|x_{k+1}^j - \overline{x}_{k+1}\| \leq \sum_{i=1}^n \left| \frac{1}{n} - \Phi(k,1)[j,i] \right| \|x_1^i\| + \sum_{l=2}^k \sum_{i=1}^n \left| \frac{1}{n} - \Phi(k,l)[j,i] \right| \|z_l^i\| + \|z_{k+1}^j\| + \frac{1}{n} \sum_{i=1}^n \|z_{k+1}^i\|. \tag{2.33}$$

We use Lemma 5 to bound terms of type $\left| \frac{1}{n} - \Phi(k,l)[j,i] \right|$ and $\max_{i \in \mathcal{V}} \|x_1^i\|$ to bound $\|x_1^i\|$, to get,

$$\|x_{k+1}^j - \overline{x}_{k+1}\| \leq n\theta\beta^k \max_{i \in \mathcal{V}} \|x_1^i\| + \theta \sum_{l=2}^k \beta^{k+1-l} \sum_{i=1}^n \|z_l^i\| + \|z_{k+1}^j\| + \frac{1}{n} \sum_{i=1}^n \|z_{k+1}^i\|. \tag{2.34}$$

We now bound each of the norms $\|z_k^j\|$, using the fact that $v_k^j \in \mathcal{X}$, the non-expansive property of projection operator and gradient boundedness,

$$\begin{aligned}\|z_{k+1}^j\| &= \|\mathcal{P}_{\mathcal{X}}[v_k^j - \alpha_k \left(\nabla \tilde{f}_j(v_k^j)\right)] - v_k^j\| \\ &\leq \alpha_k \|\nabla \tilde{f}_j(v_k^j)\| \\ &\leq \alpha_k \tilde{L}. \end{aligned} \tag{2.35}$$

Note that we used the gradient boundedness of perturbation objective functions to obtain the above relation. Combining (2.34) and (2.35),

$$\max_{j\in\mathcal{V}} \|x_{k+1}^j - \overline{x}_{k+1}\| \leq n\theta\beta^k \max_{i\in\mathcal{V}} \|x_1^i\| + n\theta\tilde{L} \sum_{l=2}^{k} \beta^{k+1-l}\alpha_{l-1} + 2\alpha_k\tilde{L}.$$

□

Next, we develop an iterate lemma to construct a bound on distance between iterate and optimum.

Lemma 7 (Iterate Lemma, [20])**.** *Consider the distributed optimization problem presented in Section 2.2. Let* $\eta_k^2 = \sum_{j=1}^n \|x_k^j - y\|^2$ *and* $\delta_k^j = x_k^j - \overline{x}_k$*. We have for all* $v \in \mathcal{X}$ *and* $k > 0$*,*

$$\eta_{k+1}^2 \leq (1 + F_k)\,\eta_k^2 - 2\alpha_k\,(f(\overline{v}_k) - f(y)) + H_k,$$

where $F_k = \alpha_k \tilde{N} \max_{j\in\mathcal{V}} \|\delta_k^j\|$ *and* $H_k = 2\alpha_k n \left(\tilde{L} + \frac{\tilde{N}}{2}\right) \max_{j\in\mathcal{V}} \|\delta_k^j\| + \alpha_k^2 n \tilde{L}^2$*.*

Observe that Iterate lemma (Lemma 2 in [20]) as seen before uses $L + \Delta$ instead of $\tilde{L}$.

Proof. To simplify analysis, we adopt the following notation,

$$\eta_k^2 = \sum_{j=1}^n \|x_k^j - y\|^2, \tag{2.36}$$

$$\xi_k^2 = \sum_{j=1}^n \|v_k^j - y\|^2. \tag{2.37}$$

Note, η_k and ξ_k are both functions of y, however for simplicity we do not explicitly show this dependence. Moreover, we also use the notation $\delta_k^j = x_k^j - \overline{x}_k$.

Note, $\mathcal{P}_{\mathcal{X}}[y] = y$ for all $y \in \mathcal{X}$. Using the non-expansive property of the projection operator, and the projected gradient descent in FS-DGD algorithm to get,

$$\begin{aligned}\|x_{k+1}^j - y\|^2 &= \|\mathcal{P}_{\mathcal{X}}\left[v_k^j - \alpha_k \left(\nabla \tilde{f}_j(v_k^j)\right)\right] - y\|^2 \leq \|v_k^j - \alpha_k \left(\nabla \tilde{f}_j(v_k^j)\right) - y\|^2 \\ &= \|v_k^j - y\|^2 + \alpha_k^2 \|\nabla \tilde{f}_j(v_k^j)\|^2 - 2\alpha_k \left(\nabla \tilde{f}_j(v_k^j)\right)^T (v_k^j - y). \end{aligned} \tag{2.38}$$

Now we add the inequalities (2.38) for all agents $j = 1, 2, \ldots, n$ followed by using expressions for η_k and ξ_k (2.36), (2.37). Next we use the boundedness of gradients, to get the following inequality,

$$\begin{aligned}
\eta_{k+1}^2 &\leq \xi_k^2 + \sum_{j=1}^{n} \alpha_k^2 \|\nabla \tilde{f}_j(v_k^j)\|^2 - 2\alpha_k \sum_{j=1}^{n} \left(\nabla \tilde{f}_j(v_k^j)\right)^T (v_k^j - y) \\
&\leq \xi_k^2 + \sum_{j=1}^{n} \alpha_k^2 \tilde{L}^2 - 2\alpha_k \sum_{j=1}^{n} \left(\nabla \tilde{f}_j(v_k^j)\right)^T (v_k^j - y) \\
\eta_{k+1}^2 &\leq \xi_k^2 + \alpha_k^2 n \tilde{L}^2 - 2\alpha_k \sum_{j=1}^{n} (\nabla \tilde{f}_j(v_k^j))^T (v_k^j - y). \qquad (2.39)
\end{aligned}$$

We use consensus relationship used for information fusion. We know that in D-dimension the consensus step can be rewritten using Kronecker product of D-dimension identity matrix (I_D) and the doubly stochastic weight matrix (B_k) [83]. Consider the following notation of vectors. We use bold font to denote a vector that is stacked by its coordinates. As an example, consider three vectors in $\mathbb{R}^3$ given by $a = [a_x,\ a_y,\ a_z]^T$, $b = [b_x,\ b_y,\ b_z]^T$, $c = [c_x,\ c_y,\ c_z]^T$. Let $\mathbf{a}$ be a vector of a, b and c stacked by coordinates, then it is defined as $\mathbf{a} = [a_x,\ b_x,\ c_x,\ a_y,\ b_y,\ c_y,\ a_z,\ b_z,\ c_z]^T$. Similarly we can write stacked model parameter vector as, $\mathbf{x}_k = [x_k^1[1], x_k^2[1], \ldots, x_k^n[1], x_k^1[2], x_k^2[2], \ldots, x_k^n[2], \ldots, x_k^1[D], \ldots, x_k^n[D]]^T$. Next, we write the consensus term using the new notation and Kronecker products and compare norms of both sides (2-norm),

$$\begin{aligned}
\mathbf{v}_k &= (I_D \otimes B_k)\mathbf{x}_k \qquad (2.40) \\
\mathbf{v}_k - \mathbf{y} &= (I_D \otimes B_k)(\mathbf{x}_k - \mathbf{y}) \\
\|\mathbf{v}_k - \mathbf{y}\|_2^2 &= \|(I_D \otimes B_k)(\mathbf{x}_k - \mathbf{y})\|_2^2 \\
&\leq \|(I_D \otimes B_k)\|_2^2 \|(\mathbf{x}_k - \mathbf{y})\|_2^2. \qquad (2.41)
\end{aligned}$$

We use the property of eigenvalues of Kronecker product of matrices. The eigenvalues of $I_D \otimes B_k$ are essentially D copies of eigenvalues of B_k. Since B_k is a doubly stochastic matrix, its eigenvalues are upper bounded by 1. Recall that $\|A\|_2 = \sqrt{\lambda_{\max}(A^\dagger A)}$ where $A^\dagger$ represents the conjugate transpose of matrix A and $\lambda_{\max}$ represents the maximum eigenvalue. Observe that $I_D \otimes B_k$ is a doubly stochastic matrix and $(I_D \otimes B_k)^\dagger (I_D \otimes B_k)$ is also doubly stochastic matrix since product of two doubly stochastic matrices is also doubly stochastic. Clearly, $\|(I_D \otimes B_k)\|_2^2 = \lambda_{\max}((I_D \otimes B_k)^\dagger (I_D \otimes B_k)) \leq 1$,[15] allowing us to show

$$\xi_k^2 = \|\mathbf{v}_k - \mathbf{y}\|_2^2 \leq \|(\mathbf{x}_k - \mathbf{y})\|_2^2 = \eta_k^2. \qquad (2.42)$$

[15] An alternate way to prove this inequality would be to follow the same process used to prove (2.46) except that we start with squared terms and use the doubly-stochasticity of B_k. We include this proof in Appendix H in [69].

Merging the inequalities in (2.42) and (2.39), we get,

$$\eta_{k+1}^2 \leq \eta_k^2 + \alpha_k^2 n \tilde{L}^2 \underbrace{-2\alpha_k \sum_{j=1}^{n} (\nabla \tilde{f}_j(v_k^j))^T (v_k^j - y)}_{\Lambda}. \tag{2.43}$$

Typically, at this step one would use convexity of $\tilde{f}_j(x)$ to simplify the term Λ in (2.43). However, since $\tilde{f}_j(x)$ can be non-convex we need to follow a few more steps before we arrive at the iterate lemma.

Consider the iterates v_k^j, the average $\overline{v}_k \triangleq (1/n)\sum_{j=1}^n v_k^j$ and the deviation of iterate from the average,

$$q_k^j = v_k^j - \overline{v}_k. \tag{2.44}$$

We now derive a simple inequality here that will be used later.

$$\begin{aligned}
\|q_k^j\| = \|v_k^j - \overline{v}_k\| &= \|\sum_{i=1}^{n} B_k[j,i] x_k^i - \overline{v}_k\| \\
&\overset{(a)}{=} \|\sum_{i=1}^{n} B_k[j,i] x_k^i - \overline{x}_k\| \\
&\overset{(b)}{\leq} \sum_{i=1}^{n} B_k[j,i] \|x_k^i - \overline{x}_k\| \\
&\leq \left(\sum_{i=1}^{n} B_k[j,i]\right) \max_{i \in \mathcal{V}} \|x_k^i - \overline{x}_k\| \\
&\overset{(c)}{\leq} \max_{i \in \mathcal{V}} \|x_k^i - \overline{x}_k\|,
\end{aligned} \tag{2.45}$$

where (a) follows from the fact that $\overline{x}_k = \overline{v}_k$, (b) from $\|\sum_i a_i\| \leq \sum_i \|a_i\|$ for all a_i and (c) follows from column stochasticity of B_k. Note that similarly, we can derive another inequality that will be used later.

$$\begin{aligned}
\sum_{j=1}^{n} \|v_k^j - y\| &\overset{(a)}{=} \sum_{j=1}^{n} \|\sum_{i=1}^{n} B_k[j,i] x_k^i - y\| \\
&\overset{(b)}{=} \sum_{j=1}^{n} \|\sum_{i=1}^{n} B_k[j,i] \left(x_k^i - y\right) \| \\
&\overset{(c)}{\leq} \sum_{j=1}^{n} \sum_{i=1}^{n} B_k[j,i] \|x_k^i - y\| \\
&= \sum_{i=1}^{n} \left(\sum_{j=1}^{n} B_k[j,i]\right) \|x_k^i - y\| \\
&\overset{(d)}{\leq} \sum_{i=1}^{n} \|x_k^i - y\|,
\end{aligned} \tag{2.46}$$

where (a) follows from information fusion (consensus step) in FS-DGD, (b) follows from B_k being row stochas-

tic, (c) from $\|\sum_i a_i\| \leq \sum \|a_i\|$ for all a_i and finally (d) follows from column stochasticity of B_k.
We use gradient Lipschiztness assumption and write the following relation,

$$\nabla \tilde{f}_j(v_k^j) = \nabla \tilde{f}_j(\overline{v}_k) + l_k^j, \tag{2.47}$$

where l_k^j is the (vector) difference between gradient computed at v_k^j (i.e. $\nabla \tilde{f}_j(v_k^j)$) and the gradient computed at $\overline{v}_k$ (i.e. $\nabla \tilde{f}_j(\overline{v}_k)$). Next, we bound the vector l_k^j, using Lipschitzness of gradients of perturbed functions,

$$\begin{aligned}
\max_{j\in\mathcal{V}} \|l_k^j\| &= \max_{j\in\mathcal{V}} \|\nabla \tilde{f}_j(v_k^j) - \nabla \tilde{f}_j(\overline{v}_k)\| \\
&\leq \max_{j\in\mathcal{V}} \tilde{N} \|v_k^j - \overline{v}_k\| &(2.48)\\
&\overset{(a)}{\leq} \tilde{N} \max_{j\in\mathcal{V}} \|x_k^j - \overline{x}_k\|, &(2.49)
\end{aligned}$$

where (a) follows from two facts, $\overline{x}_k = \overline{v}_k$ and $\|v_k^j - \overline{v}_k\| \leq \max_{i\in\mathcal{V}} \|x_k^i - \overline{x}_k\|$, see (2.45).

We use above expressions to bound the term Λ, in (2.43). We use $v_k^j = \overline{v}_k + q_k^j$ from (2.44) and the gradient relation in (2.47) to get,

$$\begin{aligned}
&\Lambda = -2\alpha_k \sum_{j=1}^{n} (\nabla \tilde{f}_j(v_k^j))^T (v_k^j - y) = 2\alpha_k \sum_{j=1}^{n} \left[(\nabla \tilde{f}_j(\overline{v}_k) + l_k^j)^T (y - \overline{v}_k - q_k^j) \right] = 2\alpha_k \left[T_1 + T_2 + T_3 \right], \\
&\text{where } T_1 = \sum_{j=1}^{n} \left(\nabla \tilde{f}_j(\overline{v}_k) \right)^T (y - \overline{v}_k), \quad T_2 = \sum_{j=1}^{n} \left(\nabla \tilde{f}_j(\overline{v}_k) \right)^T (-q_k^j), \text{ and} \\
&T_3 = \sum_{j=1}^{n} (l_k^j)^T (y - \overline{v}_k - q_k^j) = \sum_{j=1}^{n} (l_k^j)^T (y - v_k^j).
\end{aligned}$$

Individually T_1, T_2 and T_3 can be bound as follows,

$$T_1 \overset{(a)}{=} \sum_{j=1}^{n} \left(\nabla \tilde{f}_j(\overline{v}_k) \right)^T (y - \overline{v}_k) = \nabla f(\overline{v}_k)^T (y - \overline{v}_k) \overset{(b)}{\leq} f(y) - f(\overline{v}_k), \tag{2.50}$$

where (a) follows from aggregate invariance property of obfuscation step as seen in (2.5) and (b) follows from convexity of $f(x)$.

Next we bound T_2 as follows,

$$\begin{aligned}
T_2 &= \sum_{j=1}^{n} \left(\nabla \tilde{f}_j(\overline{v}_k) \right)^T (-q_k^j) \\
&\overset{(a)}{\leq} \sum_{j=1}^{n} \|\nabla \tilde{f}_j(\overline{v}_k)\| \|(-q_k^j)\| \\
&\overset{(b)}{\leq} \tilde{L} n \max_{j\in\mathcal{V}} \|q_k^j\| \overset{(c)}{\leq} \tilde{L} n \max_{j\in\mathcal{V}} \|x_k^j - \overline{x}_k\| = \tilde{L} n \max_{j\in\mathcal{V}} \|\delta_k^j\|, &(2.51)
\end{aligned}$$

where (a) follows from Cauchy-Schwarz Inequality, (b) follows boundedness of obfuscated objective function gradients and (c) uses expression (2.45). Finally,

$$\begin{aligned}
T_3 &= \sum_{j=1}^{n} (l_k^j)^T (y - v_k^j) \\
&\overset{(a)}{\leq} \max_{j\in\mathcal{V}} \|l_k^j\| \sum_{j=1}^{n} \|v_k^j - y\| \\
&\overset{(b)}{\leq} \tilde{N} \left(\max_{j\in\mathcal{V}} \|q_k^j\| \right) \sum_{j=1}^{n} \|v_k^j - y\| \\
&\overset{(c)}{\leq} \tilde{N} \left(\max_{j\in\mathcal{V}} \|x_k^j - \overline{x}_k\| \right) \sum_{j=1}^{n} \|v_k^j - y\| \\
&\overset{(d)}{\leq} \tilde{N} \left(\max_{j\in\mathcal{V}} \|x_k^j - \overline{x}_k\| \right) \left[\sum_{j=1}^{n} \|x_k^j - y\| \right] \\
&\overset{(e)}{\leq} \frac{\tilde{N}}{2} \left(\max_{j\in\mathcal{V}} \|\delta_k^j\| \right) \left[\sum_{j=1}^{n} \left(1 + \|x_k^j - y\|^2 \right) \right],
\end{aligned} \tag{2.52}$$

where (a) follows from property of norm, (b) follows from (2.48), (c) follows (2.45), (d) follows from (2.46) and (e) follows from the definition of δ_k^j and $2\|a\| \leq 1 + \|a\|^2$ to bound term T_3.

We combine the bounds on T_1, T_2 and T_3 from (2.50), (2.51) and (2.52) to get,

$$\begin{aligned}
\Lambda &\leq 2\alpha_k (f(y) - f(\overline{v}_k)) + 2\alpha_k n \tilde{L} \max_{j\in\mathcal{V}} \|\delta_k^j\| + \alpha_k \tilde{N} \max_{j\in\mathcal{V}} \|\delta_k^j\| \left[\sum_{j=1}^{n} \left(1 + \|x_k^j - y\|^2 \right) \right] \\
&\leq -2\alpha_k \left(f(\overline{v}_k) - f(y) \right) + 2\alpha_k n \tilde{L} \max_{j\in\mathcal{V}} \|\delta_k^j\| + \alpha_k \tilde{N} \max_{j\in\mathcal{V}} \|\delta_k^j\| \left[n + \eta_k^2 \right].
\end{aligned} \tag{2.53}$$

Recall from (2.43),

$$\eta_{k+1}^2 \leq \eta_k^2 + \alpha_k^2 n (\tilde{L})^2 \underbrace{-2\alpha_k \sum_{j=1}^{n} (\nabla \tilde{f}_j(v_k^j) - e_k^j)^T (v_k^j - y)}_{\Lambda}. \tag{2.54}$$

We replace Λ with its bound from (2.53), and use the fact that $\overline{x}_k = \overline{v}_k$ to we replace, $f(\overline{v}_k)$ with $f(\overline{x}_k)$,

$$\begin{aligned}
\eta_{k+1}^2 &\leq \eta_k^2 + \alpha_k^2 n (\tilde{L})^2 - 2\alpha_k \left(f(\overline{v}_k) - f(y) \right) + 2\alpha_k n (\tilde{L}) \max_{j\in\mathcal{V}} \|\delta_k^j\| + \alpha_k \tilde{N} \max_{j\in\mathcal{V}} \|\delta_k^j\| \left[n + \eta_k^2 \right] \\
&\leq \left(1 + \alpha_k \tilde{N} \max_{j\in\mathcal{V}} \|\delta_k^j\| \right) \eta_k^2 - 2\alpha_k \left(f(\overline{v}_k) - f(y) \right) + 2\alpha_k n (\tilde{L} + \frac{\tilde{N}}{2}) \max_{j\in\mathcal{V}} \|\delta_k^j\| + \alpha_k^2 n \tilde{L}^2 \\
&\leq (1 + F_k)\, \eta_k^2 - 2\alpha_k \left(f(\overline{x}_k) - f(y) \right) + H_k,
\end{aligned} \tag{2.55}$$

where $F_k = \alpha_k \tilde{N} \max_{j\in\mathcal{V}} \|\delta_k^j\|$ and $H_k = 2\alpha_k n (\tilde{L} + \frac{\tilde{N}}{2}) \max_{j\in\mathcal{V}} \|\delta_k^j\| + \alpha_k^2 n \tilde{L}^2$. □

We first state a claim about asymptotic behavior of iterates x_k^j and correspondingly of $\max_j \|\delta_k^j\|$. The claim is then used to prove Theorem 4.

Claim 8 (Consensus)**.** *All agents asymptoticaly reach consensus, i.e. for all $i \in [n],\ j \in [n]$,*

$$\lim_{k\to\infty} \max_j \|\delta_k^j\| = 0 \text{ and } \lim_{k\to\infty} \|x_k^i - x_k^j\| = 0.$$

Proof. We begin with the iterate disagreement relation in Lemma 6,

$$\max_{j\in\mathcal{V}} \|\delta_{k+1}^j\| \leq \underbrace{n\theta\beta^k \max_{i\in\mathcal{V}} \|x_1^i\|}_{V_1} + \underbrace{n\theta\tilde{L}\sum_{l=2}^{k} \beta^{k+1-l}\alpha_{l-1}}_{V_2} + \underbrace{2\alpha_k\tilde{L}}_{V_3}. \tag{2.56}$$

The first term V_1 decreases exponentially with k. Hence, for any $\epsilon > 0$, $\exists\ K_1 = \lceil \log_\beta \frac{\epsilon}{3n\theta \max_{i\in\mathcal{V}} \|x_1^i\|} \rceil$ such that, $\forall k > K_1$, we have $V_1 < \epsilon/3$.

For given $\xi = \epsilon(1-\beta)/6\beta n\theta\tilde{L}$, $\exists K_2$ such that, $\alpha_k < \xi$, $\forall\ k \geq K_2$, due to the non-increasing property of α_k and $\sum_k \alpha_k^2 < \infty$. Observe that,

$$\sum_{i=1}^{k-1} \left(\alpha_i\beta^{k-i}\right) = \underbrace{\left(\alpha_1\beta^{k-1} + \ \ldots + \alpha_{K_2-1}\beta^{k-K_2+1}\right)}_{A} + \underbrace{\left(\alpha_{K_2}\beta^{k-K_2} + \ldots + \alpha_{k-1}\beta^1\right)}_{B}.$$

We can bound the terms A and B individually as follows,

$$\begin{aligned} A &= \alpha_1\beta^{k-1} + \alpha_2\beta^{k-2} + \ldots + \alpha_{K_2-1}\beta^{k-K_2+1} \\ &\overset{(a)}{\leq} \alpha_1(\beta^{k-1} + \ldots + \beta^{k-K_2+1}) \\ &\leq \alpha_1\beta^{k-K_2+1}\left(\frac{1-\beta^{K_2-1}}{1-\beta}\right) \\ &\overset{(b)}{\leq} \frac{\alpha_1\beta^{k-K_2+1}}{1-\beta}, \end{aligned} \tag{2.57}$$

where (a) follows $\alpha_1 \geq \alpha_i\ \forall\ i \geq 1$ and (b) follows $0 \leq \beta < 1$. Moreover,

$$\begin{aligned} B &= \alpha_{K_2}\beta^{k-K_2} + \ldots + \alpha_{k-1}\beta^1 \\ &\overset{(a)}{<} \xi\beta\left(\frac{1-\beta^{k-K_2}}{1-\beta}\right) \\ &\overset{(b)}{\leq} \frac{\xi\beta}{1-\beta}, \end{aligned} \tag{2.58}$$

where (a) follows $\alpha_i < \xi,\ \forall i \geq K_2$ and (b) follows $0 \leq \beta < 1$. The right side of inequality in (2.57) is monotonically decreasing in k ($\beta < 1$) with limit 0 as $k \to \infty$. Hence $\exists K_3 > K_2$ such that $A < \epsilon/6n\theta\tilde{L}$, $\forall\ k \geq K_3$. We know $\alpha_i < \xi = \epsilon(1-\beta)/6\beta n\theta\tilde{L}$ for all $k \geq K_2$. Hence, following (2.58), $B \leq \frac{\xi\beta}{1-\beta} =$

$\frac{\epsilon\beta(1-\beta)}{6(1-\beta)\beta n\theta\tilde{L}} < \epsilon/6n\theta\tilde{L}$, for all $k \geq K_2$. Hence, $\exists K_4 = \max\{K_2, K_3\}$ such that $V_2 = n\theta\tilde{L}(A+B) < \epsilon/3$ for all $k > K_4$.

The third term in (2.56), V_3, decreases at the same rate as α_k. Hence, for any $\epsilon > 0$, $\exists\ K_6 = \min\{k|\alpha_k < \frac{\epsilon}{6\tilde{L}}\}$, such that $\forall k > K_6$, we have $V_3 < \epsilon/3$.

We have convergence based on $\epsilon - \delta$ definition of limits. For any $\epsilon > 0$, there exists $K_{\max} = \max\{K_1, K_5, K_6\}$ such that $\max_{j\in\mathcal{V}} \|\delta_k^j\| \leq V_1 + V_2 + V_3 < \epsilon$ for all $k \geq K_{\max}$. This implies that,

$$\lim_{k\to\infty} \max_{j\in\mathcal{V}} \|\delta_k^j\| = \lim_{k\to\infty} \max_{j\in\mathcal{V}} \|x_k^j - \overline{x}_k\| \leq 0.$$

Since, $\max_{j\in\mathcal{V}} \|\delta_k^j\| \geq 0$, the above statement implies $\lim_{k\to\infty} \max_{j\in\mathcal{V}} \|\delta_k^j\| = \lim_{k\to\infty} \max_{j\in\mathcal{V}} \|x_k^j - \overline{x}_k\| = 0$.

Now note that $\lim_{k\to\infty} \max_{j\in\mathcal{V}} \|x_k^j - \overline{x}_k\| = 0 \implies \lim_{k\to\infty} \|x_k^j - \overline{x}_k\| = 0\ \forall j$. Hence, we can also show that the following relationship holds,

$$\begin{aligned}
\lim_{k\to\infty} \|x_k^j - x_k^i\| &= \lim_{k\to\infty} \|(x_k^j - \overline{x}_k) + (\overline{x}_k - x_k^i)\| \\
&\overset{(a)}{\leq} \lim_{k\to\infty} (\|x_k^j - \overline{x}_k\| + \|\overline{x}_k - x_k^i\|) \\
&= \lim_{k\to\infty} \|x_k^j - \overline{x}_k\| + \lim_{k\to\infty} \|\overline{x}_k - x_k^i\| \\
&= 0,
\end{aligned}$$

where (a) follows from Triangle inequality.

Since, $\|x_k^j - x_k^i\| \geq 0$, for all i, j, the above statement implies $\lim_{k\to\infty} \|x_k^j - x_k^i\| = 0$. □

Proof of Theorem 4

Proof. We prove convergence using Lemma 2. We begin by using the relation between iterates given in Lemma 7 with $y = x^* \in \mathcal{X}^*$, and for $k \geq 1$,

$$\eta_{k+1}^2 \leq (1 + F_k)\,\eta_k^2 - 2\alpha_k\,(f(\overline{x}_k) - f(y)) + H_k. \tag{2.59}$$

Note, that F_k, H_k and $f(\overline{x}_k) - f(x^*)$ are scalar, real, and non-negative sequences. We check if the above inequality satisfies the conditions in Lemma 2 viz. $\sum_{k=1}^{\infty} F_k < \infty$ and $\sum_{k=1}^{\infty} H_k < \infty$. F_k and H_k are defined in Lemma 7 (2.55).

We first show that $\sum_{k=1}^{\infty} \alpha_k \max_{j\in\mathcal{V}} \|\delta_k^j\| < \infty$ using the expression for state disagreement from average

given in Lemma 6. We have,

$$\sum_{k=1}^{\infty}\alpha_k \max_{j\in\mathcal{V}}\|\delta_k^j\| = \alpha_1 \max_{j\in\mathcal{V}}\|\delta_1^j\| + \sum_{k=1}^{\infty}\alpha_{k+1}\max_{j\in\mathcal{V}}\|\delta_{k+1}^j\|$$

$$\leq \underbrace{\alpha_1 \max_{j\in\mathcal{V}}\|\delta_1^j\|}_{U_0} + \underbrace{n\theta \max_{i\in\mathcal{V}}\|x_1^i\|\sum_{k=1}^{\infty}\alpha_{k+1}\beta^k}_{U_1} + \underbrace{n\theta\tilde{L}\sum_{k=1}^{\infty}\alpha_{k+1}\sum_{l=2}^{k}\beta^{k+1-l}\alpha_{l-1}}_{U_2} + \underbrace{2\tilde{L}\sum_{k=1}^{\infty}\alpha_k\alpha_{k+1}}_{U_3}. \tag{2.60}$$

The first term U_0 is finite since $\max_{j\in\mathcal{V}}\|\delta_1^j\|$ and α_1 are both finite. The second term U_1 can be shown to be convergent by using the ratio test. We observe that,

$$\limsup_{k\to\infty}\frac{\alpha_{k+2}\beta^{k+1}}{\alpha_{k+1}\beta^k} = \limsup_{k\to\infty}\frac{\alpha_{k+2}\beta}{\alpha_{k+1}} < 1 \Rightarrow \sum_{k=1}^{\infty}\alpha_{k+1}\beta^k < \infty,$$

since $\alpha_{k+1}\leq\alpha_k$ and $\beta<1$. Now we move on to show that U_2 is finite. It follows from $\alpha_k\leq\alpha_l$ when $l\leq k$, and Lemma 4 and $\sum_k\alpha_k^2<\infty$,

$$\sum_{k=1}^{\infty}\alpha_{k+1}\sum_{l=2}^{k}\beta^{k+1-l}\alpha_{l-1} \leq \sum_{k=1}^{\infty}\sum_{l=2}^{k}\beta^{k+1-l}\alpha_{l-1}^2 < \infty.$$

U_3 is finite because $U_3\leq 2(L+\Delta)\sum_{k=1}^{\infty}\alpha_k^2<\infty$. Since we have shown, $U_1<\infty$, $U_2<\infty$, and $U_3<\infty$, we conclude $\sum_{k=1}^{\infty}\alpha_k\max_{j\in\mathcal{V}}\|\delta_k^j\|<\infty$.

Clearly, $\sum_{k=1}^{\infty}F_k<\infty$ and $\sum_{k=1}^{\infty}H_k<\infty$, since we proved that $\sum_{k=1}^{\infty}\alpha_k\max_{j\in\mathcal{V}}\|\delta_k^j\|<\infty$ and we know that $\sum_k\alpha_k^2<\infty$. We can now apply Lemma 2 and conclude $\sum_{k=1}^{\infty}2\alpha_k\left(f(\overline{x})-f(x^*)\right)<\infty$.

We use $\sum_{k=1}^{\infty}2\alpha_k\left(f(\overline{x})-f(x^*)\right)<\infty$ to show the convergence of the iterate-average to the optimum. Since we know $\sum_{k=1}^{\infty}\alpha_k=\infty$, it follows directly that $\liminf_{k\to\infty}f(\overline{x}_k)=f(x^*)=f^*$.

Also note that Lemma 2 states that η_k^2 has a finite limit. Let $\lim_{k\to\infty}\eta_k^2=\eta_{x^*}$ $(\forall x^*\in\mathcal{X}^*)$. We have,

$$\begin{aligned}
\lim_{k\to\infty}\eta_k^2 &= \lim_{k\to\infty}\sum_{i=1}^{n}\|x_k^i-x^*\|^2 = \lim_{k\to\infty}\sum_{i=1}^{n}\|\overline{x}_k+\delta_k^i-x^*\|^2\\
&= \lim_{k\to\infty}\sum_{i=1}^{n}\left[\|\overline{x}_k-x^*\|^2+\|\delta_k^i\|^2+2(\overline{x}_k-x^*)^T\delta_k^i\right]\\
&= \lim_{k\to\infty}\left[n\|\overline{x}_k-x^*\|^2+\sum_{i=1}^{n}\|\delta_k^i\|^2+2(\overline{x}_k-x^*)^T\left(\sum_{i=1}^{n}\delta_k^i\right)\right]\\
&\overset{(a)}{=} n\lim_{k\to\infty}\|\overline{x}_k-x^*\|^2+\lim_{k\to\infty}\sum_{i=1}^{n}\|\delta_k^i\|^2\\
&\overset{(b)}{=} n\lim_{k\to\infty}\|\overline{x}_k-x^*\|^2 \triangleq \eta_{x^*}^2,
\end{aligned}$$

where (a) follows from definition of $\overline{x}_k$ and $\sum_i\delta_k^i=0$, and (b) follows from consensus result Claim 8. From the statement above, we know $\lim_{k\to\infty}\|\overline{x}_k-x^*\|=\sqrt{\frac{\eta_{x^*}^2}{n}}$. This, along with $\liminf_{k\to\infty}f(\overline{x}_k)=f(x^*)$

proves that $\overline{x}_k$ converges to a point in $\mathcal{X}^*$.

We know from Claim 8 that the agents agree to a parameter vector asymptotically (i.e. $x_k^j \to x_k^i,\ \forall i \neq j$ as $k \to \infty$). Hence, all agents agree to the iterate average. This along with the convergence of iterate-average to an optimal solution gives us that all agents converge to a point in optimal set $\mathcal{X}^*$ (i.e. $x_k^j \to x^* \in \mathcal{X}^*,\ \forall j,$ as $k \to \infty$). This completes the proof of Theorem 4. □

2.5 Conclusions

We present function sharing FS methodology that involves an obfuscation step followed by a standard distributed optimization algorithm. The obfuscation step transforms private objective functions $f_i(x)$ to $\tilde{f}_i(x)$ in order to hide the private objective functions $f_i(x)$. The obfuscation step requires only one round of secure function transmission resulting in low computational and communication overheads. FS algorithms satisfy *privacy via non-identifiability* (Definition 1) for protecting privacy of local objective functions for individual nodes and group of nodes. Moreover, we show that FS algorithms allow both privacy and correctness (accuracy) to be achieved simultaneously.

We propose FS-GP, a privacy-preserving distributed optimization algorithm for time-varying directed graphs. We prove asymptotic convergence of iterates to the optimum (Theorem 1). We also characterize the finite-time convergence rate for FS-GP (Theorem 4) and show that it is similar to rate achieved by GP [9]. The privacy provided by FS-GP algorithm is dependent only on connectivity of $\mathcal{G}$ at time 0 (obfuscation step). We show that any strict subset of non-adversarial nodes, needs to have at least $\tau + 1$ neighbors (necessary and sufficient) for privacy (Theorem 2). The finite-time convergence rate $\mathcal{O}(\tilde{L}^2 \log(T)/\sqrt{T})$ has a multiplicative factor of $\tilde{L}2$ instead of L^2. As the Lipschitz constant for perturbed objective function $\tilde{L}$ is larger than that of original functions L, this results in a slowdown of convergence rate. This is the price we pay for privacy.

We propose FS-DGD, a privacy-preserving distributed optimization algorithm for undirected graphs. We prove asymptotic convergence of iterates to the optimum (Theorem 4). We also characterize necessary and sufficient conditions for privacy of local objective functions (Theorem 5).

CHAPTER 3

PRIVATE OPTIMIZATION FOR FEDERATED LEARNING

3.1 Introduction

Machine learning has recently found applications in increasingly many fields ranging from personal preference predictions to finance and healthcare. Machine learning algorithms are powered by rich datasets. These datasets are often highly sensitive and demand strict privacy protections. It has becoming increasingly important to design algorithms and systems that will honor privacy considerations. In this chapter we present a synchronous distributed learning and optimization algorithm with privacy properties for parameter server architecture.

In a distributed learning scenario, the data is segregated among several machines, possibly mobile devices [84]. Clients compute updates based on local data and iteratively improve the model. This quest to find the best predictive model, essentially implies solving the following optimization problem,

$$\text{find } x^* \in \arg\min \sum_{i=1}^{C} f_i(x),$$

where $f_i(x)$ is the local objective function of a client i, known only to client i ($1 \leq i \leq C$). The vector x parameterizes the predictive model and x^* represents the best "model parameters". The objective function of agent i is the loss function over the dataset stored at agent i for model parameters x. Observe that often the data stored at a client is large and gradient update computation is very expensive. In such a scenario, one may replace the gradient update with an unbiased estimator of the gradient. This is usually done by dividing the dataset into several batches and then randomly picking a batch to represent the dataset for gradient computation. The gradient computed over a batch is called stochastic gradient. It is an unbiased estimate of gradient and used in its place in learning algorithms.

Distributed optimization is often attractive due to the reduced communication requirements, since the agents (clients) communicate updates that are often much smaller in size than each agent's local dataset that characterizes its local objective (loss) function. Moreover, distributed methods provide scalability with respect to number of participants and naturally fit the fragmented and segregated data sources [26, 32, 33].

However, the updates transmitted by the clients often can give strong indications of the dataset (loss function) that generated the update and hence classical optimization algorithms are vulnerable to privacy

violations [35].

In this chapter we present POLAR-SGD (Private Optimization and Learning Algorithm - Stochastic Gradient Descent) that protects privacy by obfuscating the updates transmitted by the clients. In particular, we use correlated additive and multiplicative perturbations to obfuscate the stochastic gradient provided by clients. Introduction of perturbation helps clients protect the privacy of their information while the correlated nature of perturbations helps us maintain correctness.

3.1.1 Our Contributions

We present POLAR-SGD (Private Optimization and Learning Algorithm - Stochastic Gradient Descent). It is a synchronous protocol where clients use correlated additive and multiplicative perturbations to obfuscate the stochastic gradient. These obfuscated stochastic gradients are uploaded to multiple parameter servers, who then use consensus iteration and projected stochastic gradient descent to learn predictive models.

We prove convergence of POLAR-SGD under a few conditions on the perturbations. While other perturbation based algorithms such as differential privacy incur accuracy loss due to privacy, we show that our algorithm solves the optimization problem correctly. We also discuss privacy characteristics of POLAR-SGD for polynomial objective functions.

3.2 Problem Formulation

We first review the system architecture, distributed learning problem and some preliminaries on distributed learning in client-server models.

3.2.1 System Architecture

We consider a modified parameter server architecture as depicted in Figure 3.1. Our system consists of S parameter servers (also referred to as "servers") and C clients. As discussed in Section 3.1, client i represents a computer that has access to its private data and corresponding loss function $f_i(x)$. Parameter server J maintains a copy of the model x^J, but does not store private data.

The clients receive latest model parameters from the parameter servers and the clients in turn transmit updates computed using local private data to improve the latest model estimate. We also assume that each client communicates with more than one parameter servers in every iteration. The parameter servers communicate with each other every few iterations. We assume that parameter servers form a fully connected graph whenever they wish to exchange models (parameter vector) with each other.

We will assume synchronous and fault-free execution of protocol at all clients and parameter servers. This architecture was analyzed in our prior work in [85] and explored via numerical experiments by Hsieh *et al.* in [34], although [34] does not address privacy as we do.

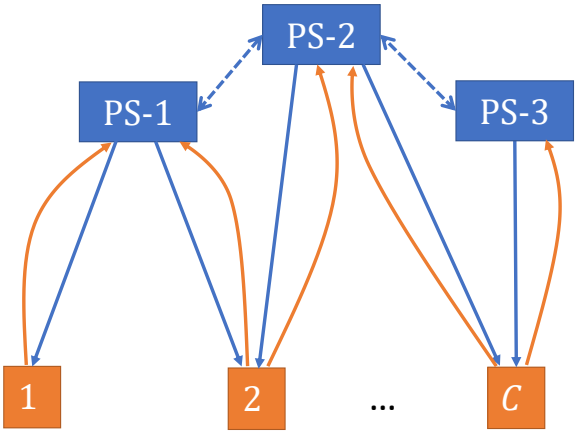


Figure 3.1: Multiple Parameter Server Architecture. Blue squares represent parameter servers and servers are connected in a fully connected graph topology. Orange squares represent clients. Servers behave as coordinating entities and keep track of latest model parameters. Clients store private datasets and store loss functions.

3.2.2 Learning Problem

The central problem here is to build a predictive model based on private local information. We translate this into a distributed optimization problem as discussed below.

The machine learning model is parameterized as a d-dimensional vector $x \in \mathbb{R}^d$. The set of all feasible model parameters is denoted as $\mathcal{X}$, which is a convex, compact subset of $\mathbb{R}^d$ i.e. $x \in \mathcal{X} \subset \mathbb{R}^d$.

We assume that the objective function at client i, denoted as $f_i(x)$ is convex. We also assume that the gradients denoted as $\nabla f_i(x)$ are Lipschitz. We discuss relaxing the Lipschitz continuous assumption later. The learning problem is formally presented below.

Problem 1. *Distributed learning implies finding a minimizer to objective* $f(x) \triangleq \sum_{i=1}^{C} f_i(x)$.

$$\textit{Find } x^* \in \arg\min_{x \in \mathcal{X}} f(x).$$

3.2.3 Privacy Model

Before we discuss the privacy model, we will review the basic non-private learning algorithm for parameter server architecture shown in Figure 3.2. The algorithm involves iteratively running the following steps:

- Clients download latest model parameters x_k from parameter server and compute gradient of local objective function $\nabla f_i(x_k)$.
- Clients upload gradients $\nabla f_i(x_k)$ to the server.
- Server performs projected gradient descent using the sum of all received gradients,

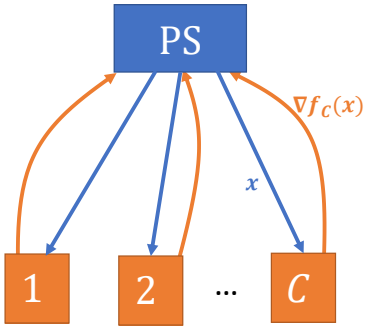


Figure 3.2: System with single parameter server [25].

$$x_{k+1} \leftarrow \mathcal{P}_{\mathcal{X}} \left[x_k - \alpha_k \sum_{i=1}^{C} \nabla f_i(x_k) \right],$$

where α_k is suitably chosen step size and $\mathcal{P}_{\mathcal{X}}$ is the projection operator.

Note that the basic parameter server algorithm discussed above and depicted in Figure 3.2 directly exposes local gradients to the parameter server. If the server were an honest but curious adversary, the server could use these gradients to infer membership of specific data points in the client's private local datasets. In fact, observing even a part of gradient can potentially leak membership information about specific data points or a class of data points that are used for training.

In this work we will assume that multiple parameter servers are available and at most one of the parameter servers is an honest-but-curious adversary (this can be easily generalized to multiple honest but curious adversaries). The parameter servers are often controlled by corporations that provide machine learning as a service that want to learn the best possible machine learning model. However, they may be interested in uncovering private individual data. Our algorithm will protect private data against such honest-but-curious parameter server(s).

Notation: Time is indexed as as $\{i, k\}$. The first index i denotes the number of gradient based descent steps performed since last consensus and the second index refers to the number of consensus operations. We consider consensus to happen after every Δ gradient-based updates. Let $\mathcal{N}_h$ denote the servers that can communicate with client h.

3.3 POLAR-SGD Algorithm

We reviewed the basic distributed optimization algorithm for parameter server framework [25]. It can be described as iterative projected gradient descent algorithm where the gradients are collected at a central parameter server. In what follows, we present POLAR-SGD for distributed optimization for multiple parameter server architecture shown in Figure 3.1.

3.3.1 Algorithm

We first present a sketch of the POLAR-SGD algorithm.

Algorithm Sketch:

1. In the POLAR-SGD algorithm, clients download latest model parameters from one or more servers and compute stochastic gradients at these values.
2. Clients then obfuscate the stochastic gradients using multiplicative and additive perturbations (detailed later in the section). These obfuscated stochastic gradients are uploaded to the server(s). A client may communicate with any subset of available servers.
3. Each server uses the received stochastic gradients to perform a projected gradient descent update of model parameters stored at the server.
4. Servers periodically perform a secure consensus iteration over the model parameters.

POLAR-SGD algorithms for Clients and Servers are presented as Algorithms 3 and 4 respectively.

POLAR-SGD Client: At each iteration $\{i, k\}$, the client first requests the latest model parameters, $x^J_{i,k}$ from parameter servers J (Line 5, Algorithm 3). Clients receive model parameters and use them for gradient computation.

We define three variants of POLAR-SGD based on the model parameters used for stochastic gradient computation (Line 6, Algorithm 3).

1. Minimum-wait: The model parameters that are received first are used by the client (Line 7, Algorithm 3), i.e. if model parameters from server I are received first by client h then,

$$u^{J,h}_{i,k} = x^I_{i,k}, \text{ for each } J \in \mathcal{N}_h. \tag{3.1}$$

   Note that I may be different for different clients and may be dependent on iteration $\{i, k\}$. However, we abuse the notation and drop the dependence of I on client h and time $\{i, k\}$. As we select the parameters that arrive first we call this algorithm *minimum-wait* variant of POLAR-SGD. This variant is robust to delays in few communication links.

2. Client-averaged: All received model parameters are averaged and used for gradient computation (Line 8, Algorithm 3),

$$u^{J,h}_{i,k} = \frac{1}{\mathcal{N}_h} \sum_{I \in \mathcal{N}_h} x^I_{i,k} \text{ for each } J \in \mathcal{N}_h. \tag{3.2}$$

Algorithm 3 POLAR-SGD: Client h

1: Input: $x_{i,k}^J$, Δ, NSteps
2: Result: Upload Obfuscated Gradient - $g_{i,k}^{J,h}$
3: **for** $k = 1$ to NSteps **do**
4: **for** i = 0 to Δ-1 **do**
5: Download $x_{i,k}^J$ from servers $J \in \mathcal{N}_h$
6: Compute $u_{i,k}^{J,h}$ based on algorithm variant
7: minimum-wait: $u_{i,k}^{J,h} = x_{i,k}^I$ $(I \in \mathcal{N}_h)$ OR
8: client-averaged: $u_{i,k}^{J,h} = \frac{1}{|\mathcal{N}_h|} \sum_{I \in \mathcal{N}_h} x_{i,k}^I$ OR
9: basic: $u_{i,k}^{J,h} = x_{i,k}^J$
10: Compute Stochastic Gradient:
 $g_h(u_{i,k}^{J,h}, \xi_{i,k}^h) = \nabla f_h(u_{i,k}^{J,h}, \xi_{i,k}^h)$
11: Select Perturbations for each server $J \in \mathcal{N}_h$:
 $W_{i,k}^{J,h}$, $d_{i,k}^{J,h}$
12: Compute Obfuscated Stochastic Gradient ($g_{i,k}^{J,h}$):
 $g_{i,k}^{J,h} = \left(W_{i,k}[J,h] g_h(u_{i,k}^{J,h}, \xi_{i,k}^h) + d_{i,k}^{J,h}\right)$
13: Upload Obfuscated Stochastic Gradient to Servers $J \in \mathcal{N}_h$:
 $g_{i,k}^{J,h}$
14: **end for**
15: **end for**

Use of average parameter ensures that we only compute one stochastic gradient per iteration (similar to minimum-wait), however, this requires complete download of parameters from several servers, i.e. a client must wait for the slowest server.

3. Basic: In this variant, each received model parameter is used by the client (Line 9, Algorithm 3), i.e.

$$u_{i,k}^{J,h} = x_{i,k}^J. \tag{3.3}$$

Clients compute one stochastic gradient per downloaded model parameter. This results in larger computational cost for clients and hence is not preferred. This is also not robust to delays in communication links as the client needs to wait for the slowest server.

Next, each client h computes stochastic gradient using its private local objective function at the parameter value u (Line 9, Algorithm 3). Clients use a standard technique to compute stochastic gradients. Each client divides its local dataset into several batches and at each iteration picks a batch uniformly at random. Gradient of the loss function is then computed over this selected batch. We denote this stochastic gradient as $g_h(u_{i,k}^{J,h}, \xi_{i,k}^h)$, where $\xi_{i,k}^h$ represents the randomness due to batch selection by a client h at iteration $\{i,k\}$.

Clients then obfuscate the stochastic gradient using additive and multiplicative perturbations. We select these perturbations in the Line 11, Algorithm 3 and use these steps to compute the obfuscated gradient in Line 12, Algorithm 3. The obfuscated gradient uploaded by client h to server J at time $\{i,k\}$, denoted as

$g_{i,k}^{J,h}$, is defined as

$$g_{i,k}^{J,h} = \left(W_{i,k}[J,h] g_h(u_{i,k}^{J,h}, \xi_{i,k}^h) + d_{i,k}^{J,h} \right), \tag{3.4}$$

where $W_{i,k}[J,h]$ is the multiplicative perturbation and $d_{i,k}^{J,h}$ is the additive perturbation assigned by client h to server J at time $\{i,k\}$.

The multiplicative ($W_{i,k}[J,h]$) and additive ($d_{i,k}^{J,h}$) perturbations are arbitrary so long as they satisfy Symmetric Learning and Bounded Update Conditions elaborated below.

Symmetric Learning Condition (SLC) ensures that over a period of Δ gradient descent steps, the multiplicative perturbations assigned by clients are equal and that the additive perturbations sum to zero.

Assumption 1 (SLC)**.** *Equal multiplicative perturbations are assigned to updates from every client over a period of Δ steps. Formally, we have $M > 0$ such that for each client h,*

$$M \triangleq \sum_{J=1}^{S} \left(\sum_{i=0}^{\Delta-1} W_{i,k}[J,h] \right).$$

Also, the additive perturbations add to zero for all $\{i,k\}$ for each client h.

$$\sum_{J=1}^{S} d_{i,k}^{J,h} = 0.$$

Bounded Update Condition (BUC), ensures that the multiplicative and additive perturbations are bounded.

Assumption 2 (BUC)**.** *The sum of absolute value of multiplicative weights over Δ steps is upper bounded by a finite constant ($0 < \overline{M} < \infty$). Formally, for each client h*

$$\sum_{J=1}^{S} \left(\sum_{i=1}^{\Delta} |W_{i,k}[J,h]| \right) \leq \overline{M}.$$

Also, the additive perturbations are bounded by $0 < Y < \infty$.

$$\|d_{i,k}^{J,h}\| \leq Y.$$

Finally, in Line 13 Algorithm 3, each client h uploads the obfuscated stochastic gradient $g_{i,k}^{J,h}$ to server J. This completes the client algorithm for POLAR-SGD.

POLAR-SGD Server: Any parameter server J performs two tasks – (1) using received gradients to perform projected stochastic gradient descent and (2) perform secure consensus over model parameters from all servers.

Algorithm 4 POLAR-SGD: Parameter Server J

1: Input: $x^J_{0,1}$, α_k, Δ, NSteps
2: Result: $x^* = \arg\min_{x\in\mathcal{X}} \sum_{i=1}^{C} f_i(x)$
3: **for** $k = 1$ to NSteps **do**
4: **for** $i = 0$ to Δ-1 **do**
5: $x^J_{i+1,k} = \mathcal{P}_{\mathcal{X}}\left[x^J_{i,k} - \alpha_k \sum_{h=1}^{C} g^{J,h}_{i,k}\right]$
6: **end for**
7: **Secure Average Consensus** performed for each coordinate $p = \{1, \ldots, d\}$:
8: Server J send $R_{J,L}[p] \sim \mathcal{U}[0, 2S|\mathcal{X}[p]|]$ to each neighboring server L
and receives $R_{L,J}[p] \sim \mathcal{U}[0, 2S|\mathcal{X}[p]|]$ from each neighboring server L
9: Compute $A^J[p] = \left(x^J_{\Delta,k}[p] + |\mathcal{X}[p]| + \sum_L R_{L,J}[p] - \sum_L R_{J,L}[p]\right) (mod\ 2S|\mathcal{X}[p]|)$
10: Send $A^J[p]$ to all neighboring servers
11: Compute $x^J_{0,k+1}[p] = \frac{1}{S}\left[\left(\sum_{I=1}^{S} A^I[p]\right) (mod\ 2S|\mathcal{X}[p]|)\right] - |\mathcal{X}[p]|$
12: **end for**

At every iteration $\{i, k\}$, parameter server J uses the obfuscated stochastic gradients $g^{J,h}_{i,k}$ received from clients h at iteration $\{i, k\}$ to perform projected stochastic gradient descent (Line 5, Algorithm 4),

$$x^J_{i+1,k} = \mathcal{P}_{\mathcal{X}}\left(x^J_{i,k} - \alpha_k \sum_{h=1}^{C} g^{J,h}_{i,k}\right), \tag{3.5}$$

where α_k is the step size. We assume that $\{\alpha_k\}$ is monotonically non-increasing, positive sequence with $\sum_k \alpha_k = \infty$ and $\sum_k \alpha_k^2 < \infty$.

The servers perform secure consensus over the model parameters (Lines 8-11, Algorithm 4) after every Δ gradient based updates. The secure multi-party computing based private consensus is perfromed using algorithm in [73] for each coordinate.

Let $p \in \{1, \ldots, d\}$ denote a coordinate. We represent p^{th} coordinates of $x^J_{0,k}$ and $\mathcal{X}$ as $x^J_{0,k}[p]$ and $\mathcal{X}[p]$ respectively. If $x^J_{0,k}[p] \in [a, b]$ as $x^J_{0,k}$ lies in compact set $\mathcal{X}$, then let $|\mathcal{X}[p]|$ represent $\max(|a|, |b|)$. For each coordinate p, we add $|\mathcal{X}[p]|$ to the model parameters $x^J_{0,k}[p]$ to translate the model parameters to non-negative regime. We subtract $|\mathcal{X}[p]|$ in the last step to ensure correctness. Each server J sends uniformly distributed random vectors $R_{J,L}$ to each neighbor server L. Servers then add received random vectors and subtract transmitted random vectors to the translated model parameters followed by a modulo operation. As discussed in [73], the private consensus algorithm uses *wrap-around* feature of modulo arithmetic and uniformly distributed ($\mathcal{U}$) noise to provide information theoretic privacy while computing sum/average over a fully connected graph.

The analysis from [73] proves that observations of A^J from other servers do not leak information about $x^J_{\Delta,k}$. If the servers are connected in a complete network (fully connected graph), then average consensus is reached in one iteration and the inputs are private (information - theoretic privacy) [73].

3.3.2 POLAR-SGD Variants

We introduced minimum-wait, client-averaged and basic variants of POLAR-SGD. As we discussed, minimum-wait is a superior algorithm variant as it is robust to delays by the slowest servers and only requires one stochastic gradient computation in each iteration. The client-averaged variant needs to wait for the servers that are slowest to respond, however, involves only one stochastic gradient computation. The basic variant is the least attractive as it needs to wait for the slowest server and requires multiple stochastic gradient calculations.

3.4 Main Results and Discussion

3.4.1 Correctness

We first show that the iterates of POLAR-SGD converge to the optimum. Theorem 6 states that any variant of POLAR-SGD that uses the perturbations satisfying SLC and BUC (Assumptions 1,2), converges to the optimum asymptotically with probability 1.

Theorem 6 (Correctness)**.** *Let $f_h(x)$ be convex functions and gradients $\nabla f_h(x)$ be bounded and Lipschitz continuous for each h. The iterates generated by any POLAR-SGD variants Minimum-wait, Client-averaged or Basic satisfying SLC and BUC (Assumptions 1, 2) converge to the solution of Problem 1 in $\mathcal{X}^*$ with probability 1.*

The symmetric learning condition ensures that each client assigns the same aggregate weight to its stochastic gradient thereby guaranteeing accuracy of computed solution. The bounded update condition ensures that the obfuscated stochastic gradients are close enough to the exact stochastic gradient. The notion of privacy that we provide is weaker than differential privacy, and as a consequence we circumvent the trade-off between accuracy and privacy. The privacy and correctness are independent and can be achieved simultaneously. However, we do observe a slowdown in the convergence rate.

In this chapter, we only consider differentiable objective functions. However, if the objective functions are non-differentiable, we may still use POLAR-SGD. Our paper [27] presents convergence results for POLAR-SGD where we use stochastic sub-gradients instead. The result states that we can still correctly solve the problem as long as the sub-gradients are bounded and we use either minimum-wait or client-averaged variants of POLAR-SGD with symmetric learning condition satisfied at each $\{i, k\}$. This implies that if $\sum_{J=1}^{S} W_{i,k}[J, h] = M$ and $\sum_{J=1}^{S} |W_{i,k}[J, h]| \leq \overline{M}$ for each client h at any $\{i, k\}$, then the iterates converge to the optimum with probability 1 asymptotically even when $f_h(x)$ are non-differentiable. The proofs are very similar to those in these paper with a required modification to the proof of iterate lemma. We elaborate on this in our paper and its technical report [27, 86].

In this chapter, we focus on performance of algorithm that uses stochastic gradient based updates. If the stochastic gradients are replaced by gradients, computed over entire local dataset by clients, similar correctness and privacy claims hold. We report the results and analysis in technical report [85].

3.4.2 Privacy

The claim asserts that private gradients are private against honest-but-curious adversaries.

Claim 9. *A honest but curious adversary server J can only observe $g_{i,k}^{J,h}$ from any client h.*

Information is observed by adversarial parameter server via two sources – (1) gradient estimate received from client and (2) information exchanged between different servers. Obfuscation via additive and multiplicative perturbations protects the gradients $g_h(x)$ from honest-but-curious parameter servers. Private consensus (adapted from [73]) protects the model parameters from a curious adversary using *wrap-around* feature of modulo arithmetic. The parameters exchanged by servers appear to be uniformly distributed similar to the noise added by agents. The component of gradient received by servers other than J is hence protected against honest-but-curious adversarial servers.

Consider special case where each local objective function $f_h(x)$ is polynomial of bounded degree (p). We can design additive and multiplicative weights (Step 10, Algorithm 3) in order to ensure that obfuscated gradient $g_{i,k}^{J,h}$ appears to be a gradient of a fake polynomial ($\nabla f_h^J(x)$) of bounded degree (p). The symmetric learning condition ensures the obfuscated gradients from a client add to the correct gradient and this permits us to write $\sum_J f_h^J(x) = f_h(x)$. The correctness results for polynomial optimization follow from the analysis above (Theorem 6). The gradient updates appear to be from a fake polynomial, creating an ambiguity and protecting the privacy of local objective functions (Claim 9).

We make an important note about the privacy of aggregate cost function $f(x)$. As we solve the problem correctly, an adversary may learn the exact cost function by observing the progress of the algorithm execution. This is even more apparent when we consider that the secure sum protocol protects the updates received from each client but not the aggregate update. An adversary may monitor the aggregate update to reconstruct $f(x)$. As we saw in Chapter 2, correctness/accuracy preserving mechanisms suffer from this fundamental privacy limitation.

3.5 Analysis and Discussion

We will begin by first restating all the assumptions used in the analysis and proofs. Next, we prove some elementary results and finally present proof for Theorem 6. The proofs will be slightly different for the three variants of POLAR-SGD and we will specifically discuss the differences for each variant. Next, we develop a couple of key lemmas to prove correctness results.

3.5.1 Notation and Assumptions

Let $F_{i,k}$ denote the σ-algebra, $\sigma(\xi^h_{[i,k]}; \forall h)$ generated by the stochastic choice of batches until the time step $\{i,k\}$ (note, $\xi^h_{[i,k]} = (\xi^h_{0,0}, \ldots, \xi^h_{\Delta-1,0}, \xi^h_{0,1}, \ldots, \xi^h_{\Delta-1,1}, \ldots, \xi^h_{0,k-1}, \ldots, \xi^h_{\Delta-1,k-1}, \xi^h_{0,k}, \ldots, \xi^h_{i,k})$). $F_{i,k}$ denotes the history of all stochastic gradients uploaded by clients. Clearly $F_{i,k} \subset F_{j,m}$ whenever $k < m$ or $k = m$ and $i < j$. This follows directly from the construction of $\sigma-$algebra. Observe that the randomness appears from the gradient computation step (batch selection). The multiplicative and additive weights are arbitrary (possibly random) but they *balance-out* due to symmetric learning condition.

We will next review important assumptions required for analysis. Recall that the set of all feasible model parameters, denoted as $\mathcal{X}$, is a convex, compact subset of $\mathbb{R}^d$ (d is the dimension). The local objective functions $f_i(x)$ are convex and the gradients are norm-bounded.

Assumption 3. *Objective functions $f_i : \mathcal{X} \to \mathbb{R}$, are convex functions for all i. Thus, $f(x) \triangleq \sum_{h=1}^{C} f_h(x)$ is also a convex function.*

We consider differentiable objective functions and we make an assumptions on gradients being Lipschitz (Assumption 4).

Assumption 4 (Differentiable Objective Functions)**.** *The differentiable functions $f_i(x)$ satisfy the following conditions,*

1. *The gradients of objective functions are norm-bounded, i.e. $\exists L > 0$, such that, $\|\nabla f_h(x)\| \leq L$, $\forall x \in \mathcal{X}$ and $\forall h$. The stochastic gradients are also norm-bounded, i.e. $\|g_h(x,\xi)\| \leq L$ for all $x \in \mathcal{X}$ and $\forall\ \xi$.*
2. *The gradients of objective functions are Lipschitz, i.e. $\exists N > 0$, such that, $\|\nabla f_h(x) - \nabla f_h(y)\| \leq N\|x - y\|$, $\forall h$ and $\forall x \neq y \in \mathcal{X}$.*

We first state a result on convergence of non-negative almost supermartingales by Robbins and Siegmund (Theorem 1, [75]). This lemma provides a methodology to prove convergence of most distributed optimization algorithms and we exploit this in our work too.

Lemma 10. *Let $(\Omega, F, \mathcal{P})$ be a probability space and let $F_1 \subset F_2 \subset \ldots$ be a sequence of sub σ-fields of F. Let u_k, v_k and w_k, $k = 0, 1, 2, \ldots$ be non-negative F_k-measurable random variables and let $\{\gamma_k\}$ be a deterministic sequence. Assume that $\sum_{k=0}^{\infty} \gamma_k < \infty$ a.s., and $\sum_{k=0}^{\infty} w_k < \infty$ a.s. and*

$$E[u_{k+1}|F_k] \leq (1 + \gamma_k)u_k - v_k + w_k,$$

holds with probability 1. Then, the sequence $\{u_k\}$ converges to a non-negative random variable and $\sum_{k=0}^{\infty} v_k < \infty$ almost surely.

The well-known non-expansive property (cf. [87]) of Euclidean projection onto a non-empty, closed, convex

set $\mathcal{X}$, is represented by the following inequality, $\forall\, x, y \in \mathcal{X}$,

$$\|\mathcal{P}_{\mathcal{X}}[x] - \mathcal{P}_{\mathcal{X}}[y]\| \leq \|x - y\|. \tag{3.6}$$

This non-expansive property of the projection operator is used extensively in our analysis.

Recall that we assumed the parameter servers to form a fully connected network topology in Section 3.2. We need this assumption to use secure-sum protocol for secure averaging among servers similar to [73]. This aids our privacy analysis significantly. However, for convergence we do not need the server graph to be fully connected. We will prove convergence under the weaker assumption of connected server graph. We will assume that at each consensus step, we use convex averaging using a doubly stochastic matrix [8, 88].[1] At every consensus iteration we assume that the local convex averaging protocol can be written as a doubly stochastic matrix $B_{0,k}$. We abuse the notation and drop the first time index, i.e. we denote the matrix $B_{0,k}$ as B_k. We will prove convergence when the consensus is performed over incomplete yet connected server graph. The results hold for scenario when the parameter servers form a fully connected graph.

We borrow transition matrix analaysis result from [82]. We define transition matrix product ($\Phi(k, s)$) as the product of doubly stochastic weight matrices, $\Phi(k, s) = B_k B_{k-1} \dots B_s$, $(\forall\ k \geq s \geq 0)$. We use analysis from [82] to claim linear convergence of all transition matrix entries $\Phi(k, s)[j, i]$ to $1/S$. That is, $\forall\ k \geq s \geq 0$,

$$\left| \Phi(k, s)[i, j] - \frac{1}{S} \right| \leq \theta \beta^{k-s+1},$$

where $\theta = (1 - \frac{\rho}{4S^2})^{-2}$ and $\beta = (1 - \frac{\rho}{4S^2})$ depend only on graph topology. Note, ρ is the smallest nonzero entry in matrix B_k for any k.[2]

3.5.2 Convergence Analysis

Disagreement Lemma

First we construct a bound on the disagreement between the parameters at any server J and the average, denoted by δ_k^J, defined below,

$$\delta_k^J = x_{0,k}^J - \overline{x}_{0,k}. \tag{3.7}$$

[1] In a fully connected topology, the doubly stochastic matrix B_k is just a matrix with all entries being exactly equal to $1/S$. Moreover, consensus happens in one step using the secure sum protocol. The analysis using doubly stochastic weight matrices is not required.

[2] Due to fully connected graph (of parameter servers), at every consensus iteration, the parameter servers compute average of its model parameters. However, this assumption is not strictly required for correctness. The correctness results are correct even if the parameter server graph is merely connected at every consensus iteration. Hence, we will analyze the harder scenario, and assume that the parameter servers form a connected graph. Obviously if the parameter servers have a fully connected topology the correctness results hold.

We get a deterministic bound. We use bounds on the stochastic gradients to compute the bound on $\|\delta_{k+1}^J\|$. The first term shows the geometric decrease of initial disagreement among parameter servers. The other two terms reflect the effect of stochastic gradient based updates.

Lemma 11. *The sequence generated by POLAR-SGD following the symmetric learning and bounded update conditions, satisfies,*

$$\max_J \|\delta_{k+1}^J\| \leq S\theta\beta^{k+1} \max_I \|x_{0,0}^I\| + 4\alpha_k \Delta C(\overline{M}L + Y) + 2\theta\Delta SC(\overline{M}L + Y) \sum_{l=1}^{k} \beta^{k+1-l}\alpha_{l-1}.$$

The proof is the same for all variants of POLAR-SGD. Since we bound the effect of stochastic updates provided by clients, the disagreement lemma is a deterministic statement for all variants of POLAR-SGD.

Proof. We first begin by defining an auxiliary variable, z_{k+1}^J,

$$\begin{aligned} z_{k+1}^J &= x_{0,k+1}^J - \sum_{I=1}^{S} B_k[J,I] x_{0,k}^I \\ \implies x_{0,k+1}^J &= z_{k+1}^J + \sum_{I=1}^{S} B_k[J,I] x_{0,k}^I. \end{aligned} \tag{3.8}$$

The variable z_k^J tries to capture the effect of updates to the model parameters. We unroll the iterations to write $x_{0,k+1}^J$ as a function of $x_{0,k-1}^J$ and z_k^I,

$$x_{0,k+1}^J = z_{k+1}^J + \sum_{I=1}^{S} \left[B_k[J,I] \left(z_k^I + \sum_{l=1}^{S} B_{k-1}[I,l] x_{0,k-1}^l \right) \right]. \tag{3.9}$$

We perform the unrolling equation successively and use the definition of transition matrix $\Phi(k,s)$,

$$x_{0,k+1}^J = z_{k+1}^J + \sum_{I=1}^{S} \Phi(k,0)[J,I] x_{0,0}^I + \sum_{l=1}^{k} \left[\sum_{I=1}^{S} \Phi(k,l)[J,I] z_l^I \right]. \tag{3.10}$$

Next we write the expression for iterate average.

$$\begin{aligned} \overline{x}_{0,k+1} &= \frac{1}{S} \sum_{I=1}^{S} x_{0,k+1}^I = \frac{1}{S} \sum_{J=1}^{S} \left[z_{k+1}^J + \sum_{I=1}^{S} B_k[J,I] x_{0,k}^I \right] = \frac{1}{S} \left[\sum_{I=1}^{S} \left(\sum_{J=1}^{S} B_k[J,I] \right) x_{0,k}^I + \sum_{J=1}^{S} z_{k+1}^J \right] \\ &= \overline{x}_{0,k} + \frac{1}{S} \sum_{J=1}^{S} z_{k+1}^J = \overline{x}_{0,0} + \frac{1}{S} \sum_{l=1}^{k+1} \sum_{J=1}^{S} z_l^J. \end{aligned} \tag{3.11}$$

We use expression for $x_{0,k+1}^J$ (3.10) and $\overline{x}_{0,k+1}$ (3.11) and linear convergence of transition matrix to $1/S$

using transition matrix analysis from [82], to get,

$$\begin{aligned}\|x^J_{0,k+1} - \overline{x}_{0,k+1}\| &= \| \left[\sum_{I=1}^{S} \Phi(k,0)[J,I]x^I_{0,0} - \overline{x}_{0,0}\right] + \left[\sum_{l=1}^{k}\left[\sum_{I=1}^{S}\Phi(k,l)[J,I]z^I_l\right] - \frac{1}{S}\sum_{l=1}^{k+1}\sum_{I=1}^{S} z^I_l\right] + z^J_{k+1}\| \\ &\leq \sum_{I=1}^{S}\left|\Phi(k,0)[J,I] - \frac{1}{S}\right|\|x^I_{0,0}\| + \sum_{l=1}^{k}\sum_{J=1}^{S}\left|\Phi(k,l)[J,I] - \frac{1}{S}\right|\|z^I_l\| + \|z^J_{k+1}\| + \frac{1}{S}\sum_{I=1}^{S}\|z^I_{k+1}\| \\ &\leq S\theta\beta^{k+1}\max_I \|x^I_{0,0}\| + \theta\sum_{l=1}^{k}\beta^{k+1-l}\sum_{I=1}^{S}\|z^I_l\| + \|z^J_{k+1}\| + \frac{1}{S}\sum_{I=1}^{S}\|z^I_{k+1}\|. \end{aligned} \tag{3.12}$$

Next we bound the norm of auxiliary variable $\|z^J_{k+1}\|$. Recall that $x^J_{0,k+1} = \sum_{I=1}^{S} B_k[J,I]x^I_{\Delta,k}$ and we have

$$\|z^J_{k+1}\| = \|x^J_{0,k+1} - \sum_{I=1}^{S} B_k[J,I]x^I_{0,k}\| = \|\sum_{I=1}^{S} B_k[J,I]x^I_{\Delta,k} - \sum_{I=1}^{S} B_k[J,I]x^I_{0,k}\| \leq \max_I \|x^I_{\Delta,k} - \overline{x}_{0,k}\|.$$

We use projected gradient descent update equation to get a bound. Observe that the following expression would differ for variants of POLAR-SGD. However, as seen in (3.16), we use the bound on stochastic gradient to compute the bound on auxiliary variable z^j_k as follows,

$$\begin{aligned} x^I_{\Delta,k} - x^I_{0,k} &= \sum_{t=0}^{\Delta-1}(x^I_{t+1,k} - x^I_{t,k}) \\ &\overset{(a)}{=} \sum_{t=0}^{\Delta-1}\left(e^I_t - \alpha_k\sum_{h=1}^{C} g^{I,h}_{t,k}\right) \\ &= -\alpha_k\sum_{t=0}^{\Delta-1}\left[\sum_{h=1}^{C} g^{I,h}_{t,k}\right] + \sum_{t=0}^{\Delta-1} e^I_t, \end{aligned} \tag{3.13}$$

where (a) follows from the definition of the projection error, e^I_t, defined as the correction introduced by the projection operator $e^I_t = x^I_{t+1,k} - \left(x^I_{t,k} - \alpha_k\sum_{h=1}^{C} g^{I,h}_{t,k}\right)$. We first construct a bound on the gradient based update as follows

$$\begin{aligned} \|\sum_{h=1}^{C} g^{I,h}_{t,k}\| &\leq \sum_{h=1}^{C}\|g^{I,h}_{t,k}\| \\ &\leq \sum_{h=1}^{C}\|W_{t,k}[I,h]g(u^{I,h}_{t,k},\xi^h_{t,k}) + d^{I,h}_{t,k}\| \\ &\leq C\max_h\{\|W_{t,k}[I,h]g(u^{I,h}_{t,k},\xi^h_{t,k}) + d^{I,h}_{t,k}\|\} \\ &\overset{(a)}{\leq} C\left(\overline{M}L + Y\right), \end{aligned} \tag{3.14}$$

where (a) follows from boundedness of perturbations $|W_{t,k}[J,h]| \leq \overline{M}$, and $\|d^{I,h}_{t,k}\| \leq Y$ and gradient boundedness $\|g(u^{J,h}_{t,k},\xi^h_{t,k})\| \leq L$ (Assumption 4). Since both $x^I_{t+1,k}$ and $x^I_{t,k}$ belong in $\mathcal{X}$, the projection error is

bounded by the gradient based update, i.e.

$$\|e_t^I\| \leq \alpha_k \|\sum_{h=1}^{C} g_{t,k}^{I,h}\| \overset{(a)}{\leq} \alpha_k C\left(\overline{M}L + Y\right), \tag{3.15}$$

where (a) follows from bound on gradient update as expressed in (3.14). Using this bound on projection error we get,

$$\begin{aligned}
\|z_{k+1}^J\| &\leq \max_I \|x_{\Delta,k}^I - \overline{x}_{0,k}\| \leq \| - \alpha_k \sum_{t=0}^{\Delta-1} \left[\sum_{h=1}^{C} g_{t,k}^{I,h}\right] + \sum_{t=0}^{\Delta-1} e_t^I\| \\
&\leq \| - \alpha_k \sum_{t=0}^{\Delta-1} \left[\sum_{h=1}^{C} g_{t,k}^{I,h}\right] \| + \| \sum_{t=0}^{\Delta-1} e_t^I\| \\
&\overset{(a)}{\leq} \| - \alpha_k \sum_{t=0}^{\Delta-1} \left[\sum_{h=1}^{C} g_{t,k}^{I,h}\right] \| + \sum_{t=0}^{\Delta-1} \alpha_k C\left(\overline{M}L + Y\right) \\
&\leq \alpha_k \sum_{t=0}^{\Delta-1} \| \left[\sum_{h=1}^{C} g_{t,k}^{I,h}\right] \| + \alpha_k \Delta C(\overline{M}L + Y) \\
&\overset{(b)}{\leq} \alpha_k \sum_{t=0}^{\Delta-1} C(\overline{M}L + Y) + \alpha_k \Delta C(\overline{M}L + Y) \leq 2\alpha_k \Delta C(\overline{M}L + Y),
\end{aligned} \tag{3.16}$$

where (a) follows from (3.15) and (b) follows from (3.14).
Now we use (3.16) to rewrite (3.12),

$$\begin{aligned}
\|x_{0,k+1}^J - \overline{x}_{0,k+1}\| &\leq S\theta\beta^{k+1} \max_I \|x_{0,0}^I\| + \theta \sum_{l=1}^{k} \beta^{k+1-l} \sum_{I=1}^{S} \|z_l^I\| + \|z_{k+1}^J\| + \frac{1}{S}\sum_{I=1}^{S} \|z_{k+1}^I\| \\
&\leq S\theta\beta^{k+1} \max_I \|x_{0,0}^I\| + \theta \sum_{l=1}^{k} \beta^{k+1-l} \sum_{I=1}^{S} 2\alpha_{l-1}\Delta C(\overline{M}L + Y) + 2\alpha_k \Delta C(\overline{M}L + Y) \\
&\quad + \frac{1}{S}\sum_{I=1}^{S} 2\alpha_k \Delta C(\overline{M}L + Y) \\
&\leq S\theta\beta^{k+1} \max_I \|x_{0,0}^I\| + 2\theta S \Delta C(\overline{M}L + Y) \sum_{l=1}^{k} \beta^{k+1-l}\alpha_{l-1} + 4\alpha_k \Delta C(\overline{M}L + Y).
\end{aligned}$$

Finally we use $\max_J \|\delta_{k+1}^J\| = \max_J \|x_{0,k+1}^J - \overline{x}_{0,k+1}\|$ and above expression to prove,

$$\max_J \|\delta_{k+1}^J\| \leq S\theta\beta^{k+1} \max_I \|x_{0,0}^I\| + 2\theta S \Delta C(\overline{M}L + Y) \sum_{l=1}^{k} \beta^{k+1-l}\alpha_{l-1} + 4\alpha_k \Delta C(\overline{M}L + Y).$$

□

Note that the disagreement lemma expression is relevant because we assume a connected graph for servers (not fully connected) for proving convergence. If we have a fully connected graph then, we will have exact average at every consensus step and the disagreement would be exactly zero at every iteration.

Iterate Lemma

The lemma below provides a bound on the distance between the iterates and a point $y \in \mathcal{X}$.

Lemma 12. *Let $\eta_k^2 \triangleq \sum_{J=1}^S \|x_{0,k}^J - y\|^2$ for $y \in \mathcal{X}$ and $\delta_k^J = x_{0,k}^J - \overline{x}_{0,k}$. The sequence generated by POLAR-SGD following symmetric learning and bounded update conditions, satisfies for all $y \in \mathcal{X}$,*

$$\mathbb{E}[\eta_{k+1}^2 | F_{0,k}] = (1 + A_k)\eta_k^2 - 2M\alpha_k(f(\overline{x}_{0,k}) - f(y)) + B_k,$$

$A_k = 4\frac{1}{S}\alpha_k^2 N\Delta C^2 \overline{M}(\overline{M}L + Y) + 2\frac{1}{S}\alpha_k NC\overline{M}\max_J \|\delta_k^J\|$, *and* $B_k = \alpha_k^2\left(C_1^2 + 4N\Delta C^2\overline{M}(\overline{M}L + Y)\right) + 2\alpha_k NC\overline{M}\max_J\|\delta_k^J\| + 2\alpha_k C(\overline{M}L + \Delta Y)\max_J\|\delta_k^J\|$.

The iterate lemma provides a dissipation inequality that has similar structure in Lemma 10. The proof of Lemma 12 differs slightly for each POLAR-SGD variants and we discuss it in the following proof.

Proof. First we define the distance between an iterates and a point $y \in \mathcal{X}$ as follows,

$$\begin{aligned}
\eta_{k+1}^2 &\triangleq \sum_{J=1}^S \|x_{0,k+1}^J - y\|^2 \\
&= \sum_{J=1}^S \left\| \sum_{I=1}^S B_k[J,I] x_{\Delta,k}^I - y \right\|^2 \\
&\overset{(a)}{=} \sum_{J=1}^S \left\| \sum_{I=1}^S B_k[J,I]\left(x_{\Delta,k}^I - y\right) \right\|^2 \\
&\overset{(b)}{\leq} \sum_{J=1}^S \sum_{I=1}^S B_k[J,I] \| \left(x_{\Delta,k}^I - y\right) \|^2 \\
&\overset{(c)}{\leq} \sum_{J=1}^S \|x_{\Delta,k}^J - y\|^2,
\end{aligned}$$

where (a) and (c) follow from doubly stochastic nature of B_k and (b) follows non-negativity of matrix B_k. Next we use the projected gradient descent update equation to rewrite the above expression in terms of iterates $x_{\Delta-1,k}^J$. This is followed by unrolling the expression further to include prior iterates, i.e.,

$$\begin{aligned}
\eta_{k+1}^2 &\overset{(a)}{\leq} \sum_{J=1}^S \|x_{\Delta,k}^J - y\|^2 \\
&\overset{(b)}{=} \sum_{J=1}^S \|\mathcal{P}_{\mathcal{X}}\left[x_{\Delta-1,k}^J - \alpha_k \sum_{h=1}^C g_{\Delta-1,k}^{J,h}\right] - y\|^2 \\
&\overset{(c)}{\leq} \sum_{J=1}^S \|x_{\Delta-1,k}^J - \alpha_k \sum_{h=1}^C g_{\Delta-1,k}^{J,h} - y\|^2 \\
&\leq \sum_{J=1}^S \left[\|x_{\Delta-1,k}^J - y\|^2 + \alpha_k^2 \left[\| \sum_{h=1}^C g_{\Delta-1,k}^{J,h} \|^2 \right] - 2\alpha_k \left[\sum_{h=1}^C g_{\Delta-1,k}^{J,h} \right]^T \left[x_{\Delta-1,k}^J - y \right] \right]
\end{aligned}$$

$$\overset{(d)}{\leq} \sum_{J=1}^{S}\left[\|x_{0,k}^{J}-y\|^2+\alpha_k^2\sum_{t=0}^{\Delta-1}\left[\|\sum_{h=1}^{C}g_{t,k}^{J,h}\|^2\right]-2\alpha_k\sum_{t=0}^{\Delta-1}\left[\left[\sum_{h=1}^{C}g_{t,k}^{J,h}\right]^T\left[x_{t,k}^{J}-y\right]\right]\right]$$

$$\leq \eta_k^2+\alpha_k^2\sum_{J=1}^{S}\sum_{t=0}^{\Delta-1}\left[\sum_{h=1}^{C}\|g_{t,k}^{J,h}\|^2\right]-2\alpha_k\sum_{J=1}^{S}\sum_{t=0}^{\Delta-1}\left[\left[\sum_{h=1}^{C}g_{t,k}^{J,h}\right]^T\left[x_{t,k}^{J}-y\right]\right], \tag{3.17}$$

where (a) follows from the doubly stochastic matrix B_k, (b) follows from the projected sub-gradient descent step, (c) follows from non-expansive property of projection operator and $g_{t,k}^{J,h}=W_{t,k}[J,h]g(u_{t,k}^{J,h},\xi_{t,k}^{h})+d_{t,k}^{J,h}$, and (d) follows from the unrolling $\|x_{\Delta-1,k}^{J}-y\|^2$ over iterations $\{\Delta-2,k\},\{\Delta-3,k\},\ldots,\{0,k\}$.

Next we rewrite $x_{t,k}^{J}-y$ as $x_{t,k}^{J}-y=x_{0,k}^{J}-y-\alpha_k\sum_{i=0}^{t-1}\sum_{h=1}^{C}g_{i,k}^{J,h}+\sum_{i=0}^{t-1}e_i^{J}$ following the analysis from (3.13). We plug this expression into the above expression (3.17) to get,

$$\eta_{k+1}^2\leq \eta_k^2+\alpha_k^2\sum_{J=1}^{S}\sum_{t=0}^{\Delta-1}\left[\sum_{h=1}^{C}\|g_{t,k}^{J,h}\|^2\right]-2\alpha_k\sum_{J=1}^{S}\sum_{t=0}^{\Delta-1}\left[\left[\sum_{h=1}^{C}g_{t,k}^{J,h}\right]^T\left[x_{0,k}^{J}-y-\alpha_k\sum_{i=0}^{t-1}\sum_{h=1}^{C}g_{i,k}^{J,h}+\sum_{i=0}^{t-1}e_i^{J}\right]\right]$$

$$\leq \eta_k^2+\alpha_k^2\sum_{J=1}^{S}\sum_{t=0}^{\Delta-1}\left[\sum_{h=1}^{C}\|g_{t,k}^{J,h}\|^2\right]-2\alpha_k\sum_{J=1}^{S}\sum_{t=0}^{\Delta-1}\left[\left[\sum_{h=1}^{C}g_{t,k}^{J,h}\right]^T\left[x_{0,k}^{J}-y\right]\right]$$

$$\underbrace{+2\alpha_k\sum_{J=1}^{S}\sum_{t=1}^{\Delta-1}\left\|\left\|\sum_{h=1}^{C}g_{t,k}^{J,h}\right\|\right\|-\alpha_k\sum_{i=0}^{t-1}\sum_{h=1}^{C}g_{i,k}^{J,h}+\sum_{i=0}^{t-1}e_i^{J}\Bigg\|}_{\Pi}. \tag{3.18}$$

We will first simplify Π,

$$\Pi=2\alpha_k\sum_{J=1}^{S}\sum_{t=0}^{\Delta-1}\left\|\left\|\sum_{h=1}^{C}g_{t,k}^{J,h}\right\|\right\|-\alpha_k\sum_{i=0}^{t-1}\sum_{h=1}^{C}g_{i,k}^{J,h}+\sum_{i=0}^{t-1}e_i^{J}\Bigg\|$$

$$\overset{(a)}{\leq} 2\alpha_k\sum_{J=1}^{S}\sum_{t=0}^{\Delta-1}\left\|\sum_{h=1}^{C}g_{t,k}^{J,h}\right\|\left(2\alpha_k\Delta C(\overline{M}L+Y)\right)$$

$$\leq 4\alpha_k^2\Delta C(\overline{M}L+Y)\sum_{J=1}^{S}\sum_{t=0}^{\Delta-1}\left\|\sum_{h=1}^{C}g_{t,k}^{J,h}\right\|\leq 4\alpha_k^2\Delta C(\overline{M}L+Y)\sum_{J=1}^{S}\sum_{t=0}^{\Delta-1}\left\|\sum_{h=1}^{C}g_{t,k}^{J,h}\right\|$$

$$\overset{(b)}{\leq} 4\alpha_k^2\Delta C(\overline{M}L+Y)\sum_{J=1}^{S}\sum_{t=0}^{\Delta-1}C(\overline{M}L+Y)$$

$$\leq 4\alpha_k^2S\Delta^2C^2(\overline{M}L+Y)^2, \tag{3.19}$$

where (a) follows from (3.14) and (3.15) and (b) follows from the gradient bound (3.14).

We plug the bound on Π found in (3.19) in (3.18),

$$\eta_{k+1}^2\leq \eta_k^2+\alpha_k^2\sum_{J=1}^{S}\sum_{t=0}^{\Delta-1}\left[\sum_{h=1}^{C}\|g_{t,k}^{J,h}\|^2\right]-2\alpha_k\sum_{J=1}^{S}\sum_{t=0}^{\Delta-1}\left[\left[\sum_{h=1}^{C}g_{t,k}^{J,h}\right]^T\left[x_{0,k}^{J}-y\right]\right]+4\alpha_k^2S\Delta^2C^2(\overline{M}L+Y)^2.$$

Here, we use boundedness assumption on the stochastic gradient (sub-gradient) and both SLC and BUC

(Assumptions 1, 2, and 4). Hence the expression is valid for all variants of POLAR-SGD.

Recall the definition of σ-algebra generated by random choice of batches used for stochastic gradient computation denoted by $F_{0,k}$. Note $\mathbb{E}[\eta_k^2|F_{0,k}] = \eta_k^2$ follows from definition of $F_{0,k}$. We take expectation of both sides of the expression conditioned on $F_{0,k}$,

$$\begin{aligned}\mathbb{E}[\eta_{k+1}^2|F_{0,k}] &\leq \eta_k^2 + \alpha_k^2 \sum_{J=1}^{S}\sum_{t=0}^{\Delta-1}\left[\sum_{h=1}^{C}\mathbb{E}[\|g_{t,k}^{J,h}\|^2|F_{0,k}]\right] - 2\alpha_k \sum_{J=1}^{S}\sum_{t=0}^{\Delta-1}\left(\mathbb{E}\left[\left[\sum_{h=1}^{C} g_{t,k}^{J,h}\right]^T \left[x_{0,k}^J - y\right]|F_{0,k}\right]\right) \\ &\quad + 4\alpha_k^2 S\Delta^2 C^2(\overline{M}L + Y)^2 \\ &\leq \eta_k^2 + \alpha_k^2 C_1^2 - 2\alpha_k \sum_{J=1}^{S}\sum_{t=0}^{\Delta-1}\left(\mathbb{E}\left[\left[\sum_{h=1}^{C} g_{t,k}^{J,h}\right]^T \left[x_{0,k}^J - y\right]|F_{0,k}\right]\right), \end{aligned} \tag{3.20}$$

where $C_1^2 = \sum_{J=1}^{S}\sum_{t=0}^{\Delta-1}\left[\sum_{h=1}^{C}\mathbb{E}[\|g_{t,k}^{J,h}\|^2|F_{0,k}]\right] + 4S\Delta^2C^2(\overline{M}L+Y)^2 < \infty$ follows from boundedness of stochastic gradients and square summability property of step-size α_k. Note that clients sample batches uniformly for computing stochastic gradient. As ξ is drawn from uniform distribution,

$$\mathbb{E}[g_{t,k}^{J,h}|F_{0,k}] = \mathbb{E}[W_{t,k}[J,h]g(u_{t,k}^{J,h},\xi_{t,k}^h) + d_{t,k}^{J,h}|F_{0,k}] = W_{t,k}[J,h]\nabla f_h(u_{t,k}^{J,h}) + d_{t,k}^{J,h}, \tag{3.21}$$

where $u = u_{t,k}^h$ is the downloaded model by client h. Note that the above equality is valid for all variants of POLAR-SGD: (a) basic, (b) client-averaged and (c) minimum-wait. Substituting equality (3.21) in expression (3.20) we get,

$$\mathbb{E}[\eta_{k+1}^2|F_{0,k}] \leq \eta_k^2 + \alpha_k^2 C_1^2 \underbrace{-2\alpha_k \sum_{J=1}^{S}\sum_{t=0}^{\Delta-1}\left(\left[\sum_{h=1}^{C} W_{t,k}[J,h]\nabla f_h(u_{t,k}^{J,h}) + d_{t,k}^{J,h}\right]^T \left[x_{0,k}^J - y\right]\right)}_{\Lambda}. \tag{3.22}$$

Next we estimate a bound on Λ defined in (3.22),

$$\begin{aligned}\Lambda &= -2\alpha_k \sum_{J=1}^{S}\sum_{t=0}^{\Delta-1}\left(\left[\sum_{h=1}^{C} W_{t,k}[J,h]\nabla f_h(u_{t,k}^{J,h}) + d_{t,k}^{J,h}\right]^T \left[x_{0,k}^J - y\right]\right) \\ &= -2\alpha_k \sum_{J=1}^{S}\sum_{t=0}^{\Delta-1}\left(\left[\sum_{h=1}^{C} W_{t,k}[J,h]\nabla f_h(u_{t,k}^{J,h}) + d_{t,k}^{J,h}\right]^T \left[\overline{x}_{0,k} + \delta_k^J - y\right]\right) \\ &= -2\alpha_k \sum_{J=1}^{S}\sum_{t=0}^{\Delta-1}\left(\left[\sum_{h=1}^{C} W_{t,k}[J,h]\nabla f_h(u_{t,k}^{J,h}) + d_{t,k}^{J,h}\right]^T \left[\overline{x}_{0,k} - y\right]\right) \\ &\quad - 2\alpha_k \sum_{J=1}^{S}\sum_{t=0}^{\Delta-1}\left(\left[\sum_{h=1}^{C} W_{t,k}[J,h]\nabla f_h(u_{t,k}^{J,h}) + d_{t,k}^{J,h}\right]^T \left[\delta_k^J\right]\right), \end{aligned} \tag{3.23}$$

where recall $\delta_k^J = x_{0,k}^J - \overline{x}_{0,k}$. Observe that since u is different depending upon the POLAR-SGD variant.

And hence the proof differs for different variants of the algorithm. However we will first discuss the differences and then provide a unified proof for the claims. We will first use gradient Lipschitzness (Assumption 4) to establish the following gradient approximations and use them in the proof. Since the same quantity can be used to upper bound the gradient approximation error, $l_{t,k}^{J,h} = \nabla f_h(u_{t,k}^{J,h}) - \nabla f_h(\overline{x}_{0,k})$, the proof follows a similar structure and the same super-martingale expression is valid for all three variants of POLAR-SGD.

Case 1: Basic variant: For the basic variant where $u = x_{t,k}^J$. We assume that the gradients are Lipschitz, and hence we can write, $\nabla f_h(x_{t,k}^J) = \nabla f_h(\overline{x}_{0,k}) + l_{t,k}^{J,h}$, where $\|l_{t,k}^{J,h}\| \leq N\|x_{t,k}^J - \overline{x}_{0,k}\|$. Next we define $\|l_{0,k}^h\| = \max_{J,t} \|l_{t,k}^{J,h}\|$ and construct a bound on it,

$$\begin{aligned}
\|l_{0,k}^h\| &= \max_{J,t} \|l_{t,k}^{J,h}\| = \max_{J,t} \|\nabla f_h(x_{t,k}^J) - \nabla f_h(\overline{x}_{0,k})\| \\
&\overset{(a)}{\leq} N \max_{J,t} \|x_{t,k}^J - \overline{x}_{0,k}\| \\
&\overset{(b)}{\leq} N \max_{J,t} \|x_{t,k}^J - x_{0,k}^J\| + N \max_{J} \|x_{0,k}^J - \overline{x}_{0,k}\| \\
&\overset{(c)}{\leq} 2\alpha_k N \Delta C(\overline{M}L + Y) + N \max_{J} \|\delta_k^J\|, \qquad (3.24)
\end{aligned}$$

where (a) follows from Lipschitzness of $\nabla f_h(x)$, (b) follows Triangle inequality and (c) follows from (3.16). Finally,

$$\|l_{0,k}^h\| \leq 2\alpha_k N \Delta C(\overline{M}L + Y) + N \max_{J} \|\delta_k^J\|. \qquad (3.25)$$

Case 2: Minimum-wait variant: For the minimum-wait variant where $u_{t,k}^{J,h} = x_{t,k}^I$ for some $I \in \mathcal{N}_h$. We assume that the gradients are Lipschitz, and hence we can write, $\nabla f_h(x_{t,k}^I) = \nabla f_h(\overline{x}_{0,k}) + l_{t,k}^{I,h}$, where $\|l_{t,k}^{I,h}\| \leq N\|x_{t,k}^I - \overline{x}_{0,k}\|$. Next we define $\|l_{0,k}^h\| = \max_{I,t} \|l_{t,k}^{I,h}\|$ and construct a bound on it,

$$\begin{aligned}
\|l_{0,k}^h\| &= \max_{I,t} \|l_{t,k}^{I,h}\| = \max_{I,t} \|\nabla f_h(x_{t,k}^I) - \nabla f_h(\overline{x}_{0,k})\| \\
&\overset{(a)}{\leq} N \max_{I,t} \|x_{t,k}^I - \overline{x}_{0,k}\| \\
&\overset{(b)}{\leq} N \max_{I,t} \|x_{t,k}^I - x_{0,k}^I\| + N \max_{I} \|x_{0,k}^I - \overline{x}_{0,k}\| \\
&\leq 2\alpha_k N \Delta C(\overline{M}L + Y) + N \max_{I,t} \|\delta_k^I\| \\
\|l_{0,k}^h\| &\leq 2\alpha_k N \Delta C(\overline{M}L + Y) + N \max_{I} \|\delta_k^I\|, \qquad (3.26)
\end{aligned}$$

where (a) follows from Lipschitzness of $\nabla f_h(x)$ and (b) follows from Triangle inequality.

Case 3: Client-averaged variant: For the client-averaged variant where $u_{i,k}^{J,h} = \frac{1}{|\mathcal{N}_h|} \sum_{I \in \mathcal{N}_h} x_{t,k}^I$. We

assume that the gradients are Lipschitz, and hence we can write, $\nabla f_h(u) = \nabla f_h(\overline{x}_{0,k}) + l^h_{t,k}$, where $\|l^h_{t,k}\| \leq N\|u - \overline{x}_{0,k}\|$. Next we define $\|l^h_{0,k}\| = \max_t \|l^h_{t,k}\|$ and construct a bound on it,

$$\begin{aligned}
\|l^h_{0,k}\| &= \max_t \|\nabla f_h(u) - \nabla f_h(\overline{x}_{0,k})\| \\
&\overset{(a)}{\leq} N \max_t \|u - \overline{x}_{0,k}\| \\
&\leq N \max_t \|\frac{1}{|\mathcal{N}_h|} \sum_{I \in \mathcal{N}_h} \left(x^I_{t,k} - x^I_{0,k}\right) \| + N \|\frac{1}{|\mathcal{N}_h|} \sum_{I \in \mathcal{N}_h} \left(x^I_{0,k} - \overline{x}_{0,k}\right) \| \\
&\leq N \max_t \max_I \|x^I_{t,k} - x^I_{0,k}\| + N \max_I \|x^I_{0,k} - \overline{x}_{0,k}\| \\
&\leq 2\alpha_k N \Delta C(\overline{M}L + Y) + N \max_I \|\delta^I_k\| \\
\|l^h_{0,k}\| &\leq 2\alpha_k N \Delta C(\overline{M}L + Y) + N \max_I \|\delta^I_k\|, \qquad (3.27)
\end{aligned}$$

where (a) follows from Lipschitzness of $\nabla f_h(x)$.

Observe that the bound on $\|l^{J,h}_{t,k}\|$ is the same for each POLAR-SGD variants. Next we will use the bound on $\|l^h_{0,k}\|$ developed in (3.25), (3.26) and (3.27) to rewrite the expression for Λ in (3.23) as follows,

$$\begin{aligned}
\Lambda &= 2\alpha_k \sum_{J=1}^{S} \sum_{t=0}^{\Delta-1} \left(\left[\sum_{h=1}^{C} W_{t,k}[J,h] \left(\nabla f_h(\overline{x}_{0,k}) + l^{J,h}_{t,k} \right) + d^{J,h}_{t,k} \right]^T [y - \overline{x}_{0,k}] \right) \\
&\quad - 2\alpha_k \sum_{J=1}^{S} \sum_{t=0}^{\Delta-1} \left(\left[\sum_{h=1}^{C} W_{t,k}[J,h] (\nabla f_h(u^{J,h}_{t,k})) + d^{J,h}_{t,k} \right]^T [\delta^J_k] \right) \\
&= \underbrace{2\alpha_k \sum_{J=1}^{S} \sum_{t=0}^{\Delta-1} \left(\left[\sum_{h=1}^{C} W_{t,k}[J,h] \nabla f_h(\overline{x}_{0,k}) + d^{J,h}_{t,k} \right]^T [y - \overline{x}_{0,k}] \right)}_{T_1} \\
&\quad \underbrace{+ 2\alpha_k \sum_{J=1}^{S} \sum_{t=0}^{\Delta-1} \left(\left[\sum_{h=1}^{C} W_{t,k}[J,h]\ l^{J,h}_{t,k} \right]^T [y - \overline{x}_{0,k}] \right)}_{T_2} \\
&\quad \underbrace{-2\alpha_k \sum_{J=1}^{S} \sum_{t=0}^{\Delta-1} \left(\left[\sum_{h=1}^{C} W_{t,k}[J,h] (\nabla f_h(u^{J,h}_{t,k})) + d^{J,h}_{t,k} \right]^T [\delta^J_k] \right)}_{T_3} \\
&= T_1 + T_2 + T_3.
\end{aligned}$$

We will next construct each of the individual bounds on T_1, T_2 and T_3. We begin with T_1,

$$T_1 = 2\alpha_k \sum_{J=1}^{S} \sum_{t=0}^{\Delta-1} \left(\left[\sum_{h=1}^{C} W_{t,k}[J,h] \nabla f_h(\overline{x}_{0,k}) + d^{J,h}_{t,k} \right]^T [y - \overline{x}_{0,k}] \right)$$

$$
\begin{aligned}
&= 2\alpha_k \left(\sum_{h=1}^{C} \left[\sum_{J=1}^{S} \left(\sum_{t=0}^{\Delta-1} W_{t,k}[J,h] \right) \nabla f_h(\overline{x}_{0,k}) + \sum_{t=0}^{\Delta-1} \left(\sum_{J=1}^{S} d_{t,k}^{J,h} \right) \right]^T \right) [y - \overline{x}_{0,k}] \\
&\overset{(a)}{=} 2\alpha_k \left(\sum_{h=1}^{C} \left[\sum_{J=1}^{S} \left(\sum_{t=0}^{\Delta-1} W_{t,k}[J,h] \right) \nabla f_h(\overline{x}_{0,k}) \right]^T \right) [y - \overline{x}_{0,k}] \\
&\overset{(b)}{=} 2\alpha_k \left(\sum_{h=1}^{C} \left[M \nabla f_h(\overline{x}_{0,k}) \right]^T \right) [y - \overline{x}_{0,k}] \\
&\overset{(c)}{\leq} -2\alpha_k M \left(f(\overline{x}_{0,k}) - f(y) \right), \qquad (3.28)
\end{aligned}
$$

where $(a) - (b)$ follow from symmetric learning conditions of additive and multiplicative perturbations and (c) follows from the fact that $f(x) = \sum_{i=1}^{C} f_h(x)$ is a convex function. Next we bound T_2,

$$
\begin{aligned}
\|T_2\| &= \left\| 2\alpha_k \sum_{J=1}^{S} \sum_{t=0}^{\Delta-1} \left(\left[\sum_{h=1}^{C} W_{t,k}[J,h]\ l_{t,k}^{J,h} \right]^T [y - \overline{x}_{0,k}] \right) \right\| \\
&\leq \left\| 2\alpha_k \sum_{J=1}^{S} \sum_{t=0}^{\Delta-1} \left(\left[\sum_{h=1}^{C} W_{t,k}[J,h]\ l_{t,k}^{J,h} \right] \right) \right\| \|y - \overline{x}_{0,k}\| \\
&\leq 2\alpha_k \left(\left[\sum_{h=1}^{C} \left(\sum_{J=1}^{S} \left(\sum_{t=0}^{\Delta-1} |W_{t,k}[J,h]| \right) \right) \max_{J,t} \|l_{t,k}^{J,h}\| \right] \right) \|y - \overline{x}_{0,k}\| \\
&\overset{(a)}{\leq} 2\alpha_k \left(\left[\sum_{h=1}^{C} \overline{M} \max_{J,t} \|l_{0,k}^{h}\| \right] \right) \|y - \overline{x}_{0,k}\| \\
&\overset{(b)}{\leq} 2\alpha_k \left(C\overline{M} \left[2\alpha_k N \Delta C (\overline{M} L + Y) + N \max_{J} \|\delta_k^J\| \right] \right) \|y - \overline{x}_{0,k}\| \\
&\leq 4\alpha_k^2 N \Delta C^2 \overline{M} (\overline{M} L + Y) \|\overline{x}_{0,k} - y\| + 2\alpha_k N C \overline{M} \max_{J} \|\delta_k^J\| \|\overline{x}_{0,k} - y\|, \qquad (3.29)
\end{aligned}
$$

where (a) follows from bounded update condition of multiplicative perturbations and (b) follows (3.25) and (3.27). Observe that for any positive a, $a^2 + 1 \geq 2a$. Additionally, we use the fact that $\|\overline{x}_{0,k} - y\|^2 = \|\frac{1}{S} \sum_{J=1}^{S} (x_{0,k}^J - y)\|^2 \leq \frac{1}{S} \sum_{J=1}^{S} \|x_{0,k}^J - y\|^2 = \frac{1}{S} \eta_k^2$ to update the bound on T_2,

$$
\begin{aligned}
\|T_2\| &\leq 4\alpha_k^2 N \Delta C^2 \overline{M} (\overline{M} L + Y) \|\overline{x}_{0,k} - y\| + 2\alpha_k N C \overline{M} \max_{J} \|\delta_k^J\| \|\overline{x}_{0,k} - y\| \\
&\leq 4\alpha_k^2 N \Delta C^2 \overline{M} (\overline{M} L + Y)(1 + \|\overline{x}_{0,k} - y\|^2) + 2\alpha_k N C \overline{M} \max_{J} \|\delta_k^J\| (1 + \|\overline{x}_{0,k} - y\|^2) \\
&\leq 4\alpha_k^2 N \Delta C^2 \overline{M} (\overline{M} L + Y)(1 + \frac{1}{S} \eta_k^2) + 2\alpha_k N C \overline{M} \max_{J} \|\delta_k^J\| (1 + \frac{1}{S} \eta_k^2). \qquad (3.30)
\end{aligned}
$$

Finally we construct a bound on T_3,

$$
\|T_3\| = \left\| - 2\alpha_k \sum_{J=1}^{S} \sum_{t=0}^{\Delta-1} \left(\left[\sum_{h=1}^{C} W_{t,k}[J,h] (\nabla f_h(u_{t,k}^{J,h})) + d_{t,k}^{J,h} \right]^T [\delta_k^J] \right) \right\|
$$

$$
\begin{aligned}
&\leq 2\alpha_k \left\| \sum_{h=1}^{C} \left[\left(\sum_{J=1}^{S} \sum_{t=0}^{\Delta-1} |W_{t,k}[J,h]| \right) \max_{h,t} \|\nabla f_h(u_{t,k}^{J,h})\| + \sum_{t=0}^{\Delta-1} \sum_{J=1}^{S} |d_{t,k}^{J,h}| \right]^T \left[\delta_k^J\right] \right\| \\
&\overset{(a)}{\leq} 2\alpha_k \sum_{h=1}^{C} \left\| \left[\overline{M} \max_{h,t} \|\nabla f_h(u_{t,k}^{J,h})\| + \sum_{t=0}^{\Delta-1} \sum_{J=1}^{S} |d_{t,k}^{J,h}| \right] \right\| \left\| \left[\delta_k^J\right] \right\| \\
&\overset{(b)}{\leq} 2\alpha_k C(\overline{M}L + \Delta Y) \max_J \|\delta_k^J\|, \qquad (3.31)
\end{aligned}
$$

where $(a) - (b)$ follow from bounded update conditions for multiplicative and additive perturbations. We combine the bounds on T_1, T_2 and T_3 to get a bound on Λ,

$$
\begin{aligned}
\Lambda \leq &- 2\alpha_k M \left(f(\overline{x}_{0,k}) - f(y)\right) + 4\alpha_k^2 N \Delta C^2 \overline{M}(\overline{M}L + Y)(1 + \frac{1}{S}\eta_k^2) + 2\alpha_k NC\overline{M} \max_J \|\delta_k^J\|(1 + \frac{1}{S}\eta_k^2) \\
&+ 2\alpha_k C(\overline{M}L + \Delta Y) \max_J \|\delta_k^J\|. \qquad (3.32)
\end{aligned}
$$

We use this bound in our iterate relation:

$$
\begin{aligned}
\mathbb{E}[\eta_{k+1}^2 | F_{0,k}] &\leq \eta_k^2 + \alpha_k^2 C_1^2 + \Lambda \\
&\leq \eta_k^2 + \alpha_k^2 C_1^2 - 2\alpha_k M \left(f(\overline{x}_{0,k}) - f(y)\right) + 4\alpha_k^2 N \Delta C^2 \overline{M}(\overline{M}L + Y)(1 + \frac{1}{S}\eta_k^2) \\
&\quad + 2\alpha_k NC\overline{M} \max_J \|\delta_k^J\|(1 + \frac{1}{S}\eta_k^2) + 2\alpha_k C(\overline{M}L + \Delta Y) \max_J \|\delta_k^J\| \\
&\leq (1 + A_k)\, \eta_k^2 - 2\alpha_k M \left(f(\overline{x}_{0,k}) - f(y)\right) + B_k, \qquad (3.33)
\end{aligned}
$$

where $A_k = 4\frac{1}{S}\alpha_k^2 N \Delta C^2 \overline{M}(\overline{M}L + Y) + 2\frac{1}{S}\alpha_k NC\overline{M} \max_J \|\delta_k^J\|$ and $B_k = \alpha_k^2 C_1^2 + 4\alpha_k^2 N \Delta C^2 \overline{M}(\overline{M}L + Y) + 2\alpha_k NC\overline{M} \max_J \|\delta_k^J\| + 2\alpha_k C(\overline{M}L + \Delta Y) \max_J \|\delta_k^J\|$. This proves the iterate lemma. □

Proof of Theorem 6

Note that the expression in iterate lemma has similar structure as dissipation inequality in Lemma 10 and we exploit this to prove convergence. We observe that A_k (respectively γ_k in Lemma 10) is a random sequence albeit with a deterministic bound. We can make similar statement about B_k. The randomness in both A_k and B_k is a result of the term $\max_J \|\delta_k^J\|$ being a stochastic quantity due to the use of stochastic gradient updates. However, the deterministic upper bound on the disagreement between model parameters and the average as established in Lemma 11 allows us to show $\sum_k A_k < \infty$ and $\sum_k B_k < \infty$ easily. We first prove that $\sum_k \alpha_k \max_J \|\delta_k^J\| < \infty$ a.s. and use it to prove that $\sum_k A_k < \infty$ and $\sum_k B_k < \infty$. Invoking Lemma 10 we conclude that $\sum_k \alpha_k (f(\overline{x}_{0,k}) - f(y)) < \infty$ and $\lim_{k\to\infty} \eta_k$ exists. We can prove convergence of iterate average to the optimum with probability 1. This proves Theorem 6. We reproduce the entire proof below.

Proof. We use almost supermartingale convergence result (Lemma 10) from [75]. First observe that the expression in iterate lemma (Lemma 12) has the desired structure as seen in Lemma 10. We consider

Lemma 10 with $y = x^*$ and consequently η_k^2 represents the distance of server iterates from the optimum. We can do this as the set of optimum is included in the feasible set, i.e. $\mathcal{X}^* \subset \mathcal{X}$. Recall $f(x^*) = f^*$ notation. We have,

$$\mathbb{E}[\eta_{k+1}^2 | F_{0,k}] = (1 + A_k)\eta_k^2 - 2\alpha_k M(f(\overline{x}_{0,k}) - f(x^*)) + B_k. \tag{3.34}$$

Recall that A_k and B_k are non-negative, real sequences. As $f(\overline{x}_{0,k}) - f(x^*) \geq 0$, it is also a non-negative, real sequence. Next we observe that we need to show $\sum_k A_k < \infty$ and $\sum_k B_k < \infty$ to be invoke Robbins-Siegmund lemma (Lemma 10). Our first step is to prove $\sum_k A_k < \infty$ and $\sum_k B_k < \infty$.

We begin by showing that $\sum_k \alpha_k \max_J \|\delta_k^J\| < \infty$ using disagreement lemma (Lemma 11) as follows,

$$\begin{aligned}
&\sum_k \alpha_k \max_J \|\delta_k^J\| = \alpha_1 \|\delta_1^J\| + \sum_{k=1}^{\infty} \alpha_{k+1} \max_J \|\delta_{k+1}^J\| \\
&\overset{(a)}{\leq} \alpha_1 \|\delta_1^J\| \\
&+ \sum_{k=1}^{\infty} \left(S\alpha_{k+1}\theta\beta^{k+1} \max_I \|x_{0,0}^I\| + 4\alpha_{k+1}\alpha_k \Delta C(\overline{M}L + Y) + 2\alpha_{k+1}\theta\Delta SC(\overline{M}L + Y) \sum_{l=1}^{k} \beta^{k+1-l}\alpha_{l-1} \right) \\
&\overset{(b)}{\leq} \underbrace{\alpha_1 \|\delta_1^J\|}_{U_0} + \underbrace{S\theta \max_I \|x_{0,0}^I\| \sum_{k=1}^{\infty} \alpha_{k+1}\beta^{k+1}}_{U_1} + \underbrace{4\Delta C(\overline{M}L + Y) \sum_{k=1}^{\infty} \alpha_k^2}_{U_2} \\
&\qquad \underbrace{+ 2\theta\Delta SC(\overline{M}L + Y) \sum_{k=1}^{\infty} \alpha_{k+1} \sum_{l=1}^{k} \beta^{k+1-l}\alpha_{l-1}}_{U_3},
\end{aligned} \tag{3.35}$$

where (a) follows from Lemma 11 and (b) follows from $\alpha_k \geq \alpha_{k+1}$.

First observe that $U_0 < \infty$ due to compact feasible set and $U_2 < \infty$ since $\sum_k \alpha_k^2 < \infty$ from assumptions on step size. Moreover, it is easy to see that $U_1 < \infty$. This follows from the ratio test for convergence of series,

$$\lim_{k\to\infty} \sup \frac{\alpha_{k+2}\beta^{k+2}}{\alpha_{k+1}\beta^{k+1}} = \lim_{k\to\infty} \sup \frac{\alpha_{k+2}\beta}{\alpha_{k+1}} < 1 \implies \sum_{k=1}^{\infty} \alpha_{k+1}\beta^{k+1} < \infty.$$

We observe the fact that $\sum_{k=1}^{\infty} \alpha_{k+1} \sum_{l=1}^{k} \beta^{k+1-l}\alpha_{l-1} \leq \sum_{k=1}^{\infty} \sum_{l=1}^{k} \beta^{k+1-l}\alpha_{l-1}^2$ from the fact that $\alpha_{k+1} \leq \alpha_{l-1}$. And use Lemma 3 from [38], to conclude $U_3 < \infty$. Thus $\sum_k \alpha_k \max_J \|\delta_k^J\| < \infty$.

We use the fact that $\sum_k \alpha_k \max_J \|\delta_k^J\| < \infty$ and $\sum_k \alpha_k^2 < \infty$ to conclude $\sum_k A_k < \infty$ and $\sum_k B_k < \infty$,

$$\begin{aligned}
\sum_k A_k &= \sum_k \left(4\frac{1}{S}\alpha_k^2 N\Delta C^2 \overline{M}(\overline{M}L + Y) + 2\frac{1}{S}\alpha_k NC\overline{M} \max_J \|\delta_k^J\| \right) \\
&= 4\frac{1}{S}N\Delta C^2\overline{M}(\overline{M}L + Y) \sum_k \alpha_k^2 + 2\frac{1}{S}NC\overline{M} \sum_k \alpha_k \max_J \|\delta_k^J\| < \infty,
\end{aligned}$$

$$\sum_k B_k = \sum_k \Big(\alpha_k^2 \left(C_1^2 + 4N\Delta C^2 \overline{M}(\overline{M}L + Y) \right) + 2\alpha_k C(\overline{M}L + \Delta Y) \max_J \|\delta_k^J\| + 2\alpha_k NC\overline{M} \max_J \|\delta_k^J\| \Big)$$
$$= \left(C_1^2 + 4N\Delta C^2 \overline{M}(\overline{M}L + Y) \right) \sum_k \alpha_k^2 + (2NC\overline{M} + 2C(\overline{M}L + \Delta Y)) \sum_k \alpha_k \max_J \|\delta_k^J\|.$$

We use an argument similar to observed in [38]. Using Lemma 10 we conclude that the sequence η_k^2 converges to a non-negative random variable and $\sum_k \alpha_k (f(\overline{x}_{0,k}) - f^*) < \infty$ with probability 1. From the disagreement lemma we know that the iterates converge towards each other. Using the fact that $\sum_k \alpha_k (f(\overline{x}_{0,k}) - f^*) < \infty$, $\sum_k \alpha_k = \infty$, and the continuity of objective function $f(x)$ we can conclude that the iterate average $\overline{x}_{0,k}$ must converge to $\mathcal{X}^*$ with probability 1 and hence the iterates must also converge to $\mathcal{X}^*$ with probability 1. □

3.5.3 Privacy Analysis

Proof of Claim 9

Proof. The fundamental reason for privacy in POLAR-SGD is the fact that clients use arbitrary multiplicative and additive perturbations when a stochastic gradient estimate is uploaded to the parameter server. While a balancing gradient estimate is uploaded to other parameter servers (due to SLC, Assumption 1), the secure consensus protocol used by parameter servers (POLAR-SGD Server Algorithm, Algo. 4) ensures that no honest-but-curious parameter server can estimate the balancing (sub)gradient estimate during the consensus iteration. The consensus protocol in Algorithm 4 is information theoretically secure due to the wrap-around feature of modulo arithmetic [73]. Hence, the most that an honest-but-curious parameter server can learn about a client is through directly received gradients (Claim 9). □

As discussed above there are two possible methods of violating privacy. The first being direct observation of received gradient from a specific client and the second being information about gradient sent to a different server leaking during consensus. The received gradients are perturbed by multiplicative and additive arbitrary perturbations. The additive perturbation hides gradients especially when the gradient is close to zero (since $W[J,h]g_h(u,\xi)$ will be close to zero too). This protects gradients from direct observation from an adversary. Note that the perturbations are arbitrary and unknown to the adversary, protecting the stochastic gradients. No information leakage happens during consensus as we use secure multi-party computation based private consensus algorithm to average model parameters [23, 73].

It is important to note that, while private objective functions owned by individual clients are protected against honest-but-curious adversaries, the overall cost function $f(x)$ is not. Consider a scenario where $\Delta = 1$, where projected descent and consensus steps occur alternatively. As the private consensus step computes the exact average, an adversary can estimate the effect of stochastic gradients on servers other than itself. This along with stochastic gradient directly received from a client, an adversary can estimate

the gradient of the whole function, violating the privacy of $f(x)$.

3.5.4 Extensions

We assume that a parameter server is adversarial. This can be extended to multiple (τ) adversarial parameter servers scenario, however this requires that clients upload gradients to at least $\tau + 1$ parameter servers.

We make an assumption that the parameter servers form a fully connected graph. This assumption allows us to use secure consensus scheme from [73] and prove privacy with relative ease. If the parameter servers are not fully connected but only connected we can claim correctness following our analysis Section 3.5.2, however, the privacy analysis is expected to be difficult.

Moreover, it is easy to extend POLAR-SGD to have coordinate wise different multiplicative weights. Under mild additional conditions like per coordinate satisfaction of symmetric learning and bounded update condition we can extend the correctness results. We discuss this in our technical report [85].

3.6 Experimental Validation

In this section we empirically validate POLAR-SGD. We consider linear regression problem on a large synthetic dataset. We consider 100000 data points partitioned among $C = 100$ clients. Our system consists of $S = 5$ parameter servers. The clients upload gradients to more than one parameter server in each iteration and the servers form a fully connected graph and perform secure consensus ever Δ iterations.

If dataset D_i is stored at client i, then we write the local objective function at client i as,

$$f_i(x) = \sum_{l \in D_i} \|x^T a_l - b_l\|^2,$$

where (a_l, b_l) are data points belonging to D_i. And the overall objective function becomes

$$f(x) = \sum_i f_i(x) = \sum_{i=1}^{C} \sum_{l \in D_i} \|x^T a_l - b_l\|^2.$$

We consider multiplicative perturbations and additive perturbations that satisfy SLC and BUC with parameters $M = 5$ and $\overline{M} = 50$. We consider three values of $\Delta = \{10, 20, 50\}$ and compare performance. We also compare performance of POLAR-SGD with respect to non-private algorithm and show the trade-off between privacy and convergence speed. Note that we use batch size of 10 samples for computing stochastic gradients in each iteration.

Figure 3.3 shows the sub-optimality of iterates with respect to iterations. First observe that POLAR-SGD iterates converge (in function value) to the optimum for each Δ. Observe that the non-private algorithm presented earlier converges much faster than POLAR-SGD. As expected, with higher Δ, it takes more

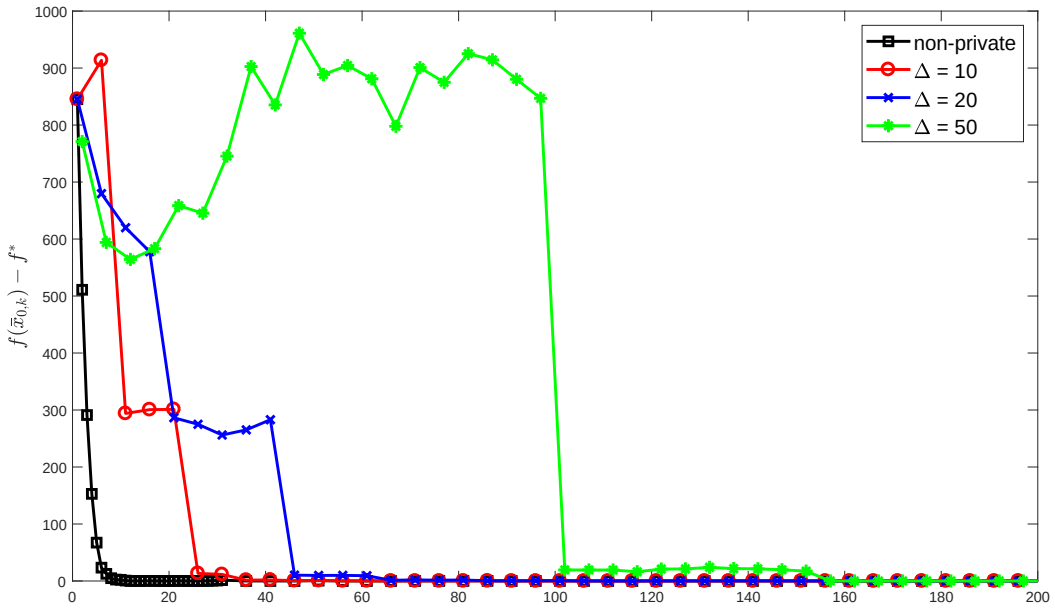


Figure 3.3: Sub-optimality v/s iterations.

iterations for multiplicative and additive perturbations to average out and hence it take longer to converge. In case of smaller Δ's (like 10 and 20) the iterates follow the general trend of the non-private algorithm very closely. However with larger Δ the function sub-optimality may increase as compared to the starting sub-optimality for few iterations.

3.7 Conclusion

In this chapter we consider privacy preserving distributed optimization algorithm for a multiple parameter server architecture. Our algorithm uses additive and multiplicative weights to perform gradient obfuscation. The parameter servers use a secure consensus protocol to average the model parameters with privacy. We show correctness of our algorithm and discuss privacy for polynomial objective functions. We show that privacy and accuracy coexist in our framework. We also experimentally validate our algorithm on the linear regression problem.

CHAPTER 4

PRIVATE EQUILIBRIUM COMPUTATION ALGORITHM FOR NETWORK AGGREGATE GAMES

4.1 Introduction

Aggregate games are non-cooperative games in which a player's payoff or cost depends on its own actions and the sum-total of the actions taken by other players. In a Cournot oligopoly for example, firms compete to supply a product in a market with a price-responsive demand with a goal to maximize profit. A firm's profit depends on its production cost as well as the market price, where the latter only depends on the aggregate quantity of the product offered in the market by all firms. Aggregate games are widely studied in the literature, e.g., see [89, 90]. Multiple strategic interactions in practice admit an aggregate game model, e.g., Cournot competition models for wholesale electricity markets in [91–93], supply function competition in general economies see [90], communication networks in [94,95] and common agency games in [96]. Aggregate games are often potential games and a pure-strategy Nash equilibrium can be guaranteed to exist. In this dissertation, we present an algorithm for networked players to compute such an equilibrium in a distributed fashion that maintains the privacy of players' cost structures.

Players in a networked game can only communicate with neighboring players in a communication graph. Distributed algorithms for computing Nash equilibrium in networked games have a rich literature, e.g., see [13, 97–100]. The obvious difficulty in computing equilibrium strategy arises due to the inability of a player to observe the aggregate decision. Naturally distributed Nash computation proceeds via iterative estimation of the aggregate decision followed by local payoff maximization (or cost minimization) with a given aggregate estimate. References [13,100] exploit consensus based averaging, [13,97] explore gossip based averaging, and [99] employs gradient play along with acceleration for aggregate estimation over networks.

4.1.1 Our Contributions

Algorithms for equilibrium computation were not designed with privacy in mind. We show in Section 4.2.5, that an honest-but-curious adversary can compromise a few nodes in the network and observe the sequence of estimates to infer other players' payoff or cost structures for the algorithm in [13]. In other words, information that allows distributed equilibrium computation can leak players' sensitive private information to adversaries.

Distributed equilibrium computation algorithms require aggregate estimates to update their own actions. Our proposed algorithm obfuscates local aggregate estimates before sharing them with neighbors. The obfuscation step involves players adding correlated perturbations to each outgoing aggregate estimate. The perturbations are designed such that they add to zero for each player. The received perturbed aggregate estimates are averaged by each player and used for updating strategy using local projected gradient descent.

Our main result (Theorem 7) reveals that obfuscation via correlated perturbations prevents an adversary from accurately learning cost structures provided the network satisfies appropriate connectivity conditions. Players converge to exact Nash equilibrium asymptotically. In other words, we simultaneously achieve both privacy and accuracy in distributed Nash computation in aggregate games. This is in sharp contrast to differentially private algorithms where trade-offs between accuracy and privacy guarantee are fundamental, e.g., see [61].

Simulations in Section 4.4 validate our results and corroborate our intuition that obfuscation slows down but does not impede the convergence of the algorithm.

4.2 Equilibrium Computation in Aggregate Games and the Lack of Privacy

We begin by introducing a networked aggregate game. We then present an adversary model and show that prior distributed equilibrium computation algorithms leak private information of players. This exposition motivates the development of privacy-preserving algorithms for equilibrium computation in the next section.

4.2.1 The Networked Aggregate Game Model

Consider a game with n players that can communicate over a fixed undirected network with reliable lossless links. Model this communication network by graph $\mathfrak{G}(\mathcal{V}, \mathcal{E})$, where each node in $\mathcal{V} := \{1, \ldots, n\}$ denotes a player. Two players i and j can communicate with each other if and only if they share an edge in $\mathcal{E}$, denoted as $(i, j) \in \mathcal{E}$. Call $\mathcal{N}_i$ the set of neighbors of node i and $i \in \mathcal{N}_i$ by definition.

Player i can take actions in a convex compact set $\mathcal{X}^i \subseteq \mathbb{R}^d$, where $\mathbb{R}$ denotes the set of real numbers. Define $\overline{\mathcal{X}}$ as the Minkowski (set) sum of $\mathcal{X}^i$'s and

$$\overline{x} := \sum_{j=1}^{n} x^j, \tag{4.1}$$

as the aggregate action of all players. For convenience, define $\overline{x}^{-i} := \sum_{j \neq i} x^i$. We assume that $\cap_{i=1}^n \mathcal{X}^i$ is non-empty. For an action profile $(x^1, \ldots, x^n)$, player i incurs a cost that takes the form $f_i(x^i, \overline{x}) := f_i\left(x^i, x^i + \overline{x}^{-i}\right)$. This defines an aggregate game in that the actions of other players affect player i only through the sum of actions of all players, $\overline{x}$.

Each player $i \in \mathcal{V}$ thus seeks to solve

$$\begin{aligned} &\text{minimize} \quad && f_i(x^i, x^i + \overline{x}^{-i}), \\ &\text{subject to} \quad && x^i \in \mathcal{X}^i. \end{aligned} \tag{4.2}$$

For each $i \in \mathcal{V}$, assume that $f_i(x^i, y)$ is continuously differentiable in (x^i, y) over a domain that contains $\mathcal{X}^i \times \overline{\mathcal{X}}$. Furthermore, for each $i \in \mathcal{V}$, let $x^i \mapsto f_i(x^i, \overline{x})$ be convex over $\mathcal{X}^i$ and the gradient $\nabla_{x^i} f_i$ be uniformly $\overline{L}$-Lipschitz, i.e., $\exists\ \overline{L} > 0$ such that,

$$\|\nabla_{x^i} f_i(x^i, u) - \nabla_{x^i} f_i(x^i, u')\| \leq \overline{L}\|u - u'\|, \tag{4.3}$$

for all u, u' in $\overline{\mathcal{X}}$, x^i in $\mathcal{X}^i$. Throughout, $\|\cdot\|$ stands for the ℓ_2-norm of its argument. Define $\mathcal{X} := \times_{i=1}^n \mathcal{X}^i$ and the gradient map

$$\phi(x) := \begin{pmatrix} \nabla_{x^1} f_1(x^1, \overline{x}) \\ \vdots \\ \nabla_{x^n} f_N(x^n, \overline{x}) \end{pmatrix}, \tag{4.4}$$

for $x := ({x^1}^\mathsf{T}, {x^2}^\mathsf{T}, \ldots, {x^n}^\mathsf{T})^\mathsf{T} \in \mathcal{X}$. Assume throughout that ϕ is strictly monotone over $\mathcal{X}$, i.e.,

$$[\phi(x) - \phi(x')]^\mathsf{T} (x - x') > 0, \tag{4.5}$$

for all $x, x' \in \mathcal{X}$ and $x \neq x'$. Denote this game in the sequel by $\mathsf{game}(\mathfrak{G}, \{f_i, \mathcal{X}^i\}_{i \in \mathcal{V}})$.

To provide a concrete example, consider the well-studied Nash-Cournot game (see [7]) among n suppliers competing to offer into a market for a single commodity where the price p varies with demand D as $p(D) := a - bD$. Supplier i offers to produce x^i amount of goods within its production capability modeled as $\mathcal{X}^i \subseteq \mathbb{R}_+$. Here $\mathbb{R}_+$ denotes the set of non-negative real numbers. To produce x^i, supplier i incurs a cost of $c_i(x^i)$, where c_i is increasing, convex and differentiable. Each supplier seeks to maximize its profit, or equivalently, minimize its loss. The loss of supplier i is

$$f_i(x^i, \overline{x}) = c_i(x^i) - x^i p(\overline{x}) = c_i(x^i) - x^i(a - b\overline{x}).$$

4.2.2 Equilibrium Definition and Existence

An action profile $(x^1_*, \ldots x^n_*)$ defines a Nash equilibrium of $\mathsf{game}(\mathfrak{G}, \{f_i, \mathcal{X}^i\}_{i \in \mathcal{V}})$ in pure strategies, if

$$f_i\left(x^i_*, x^i_* + \overline{x}^{-i}_*\right) \leq f_i\left(x^i, x^i + \overline{x}^{-i}_*\right),$$

for all $x^i \in \mathcal{X}^i$ and $i \in \mathcal{V}$.

The networked aggregate game, as described above, always admits a unique pure strategy Nash equilibrium. See Theorem 2.2.3 in [101] for details. Given that an equilibrium always exists, prior literature has studied distributed algorithms for players to compute such an equilibrium.

4.2.3 Prior Algorithms for Distributed Nash Computation

We now describe the distributed algorithm for Nash equilibrium computation from [13]. This algorithm iteratively solves $\mathsf{game}(\mathfrak{G}, \{f_i, \mathcal{X}^i\}_{i\in\mathcal{V}})$. In Section 4.2.5, we demonstrate that adversarial players can infer private information about cost structures f_i's from observing a subset of the variables during equilibrium computation using that algorithm. While we only study the algorithm in [13], our analysis can be extended to those presented in [97–100].

Recall that players in $\mathsf{game}(\mathfrak{G}, \{f_i, \mathcal{X}^i\}_{i\in\mathcal{V}})$ do not have access to the aggregate decision. To allow equilibrium computation, let players at iteration k maintain estimates of the aggregate decision $\overline{x}$ as $v_k^1, \ldots, v_k^n$, initialized as, $v_0^i = x_0^i$ for each player i. At discrete time steps $k \geq 0$, each player transmits its own estimate of the aggregate decision to its neighbors and updates its own action as follows,

$$\widehat{v}_k^i = \sum_{j=1}^{n} W_{ij} v_k^j, \tag{4.6a}$$

$$x_{k+1}^i = \mathrm{proj}_{\mathcal{X}^i}\left[x_k^i - \alpha_k \nabla_{x^i} f_i(x_k^i, n\widehat{v}_k^i)\right], \tag{4.6b}$$

$$v_{k+1}^i = \widehat{v}_k^i + x_{k+1}^i - x_k^i. \tag{4.6c}$$

Here, $\mathrm{proj}_{\mathcal{X}^i}$ stands for projection on $\mathcal{X}^i$, and α_k is a common learning rate of all players.

The algorithm has three steps. First, player i computes a weighted average of the estimates of the aggregate received from its neighbors in (4.6a), where W is a symmetric doubly-stochastic weighting matrix. The sparsity pattern of the matrix follows that of graph $\mathfrak{G}$, i.e.,

$$W_{ij} \neq 0 \iff (i,j) \in \mathcal{E}.$$

Second, player i performs a projected gradient update in (4.6b) utilizing the weighted average of local aggregate decision $\widehat{v}_k^i$ in lieu of the true aggregate decision $\overline{x}$. Finally, players update their own estimate of aggregate average in (4.6c) based on its local decision x_k^i and its update x_{k+1}^i.

4.2.4 Adversary Model and Privacy Definition

Consider an adversary A that compromises the players in $\mathcal{A} \subseteq \mathcal{V}$. A is equipped with unbounded storage and computational capabilities, and has access to all information stored, processed locally and communicated to

any compromised players at all times. We define adversary model using the information available to A.

(A) For a compromised node $i \in \mathcal{A}$, A knows all local information f_i, x_k^i, v_k^i, $\widehat{v}_k^i$ and information received from neighbors of i i.e., v_k^j for $j \in \mathcal{N}_i$ at each $k \geq 0$.

(B) A knows the algorithm for equilibrium computation and its parameters $\{\alpha_k\}$ and W.

(C) A observes aggregate decision $\overline{x}_k$, i.e. $\sum_{i=1}^n x_k^i$ per (4.1), at each k.

What does A seek to infer? The dependency of a player's cost on the player's own actions encodes private information. In the Cournot competition example, this dependency is precisely supplier i's production cost – information that is business sensitive. A seeks to exploit information sequence observed from compromised players to infer private information of other players. Intuitively, privacy implies inability of A to infer private cost functions.

Denote the set of non-adversarial nodes by $\mathcal{A}^c := \mathcal{V} \setminus \mathcal{A}$. Call $\mathfrak{G}(\mathcal{A}^c)$ the restriction of $\mathfrak{G}$ to $\mathcal{A}^c$ obtained by deleting the adversarial nodes. See Figure 4.1 for an illustration. For this example, A monitors all variables and parameters pertaining to player 5, but seeks to infer the functions $f_1, \ldots, f_4$.

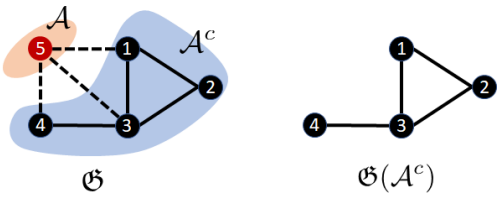


Figure 4.1: Illustration of $\mathfrak{G}$ and $\mathfrak{G}(\mathcal{A}^c)$. Here, $\mathcal{A} = \{5\}$ and $\mathcal{A}^c = \{1, 2, 3, 4\}$.

Let Π denote the set of all permutations over all non-adversarial nodes in $\mathcal{A}^c$. Define the collection of games

$$\mathcal{F} := \Big\{ \mathsf{game}(\mathfrak{G}, \{f_{\pi(i)}, \mathcal{X}^{\pi(i)}\}_{i \in \mathcal{V}}) \;\Big|\; \pi \in \Pi \Big\}.$$

Thus, $\mathcal{F}$ comprises the games where the cost functions and strategy sets of non-adversarial players are permuted. All games in $\mathcal{F}$ have the same aggregate strategy $\overline{x}_*$ at Nash equilibrium. Next, we utilize $\mathcal{F}$ to define privacy.

Definition 2 (Privacy via Non-identifiability)**.** *Consider a distributed algorithm used to compute the Nash equilibrium of* $\mathsf{game}(\mathfrak{G}, \{f_i, \mathcal{X}^i\}_{i \in \mathcal{V}})$*. If execution observed by adversary* A *is consistent with all games in* $\mathcal{F}$*, then the algorithm is private.*

We define privacy as the inability of A to distinguish between games in $\mathcal{F}$. Even if A knew all possible costs exactly–which is a tall order–our privacy definition implies that A cannot associate such costs to specific players.

4.2.5 Privacy Breach in Algorithm (4.6)

Consider a Cournot competition among five players connected according to $\mathfrak{G}$ in Figure 4.1, where A has compromised player 5. Assume that the equilibrium of the game lies in the interior of each player's strategy set. Recall that A stores observed information at each k and processes it to infer private cost information $c_i(x_i)$. We argue how A can compute cost functions $c_1(\cdot), \ldots, c_4(\cdot)$ up to a constant.

- We first show privacy breach for player 4. A observes $\{v_k^1, v_k^3, v_k^4, v_k^5\}$ at each $k \geq 0$. A uses v_k^3, v_k^4, v_k^5 and W to compute $\widehat{v}_k^4$ using (4.6a). Moreover, A uses (4.6c) to compute,

$$x_{k+1}^4 - x_k^4 = v_{k+1}^4 - \widehat{v}_k^4.$$

- For large enough k, the step-size α_k is small enough to ensure that the output after projection is approximately the gradient based update,[1]

$$\text{proj}_{\mathcal{X}^i}\left[x_k^i - \alpha_k \nabla_{x^i} f_i(x_k^i, n\widehat{v}_k^i)\right] \approx x_k^i - \alpha_k \nabla_{x^i} f_i(x_k^i, n\widehat{v}_k^i).$$

  At such large k, A uses (4.6b) along with $(x_{k+1}^4 - x_k^4)$ and α_k to calculate $\nabla_{x^4} f_4(x_k^4, n\widehat{v}_k^4)$. Even if the above inequality is satisfied only approximately, an adversary may still learn an estimate of $\nabla_{x^4} f_4(x_k^4, n\widehat{v}_k^4)$ within an error ball.

- A uses information about structure of loss function i.e. $f_4(x^4, \overline{x}) = c_4(x^4) - x^4(a - b\overline{x})$, along with $\nabla_{x^4} f_4(x_k^4, n\widehat{v}_k^4)$, $\widehat{v}_k^4$, $\overline{x}_k$ and game parameters a, b to learn $c_4'(x_k^4)$. Several observations of $(x_k^4, c_4'(x_k^4))$ allows A to learn the private cost c_4 upto a constant. The exact number of observations needed depends directly on structure of cost c_4, e.g. three observations for quadratic, four observations for cubic, and five observations for quartic $c_4(x)$.

- We showed that privacy breach for player 4, the same analysis can be used for players 1, 2 and 3 with an additional step. A observes $\overline{x}_k$, which is exactly $\frac{1}{n}\sum_i v_k^i$ (Lemma 2 in [13]). A computes

$$v_k^2 = n\overline{x}_k - (v_k^1 + v_k^3 + v_k^4 + v_k^5).$$

  Since $\{v_k^2\}$ is available for each $k \geq 0$, A uses same process as above to show privacy breach for players 1, 2 and 3.

For algorithm (4.6), A uncovers all private cost functions $c_i(\cdot)$ for an example aggregate game. Next, we design an algorithm that protects privacy of players' private information in the sense of Definition 2 against A.

[1] This follows from the definition of projection operator as used on convex sets, specifically, if $\mathcal{X}$ is a convex compact set then $\text{proj}_{\mathcal{X}}(x) = x$ for all $x \in \mathcal{X}$. In this instance, we argue that if α_k is small enough, x_k^i lies in the interior of $\mathcal{X}^i$ implies $x_k^i - \alpha_k \nabla_{x^i} f_i(x_k^i, n\widehat{v}_k^i) \in \mathcal{X}^i$ and lets us show that projection operator behaves like an identity operator.

4.3 Our Algorithm and Its Properties

We propose and analyze Algorithm 5 that computes Nash equilibrium of $\mathsf{game}(\mathfrak{G}, \{f_i, \mathcal{X}^i\}_{i\in\mathcal{V}})$ in a distributed fashion. The main result (Theorem 7) shows that the algorithm asymptotically converges to the equilibrium. Attempts by A to recover each player's cost structure, however, remain unsuccessful.

The key idea behind our design is the injection of correlated noise perturbations in the exchange of local estimates of the aggregate decision. Different neighbors of player i receive different estimates of the aggregate decision. The perturbations added by any player i add to zero. While A may still infer the true aggregate decision, the protocol does not allow the adversary to correctly infer the players' iterates or the gradients of their costs with respect to their own actions. Our assumption on network connectivity requires $\mathcal{G}(\mathcal{A}^c)$ be connected and non-bipartite. Under these conditions A cannot monitor all outgoing communication channels from any player. We further show that one can design noises in a way that A's observations are consistent with all games in $\mathcal{F}$, making it impossible for him to uncover cost for any specific player.

Throughout, assume that W is a doubly stochastic that follows the sparsity pattern of $\mathfrak{G}$. Further, assume that all non-diagonal, non-zero entries of W are identically $\delta < \frac{1}{n-1}$.

At each time k, player i generates correlated random numbers $\{r_k^{ij}\}$ satisfying $r_k^{ii} = 0$ and $\sum_{j\in\mathcal{N}_i} r_k^{ij} = 0$. Player i then adds $\alpha_k r_k^{ij}$ to v_k^i to generate v_k^{ij}, the estimate sent by player i to player j, according to (4.9). Let $\boldsymbol{r}$ denote the collection of r's for all players across time. Call $\boldsymbol{r}$ the obfuscation sequence.

Each node i computes weighted average of received aggregate estimates v_k^{ji} to construct its own estimate aggregate decision $n\widehat{v}_k^i$, following (4.10). Players perform projected gradient descent using local decision estimate x_k^i, gradient of cost function $\nabla_{x^i} f_i(x_k^i, n\widehat{v}_k^i)$, and non-summable, square-summable step size α_k (see (4.7)) to arrive at an improved local decision estimate x_{k+1}^i using (4.11). Players then update their local aggregate estimate using the change in local decision estimate $x_{k+1}^i - x_k^i$ per (4.12).

The properties of our algorithm are summarized in the next result. The proof is included in Section 4.5.

Theorem 7. *Consider a networked aggregate game defined as* $\mathsf{game}(\mathfrak{G}, \{f_i, \mathcal{X}_i\}_{i\in\mathcal{V}})$. *If* $\mathfrak{G}(\mathcal{A}^c)$ *is connected and non-bipartite, then Algorithm 5 is private as per Definition 2. Moreover, if the obfuscation sequence is bounded, then Algorithm 5 asymptotically converges to a Nash equilibrium of the game.*

The convergence properties largely mimic that of distributed descent algorithms for equilibrium computation. The locally balanced and bounded nature of the designed noise together with decaying step-sizes ultimately drown the effect of the noise. Computing balanced yet bounded perturbations can be achieved using secure multiparty computation protocols described in [20,36,73]. Our assumption on $\mathfrak{G}(\mathcal{A}^c)$ is such that given two games $F, \tilde{F}$ from $\mathcal{F}$ and an obfuscation sequence $\boldsymbol{r}$, we are able to design a different obfuscation sequence $\tilde{\boldsymbol{r}}$, such that the execution of F perturbed with $\boldsymbol{r}$ generates identical observables as $\tilde{F}$ perturbed with $\tilde{\boldsymbol{r}}$. The connectivity among non-adversarial players in $\mathcal{A}^c$ is key to the success of our algorithm design. Convergence speed depends on the size of the perturbations. We investigate this link experimentally in

Algorithm 5 Private Distributed Nash Computation

Input: Player i knows $f_i(x^i, \overline{x})$, $\mathcal{X}^i$, and δ. Consider a non-increasing non-negative sequence $\boldsymbol{\alpha}$ that satisfies

$$\sum_{k=1}^{\infty} \alpha_k = \infty \text{ and } \sum_{k=1}^{\infty} \alpha_k^2 < \infty. \tag{4.7}$$

Initialize: For $i \in \mathcal{V}$, $v_0^i = x_0^i = \varkappa \in \cap_i \mathcal{X}^i$.

For $k \geq 0$, players $i \in \mathcal{V}$ execute in parallel:

1: Construct $|\mathcal{N}_i|$ random numbers $\{r_k^{ij}\}$, satisfying

$$r_k^{ii} = 0 \text{ and } \sum_{j \in \mathcal{N}_i} r_k^{ij} = 0. \tag{4.8}$$

2: Send obfuscated aggregate estimates v_k^{ij} to $j \in \mathcal{N}_i$, where

$$v_k^{ij} = v_k^i + \alpha_k r_k^{ij}. \tag{4.9}$$

3: Compute weighted average of received estimates v_k^{ji} as

$$\widehat{v}_k^i = \sum_{j=1}^{n} W_{ij} v_k^{ji}. \tag{4.10}$$

4: Perform a projected gradient descent step as

$$x_{k+1}^i = \text{proj}_{\mathcal{X}^i}[x_k^i - \alpha_k \nabla_{x^i} f_i(x_k^i, n\widehat{v}_k^i)]. \tag{4.11}$$

5: Update local aggregate estimate as

$$v_{k+1}^i = \widehat{v}_k^i + x_{k+1}^i - x_k^i. \tag{4.12}$$

Section 4.4, but leave analytical characterization of this relationship for future work.

Remark 2. *Observe that we can select random perturbations, r_k^{ij}, to be gradients of a random function, $p^{ij}(x)$, computed at local strategy estimate x_k^i, i.e. $r_k^{ij} = \nabla p^{ij}(x_k^i)$. If agent i designs these random functions, $\{p^{ij}(x),\ j \in \mathcal{N}_i\}$, locally to satisfy $\sum_{j \in \mathcal{N}_i} p^{ij}(x) = 0$, then the random perturbations computed by using gradients will satisfy locally balanced property required in* (4.8)*. Moreover, as each $\mathcal{X}^i$ is a compact set by assumption, the random perturbations computed as gradients of random functions will be bounded.*

In what follows, we compare our algorithm and its properties to other protocols for privacy preservation.

Comparison with Differentially Private Algorithms: Differentially private algorithms for computing Nash equilibrium of potential games have been studied in [102, 103]. The algorithm in [102] executes a differentially private distributed mirror-descent algorithm to optimize the potential function. Experiments reveal that a trade-off arises between accuracy and privacy parameters, i.e., the more privacy one seeks, the less accurate the final output of the algorithm becomes. Such a tradeoff is a hallmark of differentially private

algorithms, e.g., see [61]. Our algorithm on the other hand does not suffer from that limitation. The privacy guaranteed in our work is weaker as some information, such as the aggregate action ($\overline{x}$), is exactly observed. Notice that our definition of privacy is binary in nature. That is, an algorithm for equilibrium computation can either be private or non-private. We aim to explore properties of our algorithmic architecture with notions of privacy that allow for a degree of privacy and compare them with differentially private algorithms.

Comparison to Cryptographic Methods: Authors in [104] use secure multiparty computation to compute Nash equilibrium. Such an approach guarantees privacy in an information theoretic sense. This protocol provides privacy guarantees along with accuracy, similar to our algorithmic framework. However, cryptographic protocols are typically computationally expensive for large problems (see Section V in [55]), and are often difficult to implement in distributed settings.

Comparison to Private Distributed Optimization: Our earlier work in [20] has motivated the design of Algorithm 5. While our prior work seeks privacy-preserving distributed protocols to cooperatively solve optimization problems, the current chapter focuses on non-cooperative games. Protocols in [20] advocate use of two sets of perturbations – perturbations that cancel over the network and perturbations that cancel locally. Just network-wide balanced noise design is not appropriate for networked games for two reasons. First, players must agree on noise design strategy, a premise that requires cooperation and may be unacceptable in non-cooperative games like aggregate game considered here. Second, perturbing local functions f_i's, even if the changes cancel in aggregate, can alter the equilibrium of the game.

Privacy in Client-Server Architecture: This work considers players communicating over a peer-to-peer network. However, engineered distributed systems often have a client-server architecture. Presence of a central server entity allows for easy aggregate computation. However, privacy is sacrificed if the parameter server is adversarial. We have investigated privacy preservation for distribution optimization in this architecture in [27], where we use multiple central servers instead of one, a subset of which can be adversarial. We believe our algorithm design and analysis in [27] can be extended to deal with private equilibrium computation for aggregate games in client-server framework.

4.4 A Numerical Experiment

Consider a Cournot competition with $n = 10$ players over $\mathfrak{G}$ described in Figure 4.2. Player i's cost is given by

$$c_i(x^i) = \zeta_{i,2}(x^i)^2 + \zeta_{i,1}(x^i).$$

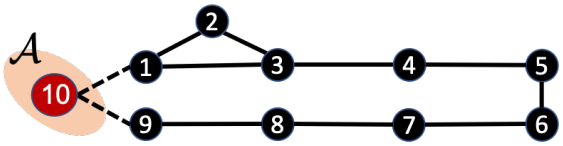


Figure 4.2: Communication network for Cournot network example on $n = 10$ players.

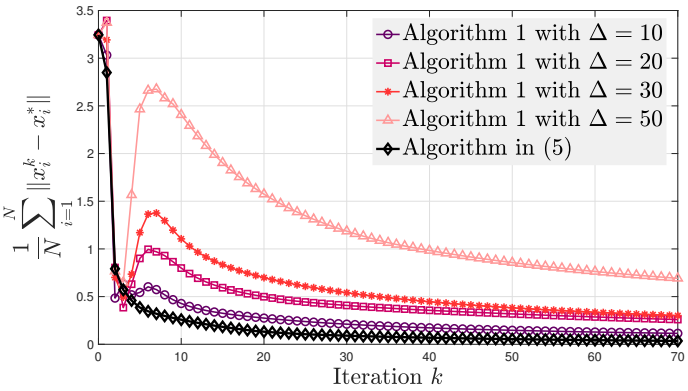


Figure 4.3: Iterates of Algorithm 5 versus the Algorithm in (4.6) for $\Delta = \{10, 20, 30, 50\}$.

The cost coefficients are drawn randomly from

$$\zeta_{i,2} \sim \mathsf{unif}[0, 1/2],\ \zeta_{i,1} \sim \mathsf{unif}[0, 1],$$

for each i. The strategy sets are identically $\mathcal{X}^i = [0, 5]$ for each i. Choose $\delta = \frac{1}{10}$ that parameterizes the matrix W. Let the price vary with demand D as

$$p(D) = 6 - \frac{1}{10}D.$$

We initialize the algorithm with $x_0^i = 1$ identically for all players i. We use secure multi-party computing technique in [20] to design obfuscation sequence $\boldsymbol{r}$ that satisfies (4.8) and

$$|r_k^{ij}| \leq \Delta.$$

The trajectory of the average distance of x_k^i's from x_*^i across players with $\alpha_k := (k+1)^{0.51}$ is shown in Figure 4.3.

Our algorithm converges to the equilibrium similar to the non-private algorithm in (4.6). However, its convergence is slower as seen in Figure 4.3. The slowdown is especially pronounced for large Δ's and is an artifact of perturbations added by players to obfuscate information from the adversary. Thus, our algorithm design achieves privacy and asymptotic convergence to equilibrium, but sacrifices speed of convergence. An

analytical characterization of the slowdown defines an interesting direction for future work.

4.5 Privacy Analysis and Proof of Theorem 7

4.5.1 Proving Algorithm 5 is Private

Recall that $\mathfrak{G}(\mathcal{A}^c)$ is the graph over non-adversarial nodes $\mathcal{A}^c$. Suppose $\mathcal{A}^c$ has M nodes. Let I, J be two players in $\mathcal{A}^c$ and

$$F := (f_i, \mathcal{X}^i)_{i\in\mathcal{V}}, \quad \tilde{F} := (\tilde{f}_i, \tilde{\mathcal{X}}^i)_{i\in\mathcal{V}},$$

be two games in $\mathcal{F}$ such that $\tilde{F}$ is identical to F, except that costs and strategy sets of players I and J are switched:

$$\tilde{f}_I = f_J,\ \tilde{f}_J = f_I,\ \tilde{\mathcal{X}}^I = \mathcal{X}^J,\ \tilde{\mathcal{X}}^J = \mathcal{X}^I.$$

For convenience, define $\pi : \mathcal{V} \to \mathcal{V}$ as the permutation that encodes the switch, i.e.,

$$\pi(I) = J,\ \pi(J) = I,\ \text{and } \pi(i) = i \text{ for all } i \neq I, J.$$

Consider the execution of Algorithm 5 on F with obfuscation sequence $\boldsymbol{r}$ used in (4.9), and x_0^j initialized as $x_0^j = \varkappa \in \cap_{i=1}^n \mathcal{X}^i$, for each $j = 1, \ldots, n$,

$$\mathsf{E}(F, \boldsymbol{r}, \varkappa) := \{(x_k^i, v_k^i, \widehat{v}_k^i) \text{ for } i \in \mathcal{V}, k \geq 0\}.$$

We require all players to initialize their actions/decisions with the same value. We prove that there exists an obfuscation sequence $\tilde{\boldsymbol{r}}$ such that execution $\mathsf{E}(\tilde{F}, \tilde{\boldsymbol{r}}, \varkappa)$ of Algorithm 5 on game $\tilde{F}$ with obfuscation sequence $\tilde{\boldsymbol{r}}$ and x_0^j initialized at $\varkappa$ (for each $j = 1, \ldots, n$), is identical to $\mathsf{E}(F, \boldsymbol{r}, \varkappa)$, from A's perspective. An arbitrary permutation over $\mathcal{A}^c$ is equivalent to a composition of a sequence of switches among two players in $\mathcal{A}^c$. As a result, the algorithm execution on games in $\mathcal{F}$ can be made to appear identical from A's standpoint, proving the privacy of Algorithm 5.

In the rest of the proof, we show how to construct $\tilde{\boldsymbol{r}}$ that ensures $\mathsf{E}(\tilde{F}, \tilde{\boldsymbol{r}}, \varkappa)$ and $\mathsf{E}(F, \boldsymbol{r}, \varkappa)$ appear identical to A.

Adversary observes $\{x_k^j, v_k^j, \widehat{v}_k^j\}$ for all $j \in \mathcal{A}$ at each $k \geq 0$. Adversary also observes all messages exchanged with a corrupted node.

All perturbations utilized by corrupted nodes $j \in \mathcal{A}$ are same in both executions,

$$\tilde{r}_k^{ji} = r_k^{ji} \text{ for all } j \in \mathcal{A}. \tag{4.13}$$

All messages received by corrupted nodes $j \in \mathcal{A}$ from non-corrupted nodes $i \in \mathcal{A}^c$, denoted by v_k^{ij}, are

identical for both executions to be exactly the same, i.e.,

$$\begin{aligned} \tilde{v}_k^{ij} = v_k^{ij} &\iff \tilde{v}_k^i + \alpha_k \tilde{r}_k^{ij} = v_k^i + \alpha_k r_k^{ij} \\ &\iff \alpha_k \tilde{r}_k^{ij} = v_k^i + \alpha_k r_k^{ij} - \tilde{v}_k^i. \end{aligned} \tag{4.14}$$

Adversary observes $\overline{x}_k$ for each $k \geq 0$. Enforcing

$$\tilde{x}_k^i = x_k^{\pi(i)},\ \tilde{v}_k^i = v_k^{\pi(i)},\ \widehat{\tilde{v}}_k^i = \widehat{v}_k^{\pi(i)}, \tag{4.15}$$

and resulting in $\overline{\tilde{x}}_k = \overline{x}_k$. Recall, the non-diagonal, non-zero entries of W matrix are δ and we get,

$$\begin{aligned} &\widehat{\tilde{v}}_k^i = \widehat{v}_k^{\pi(i)} \\ &\iff \sum_{j \in \mathcal{N}_i} W_{ij}(\tilde{v}_k^j + \alpha_k \tilde{r}_k^{ji}) = \sum_{j \in \mathcal{N}_{\pi(i)}} W_{\pi(i)j}(v_k^j + \alpha_k r_k^{j\pi(i)}) \\ &\iff \sum_{j \in \mathcal{N}_i \cap \mathcal{A}^c} \tilde{r}_k^{ji} = \frac{1}{\alpha_k \delta} \sum_{j \in \mathcal{N}_{\pi(i)}} W_{\pi(i)j}(v_k^j + \alpha_k r_k^{j\pi(i)}) - \frac{1}{\alpha_k \delta} \sum_{j \in \mathcal{N}_i} W_{ij} v_k^{\pi(j)} - \sum_{j \in \mathcal{N}_i \cap \mathcal{A}} \tilde{r}_k^{ji}. \end{aligned} \tag{4.16}$$

The obfuscation used by each player $i \in \mathcal{A}^c$ is locally balanced, and hence, we have

$$\sum_{j \in \mathcal{N}_i} \tilde{r}_k^{ij} = 0 \iff \sum_{j \in \mathcal{N}_i \cap \mathcal{A}^c} \tilde{r}_k^{ij} = - \sum_{j \in \mathcal{N}_i \cap \mathcal{A}} \tilde{r}_k^{ij}. \tag{4.17}$$

Let γ be a vector of dimension $2|\mathcal{E}^c|$, where $\mathcal{E}^c$ is the set of all edges in the graph restricted to non-corrupted nodes, i.e. $\mathcal{E}^c = \mathcal{E}(\mathfrak{G}(\mathcal{A}^c))$. γ is a vector of $\tilde{r}_k^{ij}$'s for $i, j \in \mathcal{A}^c$ ordered as follows – $\tilde{r}_k^{ij}$ for all $(i,j) \in \mathcal{E}^c$ oriented opposite to the edges in B matrix, see (4.18), followed by $\tilde{r}_k^{ji}$, where $(j,i) \in \mathcal{E}^c$ are oriented in the same direction as the edges in B matrix. In the sequel, let $\mathbb{1}$ denote a vector of ones of appropriate dimension. Let $M = |\mathcal{A}^c|$ be the number of non-corrupted nodes. For graph $\mathfrak{G}(\mathcal{A}^c)$, define its oriented incidence matrix B (dimension $M \times |\mathcal{E}^c|$) (see [78] for definition and details) as follows,

$$B_{ij} = \begin{cases} 1 & \text{if node } i \text{ is head of edge } j, \\ -1 & \text{if node } i \text{ is tail of edge } j, \\ 0 & \text{otherwise,} \end{cases} \tag{4.18}$$

for each edge $j \in \mathcal{E}^c$.

The adjacency matrix A (dimension $M \times M$), degree matrix D (diagonal matrix with dimension $M \times M$),

and the normalized graph Laplacian matrix L (dimension $M \times M$) as,

$$A_{ij} = \begin{cases} 1 & \text{if } (i,j) \text{ is edge in } \mathfrak{G}(\mathcal{A}^c), \\ 0 & \text{otherwise,} \end{cases} \tag{4.19}$$

$$D = \text{diag}(A\mathbb{1}), \tag{4.20}$$

$$L = I - D^{-1/2} A D^{-1/2}. \tag{4.21}$$

Using the notation $z_+ \coloneqq \max\{z, 0\}$ and $z_- \coloneqq z_+ - z$ for a scalar z, define B_+ and B_- as the matrices obtained from B, applying the respective operator componentwise. That is, $B_+ = [(B_{ij})_+]$ and $B_- = [(B_{ij})_-]$. We will use this notation to rewrite (4.16) - (4.17) as a system of linear equations.

First, recall (4.16) for each $i \in \mathcal{A}^c$,

$$\sum_{j \in \mathcal{N}_i \cap \mathcal{A}^c} \tilde{r}_k^{ji} = \frac{1}{\alpha_k \delta} \sum_{j \in \mathcal{N}_{\pi(i)}} W_{\pi(i)j} (v_k^j + \alpha_k r_k^{j\pi(i)}) - \frac{1}{\alpha_k \delta} \sum_{j \in \mathcal{N}_i} W_{ij} v_k^{\pi(j)} - \sum_{j \in \mathcal{N}_i \cap \mathcal{A}} \tilde{r}_k^{ji}. \tag{4.22}$$

Observe, that the left-hand side of the above expression is a linear combination (addition) of $\tilde{r}_k^{ji}$ for all $j \in \mathcal{N}_i \cap \mathcal{A}^c$ with $i \in \mathcal{A}^c$. These are elements of unknown vector γ. The right-hand side has three summation terms; the first term is exactly known from the first execution, the second term is exactly known given permutation π and first execution, and the third term is known following $\tilde{r}_k^{ji} = r_k^{ji}$ (for $i \in \mathcal{A}^c$ and $j \in \mathcal{N}_i \cap \mathcal{A}$) from (4.13). Consequently, the right-hand side for each constraint in (4.22) is exactly known for each of the M constraints, one for each $i \in \mathcal{A}^c$, and stored in vector ξ^1. The left-hand side for each linear constraint involves perturbations received by $i \in \mathcal{A}^c$ consequently they can be encoded using unsigned adjacency matrix as $[B_- \quad B_+]\,\gamma$. We can rewrite M linear equations, given by (4.22) for each $i \in \mathcal{A}^c$, as

$$[B_- \quad B_+]\,\gamma = \xi^1.$$

Next, recall (4.17) for each $i \in \mathcal{A}^c$,

$$\sum_{j \in \mathcal{N}_i \cap \mathcal{A}^c} \tilde{r}_k^{ij} = - \sum_{j \in \mathcal{N}_i \cap \mathcal{A}} \tilde{r}_k^{ij}. \tag{4.23}$$

As before, the left-hand side is a linear combination (addition) of $\tilde{r}_k^{ij}$, with $i \in \mathcal{A}^c$ and $j \in \mathcal{N}_i \cap \mathcal{A}^c$. The right-hand side is a combination of $\tilde{r}_k^{ij}$ for $i \in \mathcal{A}^c$ and $j \in \mathcal{N}_i \cap \mathcal{A}$, which are known given first execution and permutation π as per (4.14). The left-hand side for each linear constraint involves perturbations transmitted by $i \in \mathcal{A}^c$ consequently they can be encoded using unsigned adjacency matrix as $[B_+ \quad B_-]\,\gamma$. We can rewrite M constraints, one linear equation as given above in (4.23) for each $i \in \mathcal{A}^c$, as a system of linear equations given by,

$$[B_+ \quad B_-]\,\gamma = \xi^2,$$

where ξ^2 represents the stacked right-hand side residuals for each constraint (4.23).

We define matrix T and vector ξ as follows,

$$T = \begin{pmatrix} B_- & B_+ \\ B_+ & B_- \end{pmatrix} \text{ and } \xi = \begin{pmatrix} \xi^1 \\ \xi^2 \end{pmatrix}. \tag{4.24}$$

The stacked linear equations (4.22) and (4.23) are rewritten using matrix T and vector ξ as,

$$\begin{pmatrix} B_- & B_+ \\ B_+ & B_- \end{pmatrix} \gamma = \begin{pmatrix} \xi^1 \\ \xi^2 \end{pmatrix} \implies T\,\gamma = \xi, \tag{4.25}$$

where T is defined above as per (4.24), γ is the vector of unknown $\tilde{r}^{ij}$ (i and j both belong to $\mathcal{A}^c$), and ξ is formed using the residual terms from (4.23) - (4.22) as per (4.24). The expression $T\gamma = \xi$ brings out the fact that unknown $\tilde{r}^{ij}$ ($i, j \in \mathcal{A}^c$) are constrained to be in a linear sub-space characterized by linear equations given by $T\gamma = \xi$.

From the Rouché-Capelli Theorem, if rank (T) = rank $([T|\xi])$ then the system of linear equations has at least one solution. We will prove that

$$\text{rank } T = \text{rank } (T \mid \xi) = 2M - 1, \tag{4.26}$$

to show that $T\gamma = \xi$ admits at least one solution. This will imply that there exists at least one set of $\tilde{\boldsymbol{r}}$ that lead to $\mathsf{E}(\tilde{F}, \tilde{\boldsymbol{r}}, \chi)$ and $\mathsf{E}(F, \boldsymbol{r}, \chi)$ being exactly the same from A's perspective. In what follows, we show graph conditions in Theorem 7 allow us to directly prove the above statement leading to the proof of sufficiency of graph conditions for privacy.

First, notice

$$(\mathbb{1}^\intercal \mid -\mathbb{1}^\intercal) \begin{pmatrix} B_- & B_+ \\ B_+ & B_- \end{pmatrix} = \left(-\mathbb{1}^\intercal B \mid \mathbb{1}^\intercal B\right) = 0_{1\times 2M}, \tag{4.27}$$

proving that rows of T are not linearly independent. This directly gives us that rank $T \leq 2M - 1$.

Next, we show that rank $T \geq 2M - 1$. To that end, we have

$$\begin{aligned} \text{rank } T &\overset{(a)}{=} \text{rank } (TT^\intercal) \\ &\overset{(b)}{=} \text{rank } \begin{pmatrix} B_-B_-^\intercal + B_+B_+^\intercal & B_-B_+^\intercal + B_+B_-^\intercal \\ B_+B_-^\intercal + B_-B_+^\intercal & B_-B_-^\intercal + B_+B_+^\intercal \end{pmatrix} \end{aligned}$$

$$
\begin{aligned}
&\overset{(c)}{=} \text{rank } \begin{pmatrix} D & A \\ A & D \end{pmatrix} \\
&\overset{(d)}{=} \text{rank } D + \text{rank } (D - AD^{-1}A) \\
&\overset{(e)}{=} M + \text{rank } (I_M - D^{-1/2}AD^{-1}AD^{-1/2}) \\
&\overset{(f)}{=} M + \text{rank } (I_M - (I_M - L)^2) \\
&\overset{(g)}{=} M + \text{rank } (2L - L^2) \\
&\overset{(h)}{\geq} M + \underbrace{\text{rank } L}_{=M-1} + \text{rank } (2I_M - L) - M \\
&\overset{(i)}{=} M - 1 + \text{rank } (2I_M - L) \\
&= 2M - 1,
\end{aligned} \tag{4.28}
$$

where I_M is the $M \times M$ identity matrix.

Here, (a) follows from the fact that if T is a matrix of real numbers then rank (T) = rank $(T^\intercal)$ = rank $(TT^\intercal)$ = rank $(T^\intercal T)$. Next, we use the definition of T from (4.25) to get (b). We use the definition of B_+, B_-, D, A to get (c). We use Guttman rank additivity formula to get (d). Specifically if

$$
A = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix},
$$

then the Guttman rank additivity formula gives, rank (A) = rank (A_4)+rank $(A_1 - A_2A_4^{-1}A_3)$, and applying it to our situation we get rank (T) = rank (D) + rank $(D - AD^{-1}A)$ as shown in (d).

Recall, D is a diagonal matrix of size $M \times M$ (see (4.20)) and consequently D is rank M (full rank). We use the fact that for any matrix Y, rank (DY) = rank (YD) = rank (Y), to get (e) as shown below,

$$
\begin{aligned}
\text{rank } (D - AD^{-1}A) &= \text{rank } (D^{1/2}I_MD^{1/2} - AD^{-1}A) \\
&= \text{rank } (D^{1/2}\left(I_M - D^{-1/2}AD^{-1}AD^{-1/2}\right)D^{1/2}) \\
&\overset{(*)}{=} \text{rank } (D^{1/2}\left(I_M - D^{-1/2}AD^{-1}AD^{-1/2}\right)) \\
&\overset{(*)}{=} \text{rank } \left(I_M - D^{-1/2}AD^{-1}AD^{-1/2}\right),
\end{aligned}
$$

where the last two equality relations, labeled as (*), follow from the fact that D is a full rank matrix.

Next, we use the definition $L = I_M - D^{-1/2}AD^{-1/2}$ from (4.21) along with following algebraic manipulation,

$$
\begin{aligned}
I_M - (I_M - L)^2 &= I_M - I_M - L^2 + 2L \\
&= 2L - L^2 \\
&= 2(I_M - D^{-1/2}AD^{-1/2}) - \left(I_M - D^{-1/2}AD^{-1/2}\right)^2 \\
&= 2(I_M - D^{-1/2}AD^{-1/2}) - \left(I_M - 2D^{-1/2}AD^{-1/2} + D^{-1/2}AD^{-1}AD^{-1/2}\right) \\
&= I_M - D^{-1/2}AD^{-1}AD^{-1/2},
\end{aligned}
$$

to get rank $(I_M-(I_M-L)^2)$ = rank $(I_M-D^{-1/2}AD^{-1}AD^{-1/2})$, i.e. (f). Moreover, we also see rank $(I_M - D^{-1/2}AD^{-1}AD^{-1/2})$ = rank $(2L-L^2)$ following the above analysis giving us equality (g).

We use Sylvester's rank inequality to get,

$$
\text{rank } ((2I_M - L)L) \geq \text{rank } (2I_M - L) + \text{rank } (L) - M,
$$

allowing us to show (h). We use the fact that the normalized graph Laplace matrix (see definition (4.21)) is rank deficient by 1 for connected graphs (as for our graph $\mathfrak{G}(\mathcal{A}^c)$). Moreover, since $\mathfrak{G}(\mathcal{A}^c)$ is not bipartite, the eigenvalues of L are strictly less than 2, according to Lemma 1.7 in [105]. This allows us to conclude that $2I_M - L$ does not have any zero eigenvalues and rank $(2I_M - L) = M$ and we get (i). Thus, (4.27) and (4.28) together yield rank $T = 2M - 1$.

For the augmented matrix $(T \mid \xi)$, we have

$$
2M - 1 \;\leq\; \text{rank } (T \mid \xi) \;\leq\; 2M. \tag{4.29}
$$

In the above relation, the inequality on the left follows from our earlier proof that rank $T = 2M - 1$. The one on the right follows from the fact that the augmented matrix has $2M$ rows. We demonstrate that rows of $(T \mid \xi)$ are linearly dependent to conclude (4.26). From (4.27), we deduce

$$
\left(\mathbb{1}^\intercal \mid \; -\mathbb{1}^\intercal\right)(T \mid \xi) = \left(0 \mid \mathbb{1}^\intercal\xi^1 - \mathbb{1}^\intercal\xi^2\right).
$$

Now, we show $\mathbb{1}^T\xi^1 - \mathbb{1}^T\xi^2 = 0$ to conclude the proof. In the following, $|\mathcal{Z}|$ represents the cardinality of a set $\mathcal{Z}$.

$$
\begin{aligned}
&\mathbb{1}^T\xi^1 - \mathbb{1}^T\xi^2 \\
&= \frac{1}{\alpha_k\delta}\sum_{i\in\mathcal{A}^c}\sum_{j\in\mathcal{N}_{\pi(i)}} W_{\pi(i)j}(v_k^j + \alpha_k r_k^{j\pi(i)}) - \frac{1}{\alpha_k\delta}\sum_{i\in\mathcal{A}^c}\sum_{j\in\mathcal{N}_i} W_{ij}v_k^{\pi(j)} - \sum_{i\in\mathcal{A}^c}\sum_{j\in\mathcal{N}_i\cap\mathcal{A}}\tilde{r}_k^{ji} + \sum_{i\in\mathcal{A}^c}\sum_{j\in\mathcal{N}_i\cap\mathcal{A}}\tilde{r}_k^{ij} \\
&= \underbrace{\frac{1}{\alpha_k\delta}\sum_{i\in\mathcal{A}^c}\left[\sum_{j\in\mathcal{N}_{\pi(i)}} W_{\pi(i)j}v_k^j - \sum_{j\in\mathcal{N}_i} W_{ij}v_k^{\pi(j)}\right]}_{:=Q^1} + \underbrace{\sum_{i\in\mathcal{A}^c}\sum_{j\in\mathcal{N}_{\pi(i)}} r_k^{j\pi(i)} - \sum_{i\in\mathcal{A}^c}\sum_{j\in\mathcal{N}_i\cap\mathcal{A}}\tilde{r}_k^{ji} + \sum_{i\in\mathcal{A}^c}\sum_{j\in\mathcal{N}_i\cap\mathcal{A}}\tilde{r}_k^{ij}}_{:=Q^2} \\
&= Q^1 + Q^2,
\end{aligned}
$$

where we have used $r_k^{ii} = 0$ and $W_{ij} = \delta$ for $(i,j) \in \mathcal{E}$ and $i \neq j$. Utilizing $\pi(i) = i$, for all $i \neq I, J$, we can simplify Q^1 as,

$$
\begin{aligned}
\alpha_k Q^1 &= \frac{1}{\delta} \sum_{i \in \mathcal{A}^c} \left[\sum_{j \in \mathcal{N}_{\pi(i)}} W_{\pi(i)j} v_k^j - \sum_{j \in \mathcal{N}_i} W_{ij} v_k^{\pi(j)} \right] \\
&= \frac{1}{\delta} \sum_{i \in \mathcal{A}^c} \sum_{j \in \mathcal{N}_{\pi(i)}} W_{\pi(i)j} v_k^j - \frac{1}{\delta} \sum_{i \in \mathcal{A}^c} \sum_{j \in \mathcal{N}_i} W_{ij} v_k^{\pi(j)} \\
&= \frac{1}{\delta} \left(\underbrace{\sum_{i \in \mathcal{A}^c \setminus \{I,J\}} \sum_{j \in \mathcal{N}_i} W_{ij} v_k^j + \overbrace{\sum_{j \in \mathcal{N}_J} W_{Jj} v_k^j}^{i=I} + \overbrace{\sum_{j \in \mathcal{N}_I} W_{Ij} v_k^j}^{i=J}}_{\sum_{i \in \mathcal{A}^c} \sum_{j \in \mathcal{N}_i} W_{ij} v_k^j} \right) - \frac{1}{\delta} \sum_{i \in \mathcal{A}^c} \sum_{j \in \mathcal{N}_i} W_{ij} v_k^{\pi(j)} \\
&= \frac{1}{\delta} \sum_{i \in \mathcal{A}^c} \sum_{j \in \mathcal{N}_i} W_{ij} v_k^j - \frac{1}{\delta} \sum_{i \in \mathcal{A}^c} \sum_{j \in \mathcal{N}_i} W_{ij} v_k^{\pi(j)} \\
&= \frac{1}{\delta} \sum_{i \in \mathcal{A}^c} \left[\sum_{j \in \mathcal{N}_i \cap \mathcal{A}} W_{ij} v_k^j - \sum_{j \in \mathcal{N}_i \cap \mathcal{A}} W_{ij} v_k^{\pi(j)} \right] + \frac{1}{\delta} \sum_{i \in \mathcal{A}^c} \left[\sum_{j \in \mathcal{N}_i \cap \mathcal{A}^c} W_{ij} v_k^j - \sum_{j \in \mathcal{N}_i \cap \mathcal{A}^c} W_{ij} v_k^{\pi(j)} \right] \\
&\overset{(a)}{=} \frac{1}{\delta} \sum_{i \in \mathcal{A}^c} \left[\sum_{j \in \mathcal{N}_i \cap \mathcal{A}^c} W_{ij} v_k^j - \sum_{j \in \mathcal{N}_i \cap \mathcal{A}^c} W_{ij} v_k^{\pi(j)} \right] \\
&= \frac{1}{\delta} \sum_{i \in \mathcal{A}^c} \left[\left(W_{ii} v_k^i + \sum_{j \in \mathcal{N}_i \cap \mathcal{A}^c \setminus \{i\}} W_{ij} v_k^j \right) - \left(W_{ii} v_k^{\pi(i)} + \sum_{j \in \mathcal{N}_i \cap \mathcal{A}^c \setminus \{i\}} W_{ij} v_k^{\pi(j)} \right) \right] \\
&= \frac{1}{\delta} \sum_{i \in \mathcal{A}^c} \left[\left(1 - (|\mathcal{N}_i| - 1)\delta\right) v_k^i + \sum_{j \in \mathcal{N}_i \cap \mathcal{A}^c \setminus \{i\}} \delta v_k^j \right] \\
&\qquad - \frac{1}{\delta} \sum_{i \in \mathcal{A}^c} \left[\left(1 - (|\mathcal{N}_i| - 1)\delta\right) v_k^{\pi(i)} + \sum_{j \in \mathcal{N}_i \cap \mathcal{A}^c \setminus \{i\}} \delta v_k^{\pi(j)} \right] \\
&= \sum_{i \in \mathcal{A}^c} \left[\left(\frac{1}{\delta} - |\mathcal{N}_i| + 1 \right) \left(v_k^i - v_k^{\pi(i)} \right) + \sum_{j \in \mathcal{N}_i \cap \mathcal{A}^c \setminus \{i\}} \left(v_k^j - v_k^{\pi(j)} \right) \right] \\
&= \left(|\mathcal{N}_I \cap \mathcal{A}^c \setminus \{I\}| + \frac{1}{\delta} - |\mathcal{N}_I| + 1 \right) \left(v_k^I - v_k^J \right) + \left(|\mathcal{N}_J \cap \mathcal{A}^c \setminus \{J\}| + \frac{1}{\delta} - |\mathcal{N}_J| + 1 \right) \left(v_k^J - v_k^I \right) \\
\alpha_k Q^1 &= \left(|\mathcal{N}_I \cap \mathcal{A}^c \setminus \{I\}| - |\mathcal{N}_I| \right) \left(v_k^I - v_k^J \right) + \left(|\mathcal{N}_J \cap \mathcal{A}^c \setminus \{J\}| - |\mathcal{N}_J| \right) \left(v_k^J - v_k^I \right),
\end{aligned}
$$

where (a) follows from the fact that $v_k^j = v_k^{\pi(j)}, \forall j \in \mathcal{A}$.

Next, we simplify Q^2 as

$$
\begin{aligned}
Q^2 &= \sum_{i\in\mathcal{A}^c}\sum_{j\in\mathcal{N}_{\pi(i)}} r_k^{j\pi(i)} - \sum_{i\in\mathcal{A}^c}\sum_{j\in\mathcal{N}_i\cap\mathcal{A}} \tilde{r}_k^{ji} + \sum_{i\in\mathcal{A}^c}\sum_{j\in\mathcal{N}_i\cap\mathcal{A}} \tilde{r}_k^{ij} \\
&= \Bigg(\underbrace{\sum_{i\in\mathcal{A}^c\setminus\{I,J\}}\sum_{j\in\mathcal{N}_{\pi(i)}} r_k^{j\pi(i)} + \overbrace{\sum_{j\in\mathcal{N}_J} r_k^{jJ}}^{i=I} + \overbrace{\sum_{j\in\mathcal{N}_I} r_k^{jI}}^{i=J}}_{\sum_{i\in\mathcal{A}^c}\sum_{j\in\mathcal{N}_i} r_k^{ji}} \Bigg) - \sum_{i\in\mathcal{A}^c}\sum_{j\in\mathcal{N}_i\cap\mathcal{A}} \tilde{r}_k^{ji} + \sum_{i\in\mathcal{A}^c}\sum_{j\in\mathcal{N}_i\cap\mathcal{A}} \tilde{r}_k^{ij} \\
&= \sum_{i\in\mathcal{A}^c}\sum_{j\in\mathcal{N}_i} r_k^{ji} - \sum_{i\in\mathcal{A}^c}\sum_{j\in\mathcal{N}_i\cap\mathcal{A}} \tilde{r}_k^{ji} + \sum_{i\in\mathcal{A}^c}\sum_{j\in\mathcal{N}_i\cap\mathcal{A}} \tilde{r}_k^{ij} \\
&= \sum_{i\in\mathcal{A}^c}\sum_{j\in\mathcal{N}_i\cap\mathcal{A}^c} r_k^{ji} + \underbrace{\sum_{i\in\mathcal{A}^c}\sum_{j\in\mathcal{N}_i\cap\mathcal{A}} r_k^{ji} - \sum_{i\in\mathcal{A}^c}\sum_{j\in\mathcal{N}_i\cap\mathcal{A}} \tilde{r}_k^{ji}}_{=0 \text{ as per (4.13)}} + \sum_{i\in\mathcal{A}^c}\sum_{j\in\mathcal{N}_i\cap\mathcal{A}} \tilde{r}_k^{ij} \\
&\overset{(a)}{=} \sum_{i\in\mathcal{A}^c}\sum_{j\in\mathcal{N}_i\cap\mathcal{A}^c} r_k^{ji} + \sum_{i\in\mathcal{A}^c}\sum_{j\in\mathcal{N}_i\cap\mathcal{A}} \tilde{r}_k^{ij} \\
&\overset{(b)}{=} \sum_{i\in\mathcal{A}^c}\sum_{j\in\mathcal{N}_i\cap\mathcal{A}^c} r_k^{ji} + \sum_{i\in\mathcal{A}^c}\sum_{j\in\mathcal{N}_i\cap\mathcal{A}} \left[\frac{1}{\alpha_k}(v_k^i - v_k^{\pi(i)}) + r_k^{ij}\right] \\
&= \sum_{i\in\mathcal{A}^c}\left(\sum_{j\in\mathcal{N}_i\cap\mathcal{A}^c} r_k^{ji} + \sum_{j\in\mathcal{N}_i\cap\mathcal{A}} r_k^{ij}\right) + \sum_{i\in\mathcal{A}^c}\sum_{j\in\mathcal{N}_i\cap\mathcal{A}} \left[\frac{1}{\alpha_k}(v_k^i - v_k^{\pi(i)})\right] \\
&\overset{(c)}{=} \underbrace{\sum_{i\in\mathcal{A}^c}\left(\sum_{j\in\mathcal{N}_i} r_k^{ij}\right)}_{=0} + \sum_{i\in\mathcal{A}^c}\sum_{j\in\mathcal{N}_i\cap\mathcal{A}} \left[\frac{1}{\alpha_k}(v_k^i - v_k^{\pi(i)})\right] \\
&\overset{(d)}{=} \frac{1}{\alpha_k}\sum_{i\in\mathcal{A}^c} |\mathcal{N}_i\cap\mathcal{A}|(v_k^i - v_k^{\pi(i)}) \\
Q^2 &= \frac{1}{\alpha_k}|\mathcal{N}_I\cap\mathcal{A}|\left(v_k^I - v_k^J\right) + \frac{1}{\alpha_k}|\mathcal{N}_J\cap\mathcal{A}|\left(v_k^J - v_k^I\right).
\end{aligned}
$$

Here, (a) follows from $r_k^{ji} = \tilde{r}_k^{ji}$ for all $j \in \mathcal{A}$ from (4.13). The equality in (b) follows from (4.14) and (4.15), (c) from (4.17), and (d) from the properties of permutation π. Combining the expressions for Q^1 and Q^2, we get

$$
\begin{aligned}
Q^1 + Q^2 &= \frac{1}{\alpha_k}\left[|\mathcal{N}_I\cap\mathcal{A}^c\setminus\{I\}| - |\mathcal{N}_I|\right]\left(v_k^I - v_k^J\right) + \frac{1}{\alpha_k}\left[|\mathcal{N}_J\cap\mathcal{A}^c\setminus\{J\}| - |\mathcal{N}_J|\right]\left(v_k^J - v_k^I\right) \\
&\qquad + \frac{1}{\alpha_k}|\mathcal{N}_I\cap\mathcal{A}|\left(v_k^I - v_k^J\right) + \frac{1}{\alpha_k}|\mathcal{N}_J\cap\mathcal{A}|\left(v_k^J - v_k^I\right) \\
&= 0,
\end{aligned}
$$

where the last line leverages the relation,

$$|\mathcal{N}_i\cap\mathcal{A}| + |\mathcal{N}_i\cap\mathcal{A}^c\setminus\{i\}| = |\mathcal{N}_i| - 1,$$

for $i = I, J$. This completes the proof of privacy of our algorithm. $\square$

4.6 Convergence Analysis and Correctness

Recall, $x_* = (x_*^1, \ldots, x_*^n)$ is the Nash equilibrium action profile for players $i = 1, \ldots, n$. Also, following the definition of a Nash equilibrium, we know that $x_* = (x_*^1, \ldots, x_*^n)$ is a fixed point for the gradient based update equation, i.e. $x_*^i = \text{proj}_{\mathcal{X}^i}[x_*^i - \alpha_k \nabla_{x^i} f_i(x_*^i, \overline{x}_*)]$.

The non-expansiveness of the projection operator yields,

$$\begin{aligned}
\|x_{k+1}^i - x_*^i\|^2 &= \|\text{proj}_{\mathcal{X}^i}[x_k^i - \alpha_k \nabla_{x^i} f_i(x_k^i, n\widehat{v}_k^i)] - x_*^i\|^2 \\
&= \|\text{proj}_{\mathcal{X}^i}[x_k^i - \alpha_k \nabla_{x^i} f_i(x_k^i, n\widehat{v}_k^i)] - \text{proj}_{\mathcal{X}^i}[x_*^i - \alpha_k \nabla_{x^i} f_i(x_*^i, \overline{x}_*)]\|^2 \\
&\leq \|x_k^i - x_*^i - \alpha_k(\nabla_{x^i} f_i(x_k^i, n\widehat{v}_k^i) - \nabla_{x^i} f_i(x_*^i, \overline{x}_*))\|^2 \\
&= \|x_k^i - x_*^i\|^2 + \underbrace{\alpha_k^2 \|\nabla_{x^i} f_i(x_k^i, n\widehat{v}_k^i) - \nabla_{x^i} f_i(x_*^i, \overline{x}_*)\|^2}_{:=T_1^i} \\
&\qquad - \underbrace{2\alpha^k (\nabla_{x^i} f_i(x_k^i, n\widehat{v}_k^i) - \nabla_{x^i} f_i(x_*^i, \overline{x}_*))^\intercal (x_k^i - x_*^i)}_{:=T_2^i}. \qquad (4.30)
\end{aligned}$$

Owing to the compactness of $\mathcal{X}$'s, gradients $\nabla_{x^i} f_i$ are bounded. Such a bound, together with Triangle inequality, yields an upper bound on T_1^i as follows,

$$\begin{aligned}
T_1^i &= \alpha_k^2 \|\nabla_{x^i} f_i(x_k^i, n\widehat{v}_k^i) - \nabla_{x^i} f_i(x_*^i, \overline{x}_*)\|^2 \\
&\overset{(a)}{\leq} \alpha_k^2 \left(\|\nabla_{x^i} f_i(x_k^i, n\widehat{v}_k^i)\| + \|\nabla_{x^i} f_i(x_*^i, \overline{x}_*)\|\right)^2 \\
&\overset{(b)}{\leq} \alpha_k^2 C^2, \qquad (4.31)
\end{aligned}$$

where (a) follows from Triangle inequality and (b) follows from the fact that $\|\nabla_{x^i} f_i\|$ is bounded.

We define the following two quantities

$$y_k := \frac{1}{n} \sum_{i=1}^n v_k^i \text{ and } C' := \max_i \max_{x \in \mathcal{X}^i} \|x - x_*^i\|, \qquad (4.32)$$

and bound T_2^i as,

$$
\begin{aligned}
T_2^i &= 2\alpha_k(\nabla_{x^i} f_i(x_k^i, n\widehat{v}_k^i) - \nabla_{x^i} f_i(x_*^i, \overline{x}_*))^\intercal (x_k^i - x_*^i) \\
&\overset{(a)}{=} 2\alpha_k(\nabla_{x^i} f_i(x_k^i, n\widehat{v}_k^i) - \nabla_{x^i} f_i(x_k^i, ny_k) + \nabla_{x^i} f_i(x_k^i, ny_k) - \nabla_{x^i} f_i(x_*^i, \overline{x}_*))^\intercal (x_k^i - x_*^i) \\
&\overset{(b)}{=} 2\alpha_k \left[\nabla_{x^i} f_i(x_k^i, n\widehat{v}_k^i) - \nabla_{x^i} f_i(x_k^i, ny_k)\right]^\intercal (x_k^i - x_*^i) \\
&\qquad + 2\alpha_k \left[\nabla_{x^i} f_i(x_k^i, ny_k) - \nabla_{x^i} f_i(x_*^i, \overline{x}_*)\right]^\intercal (x_k^i - x_*^i) \\
&\overset{(c)}{\geq} -2\alpha_k \|\nabla_{x^i} f_i(x_k^i, ny_k) - \nabla_{x^i} f_i(x_k^i, n\widehat{v}_k^i)\| \, \|x_k^i - x_*^i\| \\
&\qquad + 2\alpha_k \left[\nabla_{x^i} f_i(x_k^i, ny_k) - \nabla_{x^i} f_i(x_*^i, \overline{x}_*)\right]^\intercal (x_k^i - x_*^i) \\
&\overset{(d)}{\geq} -2\alpha_k \overline{L} n \|y_k - \widehat{v}_k^i\| \, \|x_k^i - x_*^i\| + 2\alpha^k \left[\nabla_{x^i} f_i(x_k^i, ny_k) - \nabla_{x^i} f_i(x_*^i, \overline{x}_*)\right]^\intercal (x_k^i - x_*^i) \\
&\overset{(e)}{\geq} -2\alpha_k \overline{L} n C' \|y_k - \widehat{v}_k^i\| + 2\alpha_k \left[\nabla_{x^i} f_i(x_k^i, ny_k) - \nabla_{x^i} f_i(x_*^i, \overline{x}_*)\right]^\intercal (x_k^i - x_*^i).
\end{aligned} \tag{4.33}
$$

In the above expression, (a) follows from adding and subtracting $\nabla_{x^i} f_i(x_k^i, ny_k)$ term and (b) is the expansion of the inner product. We use Cauchy-Schwarz inequality to get (c) and use the Lipschitz continuity of $\nabla_{x^i} f_i(x_k^i, \cdot)$ to get (d). Finally, (e) follows from the definition of C' from (4.32). To further simplify the bounds on T_2^i, we show that $ny_k = \overline{x}_k$ using induction as follows. For $k = 0$, the relation follows from $v_0^i = x_0^i$. Assume that it holds for $k = 0, \ldots, K$, i.e., $ny_K = \overline{x}_K$. Then, we have

$$
\begin{aligned}
ny_{K+1} &= \sum_{i=1}^n v_{K+1}^i \\
&\overset{(a)}{=} \sum_{i=1}^n \left(\widehat{v}_K^i + x_{K+1}^i - x_K^i\right) \\
&\overset{(b)}{=} \sum_{i=1}^n \left[\sum_{j=1}^n W_{ij} \left(v_K^j + \alpha_K r_K^{ji}\right) + x_{K+1}^i - x_K^i\right] \\
&\overset{(c)}{=} \sum_{j=1}^n \underbrace{\sum_{i=1}^n W_{ij}}_{=1} v_K^j + \delta\alpha_K \sum_{j=1}^n \underbrace{\sum_{i\in\mathcal{N}_j}^n r_K^{ji}}_{=0} + \overline{x}_{K+1} - \overline{x}_K \\
&\overset{(d)}{=} ny_K + \overline{x}_{K+1} - \overline{x}_K \\
&= \overline{x}_{K+1},
\end{aligned} \tag{4.34}
$$

where (a) follows from (4.12), (b) from (4.9), (c) from the doubly stochastic nature of W and (4.8). Finally, (d) follows from the induction hypothesis.

Substitute $ny_k = \overline{x}_k$ in (4.30) and combine that with (4.31) and (4.33). The result, can be written as

$$
\begin{aligned}
\|x_{k+1}^i - x_*^i\|^2 \leq \|x_k^i - x_*^i\|^2 &+ \alpha_k^2 C^2 + 2\alpha_k \overline{L} n C' \|y_k - \widehat{v}_k^i\| \\
&- 2\alpha_k(\nabla_{x^i} f_i(x_k^i, n\widehat{v}_k^i) - \nabla_{x^i} f_i(x_*^i, \overline{x}_*))^\intercal (x_k^i - x_*^i).
\end{aligned} \tag{4.35}
$$

We sum the expression (4.35) over $i \in \mathcal{V}$ to get,

$$\begin{aligned}
\sum_{i=1}^{n} \|x_{k+1}^i - x_*^i\|^2 &\leq \sum_{i=1}^{n} \|x_k^i - x_*^i\|^2 + \alpha_k^2 nC^2 + 2\alpha_k \overline{L} nC' \sum_{i=1}^{n} \|y_k - \widehat{v}_k^i\| \\
&\quad - 2\alpha_k \sum_{i=1}^{n} (\nabla_{x^i} f_i(x_k^i, n\widehat{v}_k^i) - \nabla_{x^i} f_i(x_*^i, \overline{x}_*))^\mathsf{T} (x_k^i - x_*^i) \\
\|x_{k+1} - x_*\|^2 &\overset{(a)}{\leq} \|x_k - x_*\|^2 + \alpha_k^2 nC^2 + 2\alpha_k n\overline{L}C' \sum_{i=1}^{n} \|y_k - \widehat{v}_k^i\| \\
&\quad - 2\sum_{i=1}^{n} \alpha_k \left[\nabla_{x^i} f_i(x_k^i, \overline{x}_k) - \nabla_{x^i} f_i(x_*^i, \overline{x}_*)\right]^\mathsf{T} (x_k^i - x_*^i) \\
&\overset{(b)}{=} \|x_k - x_*\|^2 + \underbrace{\alpha_k^2 nC^2}_{U_k^1} + \underbrace{2\alpha_k n\overline{L}C' \sum_{i=1}^{n} \|y_k - \widehat{v}_k^i\|}_{U_k^2} \\
&\quad - \underbrace{2\alpha_k \left[\phi(x_k) - \phi(x_*)\right]^\mathsf{T} (x_k - x_*)}_{U_k^3}. \qquad (4.36)
\end{aligned}$$

In the above expression (4.36), we get inequality (a) using a few simple relations, $\|x_{k+1} - x_*\|^2 = \sum_{i=1}^n \|x_{k+1}^i - x_*^i\|^2$ and $\|x_k - x_*\|^2 = \sum_{i=1}^n \|x_k^i - x_*^i\|^2$ (property of norm). We prove equality (b), using the definition of $\phi(x)$ in (4.4) and the expansion of inner product. Specifically,

$$(\phi(x_k) - \phi(x_*))^\mathsf{T}(x_k - x_*) = \sum_{i=1}^{n} \left[\left(\nabla_{x^i} f_i(x_k^i, \overline{x}_k) - \nabla_{x^i} f_i(x_*^i, \overline{x}_*)\right)^\mathsf{T} (x_k^i - x_*^i)\right].$$

Observe that (4.36) has the same structure as the dissipation inequality in the convergence result for non-negative, almost supermartingales (cf. Lemma 13 below, reproduced from [75]).

Lemma 13 (Theorem 1, [75])**.** *Let $\{F_k\}$, $\{E_k\}, \{G_k\}$ and $\{H_k\}$, be non-negative, real sequences. Assume that $\sum_{k=0}^{\infty} F_k < \infty$, and $\sum_{k=0}^{\infty} H_k < \infty$ and*

$$E_{k+1} \leq (1 + F_k)E_k - G_k + H_k.$$

Then, the sequence $\{E_k\}$ converges to a non-negative real number and $\sum_{k=0}^{\infty} G_k < \infty$.

Consider, $E_k = \|x_k - x_*\|^2$, $F_k = 0$, $G_k = U_k^3$, $H_k = U_k^1 + U_k^2$, where U_k^1, U_k^2 and U_k^3 are defined in (4.36). Observe that (4.36) can be rewritten using this notation to exactly achieve the dissipation inequality in Lemma 13. We will now show that E_k, G_k, H_k satisfy conditions presented in Lemma 13.

First, observe E_k is a norm of distance between local action estimate and Nash equilibrium and is non-negative by definition. Moreover, notice the assumption on strict monotonicity of $\phi(x)$ in (4.5) ensures $G_k = U_k^3$ is positive (non-negative). U_k^1 is a square term and hence non-negative and U_k^2 is a the sum of norms of several terms making them non-negative. As a result $H_k = U_k^1 + U_k^2$ is non-negative. Next, we need

to show $\sum_k H_k = \sum_k (U_k^1 + U_k^2) < \infty$. In order to show this, we bound one of the terms on the right-hand side of (4.36) using the next result.

Lemma 14. $\sum_{k=0}^{\infty} \alpha_k \sum_{i=1}^{n} \|y_k - \widehat{v}_k^i\| < \infty$, *for all* $i \in \mathcal{V}$.

The proof for this lemma relies on the doubly stochastic nature of W and properties of obfuscation sequences from (4.8). We provide a detailed proof in Section 4.8.

Recall, the learning step-sizes, α_k satisfy non-summability, i.e. $\sum_k \alpha_k = \infty$ and square summability $\sum_k \alpha_k^2 < \infty$ leads us to conclude $\sum_k U_k^1 < \infty$. Additionally, notice that Lemma 14 guarantees $\sum_k U_k^2 < \infty$. Together they give us $\sum_k H_k = \sum_k (U_k^1 + U_k^2) < \infty$.

Together conditions – (a) $\{E_k\}, \{H_k\}, \{G_k\}$ are non-negative sequences and (b) $\sum_k H_k = \sum_k U_k^1 + U_k^2 < \infty$, along with dissipation inequality (4.36) allow us to invoke Lemma 13 and infer that $\|x_k - x_*\|^2$ converges and

$$\sum_{k=0}^{\infty} \alpha_k \left[\phi(x_k) - \phi(x_*)\right]^{\intercal} (x_k - x_*) < \infty. \tag{4.37}$$

Next, we will show that together, $\|x_k - x_*\|^2$ converges and $\sum_{k=0}^{\infty} \alpha_k \left[\phi(x_k) - \phi(x_*)\right]^{\intercal} (x_k - x_*) < \infty$, allow us to show that x_k converges to x_*. We use the following steps to prove this.

1. Let $\Phi(x_k) \coloneqq \left[\phi(x_k) - \phi(x_*)\right]^{\intercal} (x_k - x_*)$, then $\liminf_{k\to\infty} \Phi(x_k) = 0$.
   - Recall from (4.5) that due to strict monotonicity of $\phi(x)$, we know that $\Phi(x_k) > 0$, for all $x_k \neq x^*$.
   - The learning rate α_k is non-summable, i.e., $\sum_k \alpha_k = \infty$, see (4.7).
   - Observe from (4.37) and the above two statements we have $\liminf_{k\to\infty} \Phi(x_k) = 0$.
2. There exists a sub-sequence $\{x_{k_\ell}\}_\ell$ of x_k such that $\lim_{\ell\to\infty} \Phi(x_{k_\ell}) = 0$.
   - The sequence $\{x_k\}_k$ lies within $\mathcal{X}$, a compact set, and hence x_k remains bounded.
   - We know from the Bolzano-Weierstraß Theorem that any bounded sequence has a convergent sub-sequence. We select a convergent sub-sequence $\{x_{k_\ell}\}_\ell$ of $\{x_k\}_k$ along which $\Phi(\cdot)$ goes to zero, i.e. $\lim_{\ell\to\infty} \Phi(x_{k_\ell}) = \liminf_{k\to\infty} \Phi(x_k) = 0$.
3. The sub-sequence $\{x_{k_\ell}\}_\ell$ converges to x_*.
   - Strict monotonicity of ϕ ensures $\Phi(x_k) > 0$ for all $x_k \neq x_*$ and $\Phi(x_*) = 0$. As $\lim_{\ell\to\infty} \Phi(x_{k_\ell}) = 0$, we know that $\lim_{\ell\to\infty} x_{k_\ell} = x_*$.
4. The sequence $\{x_k\}_k$ converges to x_*, i.e. $\lim_{k\to\infty} x_k = x_*$.
   - Recall, from invocation of Lemma 13 we know $\|x_k - x_*\|$ converges.
   - From point 3. presented above, we know that $\{\|x_{k_\ell} - x_*\|\}_\ell$ converges to zero.

- We know from Theorem 2.5.2 of [76], that sub-sequences of a convergent sequence converge to the same limit as the original sequence.
- As $\{\|x_{k_\ell} - x_*\|\}_\ell$ is a sub-sequence of $\{\|x_k - x_*\|\}_k$, we use the three statements described above to conclude that $\{\|x_k - x_*\|\}_k$ also converges to zero.
- That is, the sequence $\{x_k\}_k$ converges to x_* or equivalently $\lim_{k\to\infty} x_k = x_*$.

4.7 Conclusions

In this chapter, we considered aggregate games played by agents that communicate over a network, each with private information. We showed that distributed algorithms for equilibrium computation in the literature are not designed with privacy requirements in mind, and consequently leak private information about players against honest-but-curious adversaries. Our proposed algorithm for NE computation exploits correlated perturbations to obfuscate aggregate estimates shared over the network. The algorithm asymptotically converges to the Nash Equilibrium. If the graph connecting non-adversarial players is connected and not bipartite, we show that our algorithm protects private information of non-adversarial players.

4.8 Proof of Lemma 14

Let us consider

$$T_k^i = x_k^i - x_{k-1}^i, \tag{4.38}$$

and we observe

$$y_k = y_0 + \sum_{s=1}^{k} (y_s - y_{s-1}) = \frac{1}{n}\left(\sum_{i=1}^{n} v_0^i + \sum_{s=1}^{k}\sum_{j=1}^{n} T_s^j\right). \tag{4.39}$$

The first equality follows from telescoping series argument. The second equality follows from the fact that $y_k \triangleq \frac{1}{n}\sum_{i=1}^n v_k^i = \frac{1}{n}\sum_{i=1}^n x_k^i$, proved using induction argument in (4.34).

Next, we build a expression for $\widehat{v}_k^j$ using the aggregate update relation, (4.10) from Algorithm 5, and use it to get an expression for v_k^j. We consider

$$\mathcal{H}_s^i = \sum_{j=1}^{n} W_{ij} r_s^{ji}, \tag{4.40}$$

and observe that $\mathcal{H}_s^i$ is the effective noise/perturbation added by player i at iteration s. Recall, $T_k^i = x_k^i - x_{k-1}^i$

from (4.38). Next, we build the following equality,

$$
\begin{aligned}
v_{k+1}^i &\overset{(a)}{=} \widehat{v}_k^i + x_{k+1}^i - x_k^i \\
&\overset{(b)}{=} \widehat{v}_k^i + T_{k+1}^i \\
&\overset{(c)}{=} \sum_{j=1}^n W_{ij} v_k^{ji} + T_{k+1}^i \\
&\overset{(d)}{=} \sum_{j=1}^n W_{ij} (v_k^j + \alpha_k r_k^{ji}) + T_{k+1}^i \\
&\overset{(e)}{=} \sum_{j=1}^n W_{ij} v_k^j + \alpha_k \mathcal{H}_k^i + T_{k+1}^i \qquad (4.41) \\
&\overset{(f)}{=} \sum_{j=1}^n W_{ij} \left(\underbrace{\sum_{l=1}^n W_{jl} v_{k-1}^l + \alpha_{k-1} H_{k-1}^j + T_k^j}_{v_k^j \text{ from (4.41)}} \right) + \alpha_k \mathcal{H}_k^i + T_{k+1}^i \\
&= \sum_{j=1}^n W_{ij} \left(\sum_{l=1}^n W_{jl} v_{k-1}^l \right) + \alpha_{k-1} \sum_{j=1}^n W_{ij} H_{k-1}^j + \sum_{j=1}^n W_{ij} T_k^j + \alpha_k \mathcal{H}_k^i + T_{k+1}^i \\
&\overset{(g)}{=} \sum_{l=1}^n \left(\underbrace{\sum_{j=1}^n W_{ij} W_{jl}}_{W_{il}^2} \right) v_{k-1}^l + \alpha_{k-1} \sum_{j=1}^n W_{ij} H_{k-1}^j + \sum_{j=1}^n W_{ij} T_k^j + \alpha_k \mathcal{H}_k^i + T_{k+1}^i \\
&\overset{(h)}{=} \sum_{l=1}^n W_{il}^2 v_{k-1}^l + \alpha_{k-1} \sum_{j=1}^n W_{ij} H_{k-1}^j + \sum_{j=1}^n W_{ij} T_k^j + \alpha_k \mathcal{H}_k^i + T_{k+1}^i \\
&\overset{(i)}{=} \sum_{l=1}^n W_{il}^2 \left(\underbrace{\sum_{t=1}^n W_{lt} v_{k-2}^t + \alpha_{k-2} H_{k-2}^l + T_{k-1}^l}_{v_{k-1}^l \text{ from (4.41)}} \right) + \alpha_{k-1} \sum_{j=1}^n W_{ij} H_{k-1}^j + \sum_{j=1}^n W_{ij} T_k^j + \alpha_k \mathcal{H}_k^i + T_{k+1}^i \\
&= \sum_{t=1}^n W_{it}^3 v_{k-2}^t + \alpha_{k-2} \sum_{l=1}^n W_{il}^2 H_{k-2}^l + \sum_{l=1}^n W_{il}^2 T_{k-1}^l + \alpha_{k-1} \sum_{j=1}^n W_{ij} H_{k-1}^j + \sum_{j=1}^n W_{ij} T_k^j + \alpha_k \mathcal{H}_k^i + T_{k+1}^i \\
&\overset{(j)}{=} \sum_{l=1}^n W_{il}^{k+1} v_0^l + \sum_{s=1}^k \alpha_{s-1} \sum_{j=1}^n W_{ij}^{k-s+1} \mathcal{H}_{s-1}^j + \sum_{s=1}^k \sum_{j=1}^n W_{ij}^{k-s+1} T_s^j + \alpha_k \mathcal{H}_k^i + T_{k+1}^i. \qquad (4.42)
\end{aligned}
$$

Here, (a) follows from (4.12), (b) follows from (4.38), (c) follows from (4.10), (d) follows from (4.9) and (e) follows from the definition (4.40). Moreover, (f) follows from recursively using an expression of v_k^j from (4.41). (g) follows from the matrix multiplication formula. If $C = AA$ where A is a $n \times n$ square matrix, then $C_{ij} = W_{ij}^2 = \sum_{l=1}^n A_{il} A_{lj}$. Note that W_{il}^2 represents the $(i,l)^{th}$ entry of matrix W^2. Equality (h) follows directly from (g). For clarity of exposition we unroll v_{k-1}^l further using (4.41) to show (i). We recursively keep doing this until we unroll all iterations until $k = 0$ to get the expression (j).

Recall from (4.40), the definition $\mathcal{H}_k^j = \sum_{l=1}^n W_{jl} r_k^{lj}$ and observe that $\sum_{j=1}^n \mathcal{H}_k^j = \sum_{j=1}^n \sum_{l=1}^n W_{jl} r_k^{lj}$. Since the non-zero weight (W_{ij}) corresponding to edge (i,j) is always δ, we have

$$\sum_{j=1}^n \sum_{l=1}^n W_{jl} r_k^{lj} = \delta \sum_{j=1}^n \sum_{l=1}^n r_k^{lj} = \delta \sum_{l=1}^n \left(\sum_{j \in \mathcal{N}_l} r_k^{lj} \right) = 0,$$

following the locally balanced property of perturbations (4.8). This implies, $(1/n)\sum_{j=1}^n \mathcal{H}_k^j = 0$ for any k. This implies that the sum of effective noise/perturbation added by players is zero, or the effective noise is correlated and adds to zero over the network. Using this, we subtract $(1/n)\sum_{j=1}^n \mathcal{H}_k^j = 0$ from (4.42), to get,

$$v_{k+1}^i = \sum_{l=1}^n W_{il}^{k+1} v_0^l + \sum_{s=1}^k \alpha_{s-1} \sum_{j=1}^n \left(W_{ij}^{k-s+1} - \frac{1}{n} \right) \mathcal{H}_{s-1}^j + \sum_{s=1}^k \sum_{j=1}^n W_{ij}^{k-s+1} T_s^j + T_{k+1}^i + \alpha_k \mathcal{H}_k^i.$$

Rearranging a few terms in this expression, using (4.12), gives us,

$$\begin{aligned} \widehat{v}_k^i &= v_{k+1}^i - T_{k+1}^i \\ &= \sum_{l=1}^n W_{il}^{k+1} v_0^l + \sum_{s=1}^k \sum_{j=1}^n W_{ij}^{k-s+1} T_s^j + \sum_{s=1}^k \alpha_{s-1} \sum_{j=1}^n \left(W_{ij}^{k-s+1} - \frac{1}{n} \right) \mathcal{H}_{s-1}^j + \alpha_k \mathcal{H}_k^i. \end{aligned} \tag{4.43}$$

Observe that, $\|T_s^i\| = \|x_s^i - x_{s-1}^i\|$ is bounded as follows, using the fact that gradients of cost function are bounded,

$$\begin{aligned} \|T_s^i\| &= \|\mathcal{P}_{\mathcal{X}^i}[x_{s-1}^i - \alpha_{s-1} \nabla_{x^i} f_i(x_{s-1}^i, n\widehat{v}_{s-1}^i)] - x_{s-1}^i\| \\ &\leq \|x_{s-1}^i - \alpha_{s-1} \nabla_{x^i} f_i(x_{s-1}^i, n\widehat{v}_{s-1}^i) - x_{s-1}^i\| \\ &\leq \alpha_{s-1} C. \end{aligned} \tag{4.44}$$

And, as W is row stochastic and $\|r_k^{ji}\| \leq \Delta$, we have,

$$\|H_k^i\| = \|\sum_{j=1}^n W_{ij} r_k^{ji}\| \leq \|\Delta\|. \tag{4.45}$$

We subtract (4.39) from (4.43) and take norm of both sides, to get

$$\begin{aligned} \|y_k - \widehat{v}_k^i\| \leq \sum_{l=1}^n \|W_{il}^{k+1} - \frac{1}{n}\| \|v_0^l\| + \sum_{s=1}^k \sum_{j=1}^n \|W_{ij}^{k-s+1} - \frac{1}{n}\| \|T_s^j\| \\ + \sum_{s=1}^k \alpha_{s-1} \sum_{j=1}^n \|W_{ij}^{k-s+1} - \frac{1}{n}\| \|\mathcal{H}_{s-1}^j\| + \alpha_k \|\mathcal{H}_k^i\|, \end{aligned}$$

and further use the geometric convergence of product of doubly stochastic transition matrices [82] and the bounds on $\|T_s^j\|$ in (4.44) and $\|H_s^j\|$ in (4.45), to get,

$$\|y_k - \widehat{v}_k^i\| \leq \theta\beta^k Mn + \theta nC \sum_{s=1}^{k} \beta^{k-s}\alpha_{s-1} + \theta n\Delta \sum_{s=1}^{k} \beta^{k-s}\alpha_{s-1} + \Delta\alpha_k,$$

where $\theta > 0$ and $0 < \beta < 1$ are dependent only on the weight matrix W, and $M = \max_{l\in\mathcal{N}} \|v_0^l\|$.
Finally, we show that $\sum_k \alpha_k \|y_k - \widehat{v}_k^i\| < \infty$ for all $i \in \mathcal{V}$. First observe,

$$\begin{aligned}
\alpha_k\|y_k - \widehat{v}_k^i\| &\leq \theta nM\alpha_k\beta^k + \theta n(C+\Delta)\sum_{s=1}^{k}\beta^{k-s}\alpha_k\alpha_{s-1} + \Delta\alpha_k^2 \\
&\leq \underbrace{\theta nM\alpha_k\beta^k}_{U_k^1} + \underbrace{\theta n(C+\Delta)\sum_{s=1}^{k}\beta^{k-s}\alpha_{s-1}^2}_{U_k^2} + \underbrace{\Delta\alpha_k^2}_{U_k^3}. \qquad (\alpha_{s-1} \geq \alpha_k, \text{ provided } s-1 \leq k)
\end{aligned}$$

Now observe that $\sum_k \alpha_k\|y_k - \widehat{v}_k^i\| \leq \sum_k (U_k^1 + U_k^2 + U_k^3)$ and we will individually show that $\sum_k U_k^1 < \infty$, $\sum_k U_k^2 < \infty$ and $\sum_k U_k^3 < \infty$ to prove our lemma.

We use the ratio test from [106] to show that $\sum_k U_k^1 < \infty$. We consider $\lim_{k\to\infty} \frac{U_{k+1}^1}{U_k^1} = \lim_{k\to\infty} \frac{\alpha_{k+1}\beta^{k+1}}{\alpha_k\beta^k} \leq \beta < 1$. This follows from the fact that $\alpha_k \geq \alpha_{k+1}$ for each k and $\beta < 1$. As $\lim_{k\to\infty} \frac{U_{k+1}^1}{U_k^1} < 1$ we know that the series is convergent and $\sum_k U_k^1 < \infty$.

We exploit Lemma 3.1 from [38]. Recall, $\{\alpha_k^2\}$ is a non-negative sequence with $\sum_k \alpha_k^2 < \infty$ and $0 < \beta < 1$. Using Lemma 3.1 from [38], we get, $\sum_k \left(\sum_{j=0}^{k} \beta^{k-j}(\alpha_j)^2\right) < \infty$. Consequently, $\sum_k U_k^2 = \theta n(C+\Delta)\sum_k \left(\sum_{s=1}^{k}\beta^{k-s}\alpha_{s-1}^2\right) = \theta n(C+\Delta)\frac{1}{\beta}\sum_k \left(\sum_{s=1}^{k}\beta^{k-(s-1)}\alpha_{s-1}^2\right) < \infty$. Here, the last equality follows from a multiplication and division by $\beta(\beta > 0)$.

We get $\sum_k U_k^3 < \infty$ directly following the square-summable nature of step-size following (4.7). We have hence showed that $\sum_k (U_k^1 + U_k^2 + U_k^3) < \infty$ implying, $\sum_k \alpha_k\|y_k - \widehat{v}_k^i\| < \infty$ for each $i \in \mathcal{V}$. This directly gives us $\sum_k \alpha_k \sum_{i=1}^{n} \|y_k - \widehat{v}_k^i\| < \infty$ as the number of players is finite. □

CHAPTER 5

PRIVATE AND FINITE-TIME ALGORITHM FOR SOLVING A DISTRIBUTED SYSTEM OF LINEAR EQUATIONS

5.1 Introduction

Consider a system of linear equations,

$$Az = b, \tag{5.1}$$

where $z \in \mathbb{R}^d$ is the d-dimensional solution to be learned, and $A \in \mathbb{R}^{p\times d}$, $b \in \mathbb{R}^{p\times 1}$ encode p linear equations in d-variables. The system of linear equations is horizontally partitioned and stored over a network of n devices. Each device $i \in \{1, \ldots, n\}$ has access to p_i linear equations denoted by

$$A_i z = b_i, \tag{5.2}$$

where $A_i \in \mathbb{R}^{p_i\times d}$ and $b_i \in \mathbb{R}^{p_i\times 1}$. For instance, in Figure 5.1 we show a network of $n = 5$ nodes and horizontal partitioning of $p = 15$ linear equations in $d = 5$ variables using colored blocks. In this chapter, we consider an honest-but-curious adversary that corrupts at most τ devices/nodes in the network and exploits observed information to infer private data. We are interested in designing a fast, distributed algorithm that solves problem (5.1), while, protecting privacy of local information (A_i, b_i) against such an honest-but-curious adversary.

Solving a system of linear algebraic equations is a fundamental problem that is central to analysis of electrical networks, sensor networking, supply chain management and filtering [107–109]. Several of these applications involve linear equations being stored at geographically separated devices/agents that are connected via a communication network. The geographic separation between agents along with communication constraints and unavailability of central servers necessitates design of distributed algorithms. Recently, several articles have proposed distributed algorithms for solving (5.1), [10, 11, 110–114] to name a few. In this work, we specifically focus on designing private methods that protect sensitive and private linear equations at each device/agent.

As an example, consider a network of n imperfect and imprecise sensors. Each sensor takes measurements to estimate the location of an object, parameterized by $z = [a, b, c] \in \mathbb{R}^3$. Each sensors records measurements

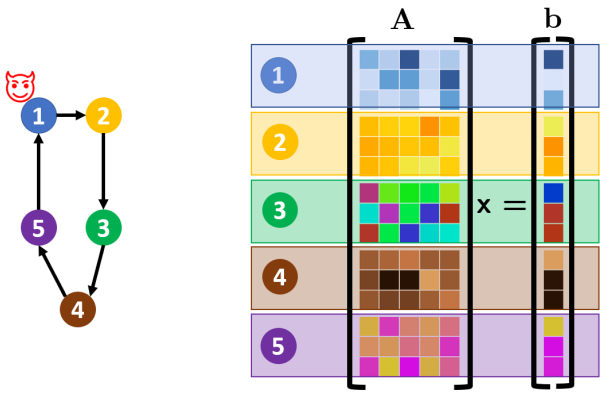


Figure 5.1: Directed network with $n = 5$ nodes and 1 adversary. A and b are horizontally partitioned and stored at n nodes. Local information is private to the nodes.

that expressed as a linear equation in a, b, c. Solving the system of linear equation formed by aggregating observations from all sensors leads us to an estimate of location. The coefficients of the linear equations store private information such as the location, model type etc. of the sensors. We need a distributed algorithm to solve the system of linear equations while protecting the privacy of coefficients of linear equations.

Literature has explored several approaches to solving a distributed system of linear equations. Authors in [108] formulated the problem as a parameter estimation task. Consensus or gossip based distributed estimation algorithms are then used to solve (5.1). Interleaved consensus and local projection based methods are explored in [10, 11, 110]. These direct methods, involving locally feasible iterates that move only along the null space of local coefficient matrix A_i, converge exponentially fast. One can also view solving (5.1) as a constrained consensus [115] problem, where agents attempt to agree to a variable z such that local equations at each agent are satisfied. Problem (5.1) can also be formulated as a convex optimization problem, specifically linear regression, and solved using plethora of distributed optimization methods as explored in [111]. Authors augment their optimization based algorithms with a finite-time decentralized consensus scheme to achieve finite-time convergence of iterates to the solution. In comparison, our approach is not incremental and only needs two steps – (a) computing local updates, followed by, (b) fast aggregation and exact solution computation. Our algorithm converges to the unique least squares solution in finite-time and additionally guarantees information-theoretic privacy of local data/equations (A_i, b_i).

Few algorithms focus on privacy of local equations (A_i, b_i). In this chapter, we design algorithms with provable privacy properties. One can leverage vast private optimization literature by reformulating problem (5.1) as a least-squares regression problem and use privacy preserving optimization algorithms [20, 27, 55, 58, 59, 61] on the resulting strongly convex cost function. Differential privacy is employed in [58, 59] for distributed convex optimization, however, it suffers from a fundamental privacy - accuracy trade-off [61]. Authors in [53, 55] use partially homomorphic encryption for privacy. However, these methods incur high computational costs and unsuitable for high-dimensional problems. A Secure Multi-party Computation (SMC) based method for privately solving system of linear equations is proposed in [116], however, this solution requires a central

Crypto System Provider for generating garbled circuits. In this work, we design a purely distributed solution. Liu *et al.* propose a private linear system solver in [71, 72], however, as we discuss in Section 5.4.1, our algorithm is faster, requiring fewer iterations. Our prior work [20, 27] proposes non-identifiability over equivalent problems as a privacy definition and algorithms to achieve it. It admits privacy and accuracy guarantees simultaneously, however, it uses a weaker adversary model that does not know distribution of noise/perturbations used by agents.

In this chapter, we consider statistical privacy from [117, 118]. This definition of privacy allows for a stronger adversary that knows distribution of random numbers used by agents and has unbounded computational capabilities. We call this definition information-theoretic because additional observations do not lead to incremental improvement of adversarial knowledge about private data. Algorithms in [117, 118] provide algorithms for private average consensus over undirected graphs. We generalize their work to solve the problem over directed graphs and show finite-time convergence guarantee.

Our Contributions:

Algorithm: We present an algorithm, TITAN (priva**T**e f**I**nite **T**ime **A**verage co**N**sensus), that solves an average consensus problem over directed graphs in finite-time, while protecting statistical privacy of local inputs. It involves a distributed *Obfuscation Step* to hide private inputs, followed by a distributed recovery algorithm to collect all perturbed inputs at each node. Agents then compute exact average using the recovered perturbed inputs. We further leverage TITAN to solve Problem (5.1) in finite time with strong statistical privacy guarantees.

Convergence Results: We show that TITAN converges to the exact average in finite time that depends only on the number of nodes and graph diameter. We show that graph being strongly connected is sufficient for convergence. Moreover, algorithm needs to know only an *an upper bound* on the number of nodes and graph diameter. We do not require out-degree of nodes to be known, a limitation commonly observed in push-sum type methods [119], for solving average consensus over directed graphs. TITAN based solver converges to the unique least squares solution of (5.1) in finite time.

Privacy Results: We show that TITAN provides statistical privacy of local inputs as long as weak vertex connectivity of communication graph is at least $\tau + 1$. This condition is also necessary and hence tight. Consider two problems (5.1) characterized by (A, b) and (A', b'), such that rows of (A, b) and (A', b') stored at corrupted nodes are identical and the least squares solution for both systems is the same. For such a pair of problems, our privacy guarantee implies that the distribution of observations made by the adversary is identical.

5.2 Problem Formulation

Consider a group of n agents/nodes connected in a directed network. We model the directed communication network as a directed graph $\mathfrak{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, \ldots, n\}$ denotes the set of nodes and $\mathcal{E}$ denote reliable, loss-less, synchronous and directed communication links.

Recall, we are interested in solving a system of linear equations, problem (5.1), that is horizontally partitioned and stored at n agents. Each agent i has access to private p_i linear equations in d variables denoted by (5.2). Equivalently our problem formulation states that each agent i has access to private data matrices (A_i, b_i) that characterize (5.2). In this work, we assume that our system of linear equations in (5.1) admits a unique least squares solution, i.e. $A^T A$ matrix is full rank. If an exact solution for (5.1) exists, then it matches the least squares solution. Let $\mathsf{ls\text{-}sol}(A, b)$ denote the unique least squares solution of $Az = b$. We wish to compute $z^* \triangleq \mathsf{ls\text{-}sol}(A, b)$ that solves the collective system $Az = b$, while, protecting privacy of local data (A_i, b_i). Next, we discuss the adversary model, privacy definition and few preliminaries.

5.2.1 Adversary Model and Privacy Definition

We consider an honest-but-curious adversary, that follows the prescribed protocol, however, is interested in learning private information from other agents. The adversary can corrupt at most τ nodes and has access to all the information stored, processed and received by the corrupted nodes. Let us denote the corrupted nodes as $\mathcal{A}$. We assume the adversary and corrupted nodes have unbounded storage and computational capabilities.

The coefficients of linear equations encode private and sensitive information. In the context of robotic or sensor networks, the coefficients of linear equations conceal sensor observations and measurements – information private to agents. In the context of supply chain management and logistics, linear equations are used to optimize transport of raw-materials, and the coefficients of linear equations often leak business sensitive information about quantity and type of raw-materials/products being transported by a company. Mathematically, adversary seeks to learn local coefficient matrices (A_i, b_i) corresponding to any non-corrupt agent i.

Privacy requires that the observations made by the adversary do not leak significant information about the private inputs. We use the definition of information-theoretic or statistical privacy from [117,118] for average consensus over private inputs $\{x^i\}_{i=1}^n$. Let $\mathsf{View}_{\mathcal{A}}(\{x^i\}_{i=1}^n)$ denote the set of random variables denoting the observations made by a set of adversarial nodes $\mathcal{A}$ given private inputs $\{x^i\}_{i=1}^n$.

Definition 3. *An algorithm is $\mathcal{A}-$private, if for all inputs $\{x^i\}_{i=1}^n$ and $\{y^i\}_{i=1}^n$, such that $x^i = y^i$ for all $i \in \mathcal{A}$ and $\sum_{i=1}^n x^i = \sum_{i=1}^n y^i$, the distributions $\mathsf{View}_{\mathcal{A}}(\{x^i\}_{i=1}^n)$ and $\mathsf{View}_{\mathcal{A}}(\{y^i\}_{i=1}^n)$ are the same.*

The definition implies that the observations made by the adversary have the same probability distributions for both private inputs x^i and y^i, making them statistically indistinguishable for adversary $\mathcal{A}$.

We borrow the privacy definition for average consensus as elaborated above to formally define statistical privacy for our problem as follows. Let $A[i,:]$ denote the i^{th} row of matrix A and $b[i]$ denote the i^{th} element of column vector b, where A and b characterize the set of linear equations from (5.1). Let $\mathsf{View}_{\mathcal{A}}((A,b))$ be the random variable denoting the observations made by set of adversarial nodes $\mathcal{A}$ given private inputs (A,b).

Definition 4. *A distributed protocol is $\mathcal{A}$-private if for all (A,b) and (A',b'), such that $A[i,:] = A'[i,:]$, $b[i] = b'[i]$ for all equations i stored at agents in $\mathcal{A}$, and $\mathsf{ls\text{-}sol}(A,b) = \mathsf{ls\text{-}sol}(A',b')$, the distributions of $\mathsf{View}_{\mathcal{A}}((A,b))$ and $\mathsf{View}_{\mathcal{A}}((A',b'))$ are identical.*

Intuitively, for all systems of equations (A',b'), such that linear equations stored at $\mathcal{A}$ are the same and $\mathsf{ls\text{-}sol}(A,b) = \mathsf{ls\text{-}sol}(A',b')$, observations made by the adversary will have the same distribution, making the system of linear equations statistically indistinguishable from adversary $\mathcal{A}$'s perspective.

5.2.2 Notation and Preliminaries

For each node $i \in \mathcal{V}$, we define in-neighbor set, $\mathcal{N}_i^{in}$, as the set of all nodes that send information to node i; and out-neighbor set, $\mathcal{N}_i^{out}$, as the set of all nodes that receive information from node i. Let $\delta(\mathfrak{G})$ denote the diameter of graph $\mathfrak{G}$, and B denote the incidence matrix of graph $\mathfrak{G}$. We define the incidence matrix as,

$$B_{ij} = \begin{cases} -1 & \text{if node } i \text{ is the tail of edge } j \in \mathcal{E}, \\ 1 & \text{if node } i \text{ is the head of edge } j \in \mathcal{E}, \\ 0 & \text{otherwise,} \end{cases} \tag{5.3}$$

for $i \in \mathcal{V}$ and $j \in \mathcal{E}$. Let $\{\perp\}^m$ denote an m-dimensional empty vector. Let $\mathcal{U}[a,b)$ be uniform distribution over $[a,b)$.

Modular arithmetic, typically defined over a finite field of integers, involves numbers wrapping around when reaching a certain value. In this work, we use real numbers and define an extension of modular arithmetic over reals:

Definition 5. *Consider a real interval $[0,a) \in \mathbb{R}$. We define $\mathrm{mod}(x,a) \in [0,a)$ as the remainder obtained when x is divided by a and the quotient is an integer. That is, $\mathrm{mod}(x,a) = x - \lfloor x/a \rfloor a$, where $\lfloor \cdot \rfloor$ is the floor function and $\lfloor x/a \rfloor$ is the unique integer such that $\mathrm{mod}(x,a) \in [0,a)$.*

Modulo operator satisfies useful properties as detailed below. The proofs are included in the appendix.

Remark 3. *Modular arithmetic over reals satisfies following properties for all real numbers $\{y^i\}_i$ and integers q and a.*

1. $\mathrm{mod}(\sum_{i=1}^{q} y^i, a) = \mathrm{mod}(\sum_{i=1}^{q} \mathrm{mod}(y^i,a), a)$.
2. *If $y \in (0,a)$, then $\mathrm{mod}(-y,a) = a - \mathrm{mod}(y,a)$. If $y = 0$, then $\mathrm{mod}(-y,a) = -\mathrm{mod}(y,a) = 0$.*

5.3 TITAN - Private Average Consensus

In this section, we develop TITAN, an algorithm for solving distributed average consensus with provable statistical privacy and finite-time convergence. In Section 5.4, we will use TITAN to solve Problem (5.1) with statistical privacy of local data (A_i, b_i) and finite-time convergence to z^*.

Consider a simple average consensus problem over n agents connected using directed graph $\mathfrak{G}$. Each node $i \in \mathcal{V}$ has access to private input $x^i \in [0, a)$. The objective is to compute average $(1/n)\sum_{i=1}^n x^i$, while, protecting statistical privacy of inputs x^i (see Section 5.2.1).

TITAN involves an obfuscation step to hide private information and generate obfuscated inputs. This is followed by several rounds of Top-k consensus primitive for distributed recovery of perturbed inputs. Consequently, we exploit *modulo aggregate invariance* property of the obfuscation step and locally process perturbed inputs to arrive at desired average. The obfuscation step guarantees statistical privacy of inputs, while, the distributed recovery and local computation of average are key to finite-time convergence of the algorithm. We detail each of the steps below.

Algorithm 6 TITAN$\big(\{x^i\}_{i=1}^n, T, k, n\big)$

Input: $\{x^i\}_{i=1}^n$, where $x^i \in [0, a)\ \forall i \in \mathcal{V}$, T, k
Output: $\overline{x} = (1/n)\sum_{i=1}^n x^i$

Initialize: Node i initializes $y^i = \{\perp\}^n$, $I^i = \{\perp\}^n$

Obfuscation Step

1: Node i sends random numbers $r^{ij} \sim \mathcal{U}[0, na)$ to each out-neighbor $j \in \mathcal{N}_i^{out}$
2: Node i constructs perturbation t^i,

$$t^i = \text{mod}\left(\sum_{j\in\mathcal{N}_i^{in}} r^{ji} - \sum_{j\in\mathcal{N}_i^{out}} r^{ij}, na\right). \tag{5.4}$$

3: Each node i perturbs private input,

$$\tilde{x}^i = \text{mod}(x^i + t^i, na). \tag{5.5}$$

Distributed Recovery using Top-k Primitive

4: Each node $i \in \mathcal{V}$ runs Top-k primitive $\lceil n/k \rceil$ times
5: for $\ell = 0, 1, \ldots, \lceil n/k \rceil - 1$
6: $\quad \big(y^i[\ell k + 1 : (\ell+1)k], I^i[\ell k + 1 : (\ell+1)k]\big) = \text{Top-k}\left(\left(\{\tilde{x}^i\}_{i=1}^n \setminus y^i[1:\ell k], \{i\}_{i=1}^n \setminus I^i[1:\ell k]\right), T\right)$

7: **Return:** $\overline{x} = (1/n)\text{mod}(\sum_{l=1}^n y^i[l], na)$

Obfuscation Step

The *obfuscation step* is a distributed method to generate network correlated noise that vanishes under modulo operation over the aggregate. First, each node i sends uniform random noise $r^{ij} \sim \mathcal{U}[0, na)$ to out-neighbors $\mathcal{N}_i^{out}$ and receives r^{li} from in-neighbors $l \in \mathcal{N}_i^{in}$ (Line 1, Algorithm 6).

Next, each agent i computes perturbation t^i using (5.4) (Line 2, Algorithm 6). Observe that due to the modulo operation, each perturbation t^i satisfies $t^i \in [0, na)$.

Finally, agent i adds perturbation t^i to its private value x^i and performs a modulo operation about na to get the perturbed input, $\tilde{x}^i$, as seen in (5.5) (Line 3, Algorithm 6). Observe that $\tilde{x}^i \in [0, na)$.

We now show the *modulo aggregate invariance* property of the obfuscation mechanism described above. Notice that each noise r^{ij} is added by node j to get t^j (before modulo operation) and subtracted by node i to get t^i (before modulo operation). This gives us,

$$\begin{aligned}\operatorname{mod}\left(\sum_{i=1}^{n} t^i, na\right) &\overset{(a)}{=} \operatorname{mod}\left(\sum_{i=1}^{n} \operatorname{mod}\left(\sum_{j\in\mathcal{N}_i^{in}} r^{ji} - \sum_{j\in\mathcal{N}_i^{out}} r^{ij}, na\right), na\right) \\ &\overset{(b)}{=} \operatorname{mod}\left(\sum_{i=1}^{n}\left[\sum_{j\in\mathcal{N}_i^{in}} r^{ji} - \sum_{j\in\mathcal{N}_i^{out}} r^{ij}\right], na\right) \overset{(c)}{=} \operatorname{mod}(0, na) = 0. \end{aligned} \tag{5.6}$$

In the above expression, (a) follows from definition of t^i in (5.4), (b) follows from property 1 in Remark 3, and (c) is a consequence of perturbation design in (5.4). We call this as *modulo aggregate invariance* of the obfuscation step.

Distributed Recovery via Top-k Consensus Primitive

We perform distributed recovery of perturbed inputs, that is, we run a distributed algorithm to "gather" all perturbed inputs ($\{\tilde{x}^i\}_{i=1}^n$) at each node. After the completion of this step, each node i will have access to the entire set of perturbed inputs $\{\tilde{x}^i\}_{i=1}^n$. Top-k consensus primitive is a method to perform distributed recovery.

The Top-k consensus primitive is a distributed protocol for all nodes to agree on the largest k inputs in the network. In TITAN, we run the Top-k consensus primitive and store the resulting list of top-k perturbed inputs. We then run Top-k primitive again while excluding the perturbed inputs recovered from prior iterations. Executing Top-k consensus primitive successively $\lceil n/k \rceil$ times leads us to the list of all perturbed inputs in the network (Lines 4–6, Algorithm 6).

Top-k Consensus Primitive: Recall, each node i has access to a perturbed input $\tilde{x}^i$ and an unique identifier i. The Top-k consensus is a consensus protocol for nodes to agree over k largest input values and associated node id's with ties going to nodes with larger id. Formally, the algorithm results in each node agreeing on $\{(x^{(1)}, \ldots, x^{(k)}), (i^{(1)}, \ldots, i^{(k)})\}$, where $x^{(1)} \geq \ldots \geq x^{(k)} \geq x^{(k+1)} \geq \ldots \geq x^{(n)}$ is the ordering of private

Algorithm 7 Top-k consensus: Top-k($\{(x^i, i)\}_{i=1}^n$,T)

Input: $\{(x^i, i)\}_{i=1}^n$, T, k
Output: $\{(x^{(1)}, \ldots, x^{(k)}), (i^{(1)}, \ldots, i^{(k)})\}$
Initialization: Node i initializes two k dimensional vectors $L^i = \{x^i, \perp, \ldots, \perp\}$ and $\ell^i = \{i, \perp, \ldots, \perp\}$

1: **for** $t = 1, \ldots, T$:
2: Node i sends L^i and ℓ^i to out-neighbors $\mathcal{N}_i^{out}$
3: Each node $i \in \mathcal{V}$ performs:
4: **for** $z = 1, \ldots, k$:
5: $L_+^i[z] = \max(L^i \cup_{j \in \mathcal{N}_i^{in}} L^j \setminus (\cup_{s=1}^{z-1} L_+^i[s]))$
6: $\ell_+^i[z] =$ id corresponding to $L^i[z]$
... Ties go to node with larger id
7: $L^i = L_+^i$ and $\ell^i = \ell_+^i$

8: **Return:** $\{L^i, \ell^i\} \rightarrow \{(x^{(1)}, \ldots, x^{(k)}), (i^{(1)}, \ldots, i^{(k)})\}$

inputs $\{x^j\}_{j=1}^n$ and $i^{(k)}$ denote the ids corresponding to $x^{(k)}$ for each k. Ties go to nodes with larger id, implying, if $x^{(j)} = x^{(j+1)}$ then $i^{(j)} > i^{(j+1)}$.

Each node i stores an estimate of k largest inputs and their id's denoted by L^i and ℓ^i respectively. These vectors, L^i and ℓ^i, are initialized with local private input and own agent id respectively (Line 1, Algorithm 7).

At each iteration $t = 1, \ldots, T$, agent i share their estimates L^i and ℓ^i to out-neighbors and receives L^j and ℓ^j from in-neighbors $j \in \mathcal{N}_i^{in}$ (Line 3, Algorithm 7). Agent i sets L_+^i equal to the k largest values of available inputs, that is L^i and $\cup_{j \in \mathcal{N}_i^{in}} L^j$; and sets ℓ_+^i as id's corresponding to the entries in L_+^i (Lines 5–6, Algorithm 7). In case of ties, that is several agents having the same input, the tie goes to agent with larger id. Next, agents update the local estimate of top k entries as $L^i = L_+^i$ and $\ell^i = \ell_+^i$ (Line 8, Algorithm 7).

This process of selection of largest perturbed input and it's id is repeated T times. As stated in Theorem 8, each L^i and ℓ^i converge to the *Top-k* values and associated id's respectively provided that $T \geq \delta(\mathfrak{G})$, the graph diameter.

Recall, that running Top-k consensus primitive $\lceil n/k \rceil$ times, successively, allows each node to recover perturbed inputs $\{\tilde{x}^i\}_{i=1}^n$. Agents then add the perturbed inputs recovered by Top-k consensus algorithm and exploit modulo aggregate invariance property of obfuscation step to exactly compute the average (Line 7, Algorithm 6), $\overline{x} = (1/n)\text{mod}(\sum_{l=1}^n y^i[l], na)$.

5.3.1 Results and Discussion

Correctness Guarantee: We first begin by a correctness result for the Top-k consensus primitive. It is a consequence of convergence of max consensus over directed graphs.

Theorem 8 (Correctness of Top-k)**.** *If $\mathfrak{G}$ is a strongly connected graph with diameter $\delta(\mathfrak{G})$ and $T \geq \delta(\mathfrak{G})$, then $L^i = L^j = \{x^{(1)}, \ldots, x^{(k)}\}$, $\ell^i = \ell^j = \{I^{(1)}, \ldots, I^{(k)}\}$, for each $i, j \in \mathcal{V}$, for the Top-k consensus*

algorithm.

The result establishes a lower bound on parameter T for correctness of Top-k consensus primitive. If $\delta(\mathfrak{G})$ is not exactly known, we can set T to be any upper bound on $\delta(\mathfrak{G})$, without worrying about correctness.

The ability of Top-k protocol to recover k largest perturbed inputs leads us to the correctness guarantee for TITAN. As a consequence of Theorem 8, we can conclude that $\lceil n/k \rceil$ successive execution of Top-k primitive over perturbed inputs leads to recovery of all perturbed inputs at each node. Moreover, we use the modulo aggregate invariant property of the obfuscation step, implying,

$$\mathrm{mod}\left(\sum_{i=1}^{n} \tilde{x}^i, na\right) = \sum_{i=1}^{n} x^i. \tag{5.7}$$

We prove this in Section 5.6. Consequently, perturbed inputs allows us to compute the correct aggregate and average. The following result formally states this result:

Theorem 9 (Correctness of TITAN)**.** *If $\mathfrak{G}$ is strongly connected and $T \geq \delta(\mathfrak{G})$, then TITAN (Algorithm 6) converges to the exact average of inputs $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x^i$ in finite time given by $T\lceil n/k \rceil$.*

Note, for finite-time convergence, we only need $\mathfrak{G}$ to be strongly connected. The time required for convergence is dependent only on number of agents n,[1] parameter T (an upper bound on graph diameter $\delta(\mathfrak{G})$) and parameter k. The proofs are presented in detail in Section 5.6.1.

Privacy Guarantee: The privacy guarantee is a consequence of the obfuscation step used in TITAN. Let $\overline{\mathfrak{G}} = (\overline{\mathcal{V}}, \overline{\mathcal{E}})$ be the undirected version of $\mathfrak{G}$. More specifically, $\overline{\mathfrak{G}}$ has the vertex set $\overline{\mathcal{V}} = \mathcal{V}$ and the edge set $\overline{\mathcal{E}}$ contains an undirected edge between any two nodes u and v if a directed edge $(u, v) \in \mathcal{E}$ or $(v, u) \in \mathcal{E}$. Consequently, $\overline{\mathfrak{G}}$ is undirected. We define *weak vertex-connectivity* of a directed graph $\mathfrak{G}$ as the vertex-connectivity of its undirected variant $\overline{\mathfrak{G}}$. Weak vertex-connectivity of $\mathfrak{G}$ is $\kappa(\overline{\mathfrak{G}})$, where κ denotes the vertex-connectivity. We show that provided the weak vertex-connectivity of $\mathfrak{G}$ is at least $\tau + 1$, TITAN preserves statistical privacy of input.

Theorem 10 (Privacy of TITAN)**.** *If weak vertex-connectivity $\kappa(\overline{\mathfrak{G}}) \geq \tau + 1$ and $\mathfrak{G}$ is strongly connected, then TITAN is $\mathcal{A}$-private against any set of adversaries $\mathcal{A}$ such that $|\mathcal{A}| \leq \tau$.*

Note, we require $\mathfrak{G}$ to be both strongly connected and possess weak vertex-connectivity of at least $\tau + 1$ for achieving both finite-time correctness and statistical privacy guarantees. Strong connectivity of $\mathfrak{G}$ is required for achieving average consensus (correctness). Moreover, the weak vertex-connectivity condition is also necessary and can be showed similar to the Proof of Theorem 2 in [69]. Consequently, the weak vertex-connectivity together with strong connectivity condition is *tight*. The proofs are included later in the chapter in Section 5.6.2.

[1] If agents know only an upper bound $\tilde{n} \geq n$, then we can run TITAN on inputs $x^i = 1$ for each $i \in \mathcal{V}$ while using $\tilde{n}$ instead of n and $a = 2$. Finally, by computing $\mathrm{mod}(\sum_i \tilde{x}^i, \tilde{n}a)$ we recover n exactly.

Memory Costs: Top-k primitive requires each node to maintain vectors L^i and ℓ^i in addition to recovered perturbed inputs. Overall the memory required per node is $(2k + n)d$ units, where d is the dimension of input x^i. This is larger as compared to standard average consensus methods [120] and ratio consensus methods [121] that require d and $2d$ units respectively. Observe the trade-off between Memory Overhead and Convergence Time. As k increases from 1 to n, the convergence time decreases from Tn to T, while the memory overhead (per node) increases from $(2 + n)d$ to $3nd$.

Communication Costs: The obfuscation step requires node i to send $|\mathcal{N}_i^{out}|d$ messages. Moreover, Top-k algorithm involves exchange of L^i and ℓ^i by each node. This additionally requires $2k|\mathcal{N}_i^{out}|T\lceil n/k\rceil d$ messages in total. Together, the communication overhead for node i is $|\mathcal{N}_i^{out}|\left(2kT\lceil n/k\rceil + 1\right)d$. Total communication cost (per node) is largely independent of k, as total information exchanged over entire execution does not change with k.

Comparison with FAIM: Oliva *et al.* propose FAIM, finite-time average-consensus by iterated max-consensus, in [122]. TITAN is a generalized form of FAIM and we can recover a statistically private version of FAIM by setting $k = 1$. Our work, in addition to finite-time average consensus, is directed toward protecting statistical privacy of local information.

5.4 Private Solver for System of Linear Equations

In this section, we develop a solver (Algorithm 8) employing TITAN to privately solve problem (5.1).

The least squares solution to system of linear equations (5.1) can be expressed in closed form as, $(A^TA)^{-1}A^Tb$. If an exact solution to (5.1) exists then the least squares solution matches it. Moreover, as the equations are horizontally partitioned, we can rewrite,

$$A^TA = \sum_{i=1}^{n} A_i^TA_i \quad \text{and} \quad A^Tb = \sum_{i=1}^{n} A_i^Tb_i. \tag{5.8}$$

Consequently, solution to system of linear equations can be computed by privately aggregating $A_i^TA_i$ and $A_i^Tb_i$ separately over the network and computing,

$$z^* = \left(\sum_{i=1}^{n} A_i^TA_i\right)^{-1}\left(\sum_{i=1}^{n} A_i^Tb_i\right). \tag{5.9}$$

Privately computing z^* is equivalent to agents privately aggregating $A_i^TA_i$ and $A_i^Tb_i$ over the directed network followed by locally computing x^* using (5.9).

We assume, w.l.o.g, that each entry in matrix local updates $A_i^TA_i$ and $A_i^Tb_i$ lies in $[0, a)$, where $a >$ largest entry in matrices $A_i^TA_i$ and $A_i^Tb_i$. Next, in Algorithm 8, we run TITAN on each entry of matrices

$\{A_i^T A_i\}_{i=1}^n$ to get update matrix X, run TITAN on each entry of vector $\{A_i^T b_i\}_{i=1}^n$ to get update vector Y. We select $T \geq \delta(\mathfrak{G})$ and a parameter $k \leq n$ following the discussion in Section 5.3.1. As a consequence of Theorem 9, the algorithms terminate in finite time and $X = \frac{1}{n}\sum_{i=1}^n A_i^T A_i$ and $Y = \frac{1}{n}\sum_{i=1}^n A_i^T b_i$. Finally, we use (5.9) to compute the least squares solution.

5.4.1 Results and Discussion

Correctness and Privacy: Our solution (Algorithm 8) involves using TITAN on each entry of matrices, $A_i^T A_i$ and $A_i^T b_i$. As a consequence of Theorem 9, we know that provided $\mathfrak{G}$ is strongly connected and parameter $T \geq \delta(\mathfrak{G})$, we get accurate estimate of $\sum_{i=1}^n A_i^T A_i$ and $\sum_{i=1}^n A_i^T b_i$ in finite time. Using (5.9), we compute solution z^*, and as a result we have solved (5.1) accurately in finite time. From Theorem 10, provided $\kappa(\overline{\mathfrak{G}}) \geq \tau + 1$, TITAN preserves the statistical privacy of local inputs $(A_i^T A_i, A_i^T b_i)$, equivalently preserves the statistical privacy of (A_i, b_i), against any adversary that corrupts $\mathcal{A}$ subject to $|\mathcal{A}| \leq \tau$.

Comparison with Relevant Literature: Liu *et al.* propose a privacy mechanism, an alternative to the obfuscation mechanism in TITAN, and augment it to gossip algorithms for privately solving average consensus. This private average consensus is used along with direct method [10] to arrive at a private linear system solver. However, it requires agents to reach complete consensus between successive direct projection based steps. This increases the number of iterations needed to solve the problem and significantly increases communication costs as the underlying method [10] is only linearly convergent. Distributed Recovery phase in TITAN also requires complete consensus, but we do it in finite time using Top-k primitive, and it is only performed once.

Authors in [111] propose a finite-time solver for solving (5.1). However, under this protocol an arbitrarily chosen node/agent observes states for all nodes, and the observations are used compute the exact solution. This algorithm was not designed to protect privacy of local information and consequently leads to large privacy violations by the arbitrarily chosen node. Algorithm 8 solves (5.1) in finite time, while additionally protecting statistical privacy of local equations. Moreover, the algorithm in [111] is computationally expensive – requiring matrix singularity checks and kernel space computation. In comparison, our algorithm is inexpensive and the most expensive step is matrix inversion in (5.9), that needs to be performed only once.

Algorithm 8 Private Solver for Problem (5.1)

Input: $\{A_i^T A_i, A_i^T b_i\}_{i=1}^n$ and $a >$ the largest entry of $[A_i^T A_i \; A_i^T b_i]$ over all $i \in \{1, 2, \ldots, n\}$
Output: z^*

1: Each node $i \in \mathcal{V}$ runs **TITAN** over each entry of $A_i^T A_i$ and $A_i^T b_i$
2: Compute: $X = \mathsf{TITAN}(\{A_i^T A_i\}_{i=1}^n, T, k, n)$
3: Compute: $Y = \mathsf{TITAN}(\{A_i^T b_i\}_{i=1}^n, T, k, n)$

4: **Return:** $z^* = (nX)^{-1}(nY)$

Direct methods [10,110], constrained consensus applied to linear systems [115] and distributed optimization methods applied to linear regression [43, 123, 124] are only linearly convergent, while, we provide superior finite-time convergence guarantee. Finite-time convergence of underlying consensus is critical for statistically private mechanisms such as the one in TITAN and the algorithms from [117,118]. These mechanisms rely on modulo arithmetic, if we add them to a linearly convergent solver which outputs in-exact average/aggregate, then performing modulo operation over the in-exact output, in final step, may arbitrarily amplify errors.

Improving Computational Efficiency: In our approach, each node computes an inverse, $\left(\sum_{i=1}^{n} A_i^T A_i\right)^{-1}$, which takes $\mathcal{O}(d^3)$ computations. Note, this inverse is not performed on private data. We can reduce computational cost by allowing one node to perform the inversion followed by transmitting the solution to all nodes either by flooding protocol or sending it over a spanning-tree of $\mathfrak{G}$.

5.5 Numerical Experiments

In this section, we perform two numerical experiments to validate Algorithm 8. First, we conduct a simple simulation over the problem defined in Figure 5.1 and show that update matrices $A_i^T A_i$ observed by the adversary appear to be random (Lemma 17). Second, we run a large scale experiment with synthetic data, with $n = 100$, $p = 10000$ and $d = 100$.

We have $n = 5$ nodes, with $p = 15$ linear equations in $d = 5$ variables being stored over five nodes (three equations each) as shown in Figure 5.1. We generated A and b matrices by drawing their entries from a Gaussian distribution (mean = 0 and variance = 2) and verified $A^T A$ is full rank. We executed Algorithm 8 at each node. We select parameter $T = n \geq \delta(\mathfrak{G}) = 4$, $k = n = 5$, and $a = 20$. The algorithm solves the problem exactly in $T = 6$ iterations. Note $\mathfrak{G}$ is strongly connected and has a weak vertex-connectivity of 2. Consequently, both accuracy and statistical privacy are guaranteed by Theorems 9 and 10. The perturbed update matrices generated after obfuscation step in TITAN are received by adversary node 1 and shown in Figure 5.2. We map the numerical values of each entry in the matrix to a color using standard python colormaps. Figure 5.2 shows that the perturbed updates are starkly different from private updates and appear random.

Consider a large scale system with $n = 100$ agents and $p = 10000$ equations in $d = 100$ variables that are horizontally partitioned for each agent to have $p_i = 100$ equations ($\forall i \in \mathcal{V}$). The coefficients of the linear equations are synthetically generated via a Gaussian process and admit a unique least squares solution. The graph $\mathfrak{G}$ is a directed ring with graph diameter $\delta(\mathfrak{G}) = n - 1 = 99$ and $\kappa(\overline{\mathfrak{G}}) = 2$. We run Algorithm 8 with parameters $T = n \geq \delta(\mathfrak{G})$ and $k = 10$. We consider an honest-but-curious adversary that corrupts at most one agent and from $\kappa(\overline{\mathfrak{G}}) = 2$ we guarantee statistical privacy of local data (A_i, b_i). The algorithm converges to the solution in 1000 iterations.

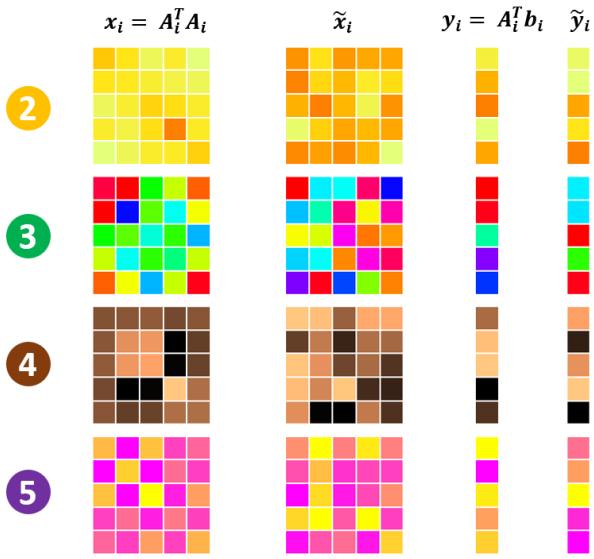


Figure 5.2: Inputs $x^i = A_i^T A_i$ and $y^i = A_i^T b_i$ for $i \in \mathcal{V}$ and perturbed inputs $\tilde{x}^i$ and $\tilde{y}^i$. The perturbed inputs appear to be random and very different from the private inputs.

5.6 Analysis and Proofs

5.6.1 Convergence Analysis

We first prove correctness of Top-k consensus primitive.

Proof of Theorem 8. In the Top-k consensus algorithm, each agent/node tries to keep track of the largest k inputs observed until then. As $T \geq \delta(\mathfrak{G})$, each one of the k largest inputs, i.e., $x^{(1)}, \ldots, x^{(k)}$, reaches each node in the network. Consequently, each nodes' local states L^i and ℓ^i converge to the k largest inputs in the network and associated id's. □

Next, we prove correctness of TITAN in finite-time.

Proof of Theorem 9. The correctness result in Theorem 9 follows from two key statements: (1) TITAN output is exactly equal to $\overline{x} = (1/n)\sum_{i=1}^n x^i$, and (2) TITAN converges in finite time. We prove both the statements next.

(1) Recall from Line 9 in Algorithm 6, that TITAN outputs $(1/n)\text{mod}(\sum_{i=1}^n y^i, na)$, where $\{y^i\}_{i=1}^n$ are obtained by running Top-k algorithm $\lceil n/k \rceil$ times on perturbed inputs $\{\tilde{x}^i\}_{i=1}^n$. The Top-k algorithm successively picks the k largest entries and ensures that $y^i = \tilde{x}^{(i)}$ for $i = 1, \ldots, n$, following Theorem 8. Consequently, $\sum_{i=1}^n y^i = \sum_{i=1}^n \tilde{x}^{(i)} = \sum_{i=1}^n \tilde{x}^i$ and the output reported by TITAN is $(1/n)\text{mod}\left(\sum_{i=1}^n \tilde{x}^i, na\right)$. We

use properties of modulo function to get,

$$\begin{aligned}\mathrm{mod}\left(\sum_{i=1}^{n}\tilde{x}_i, na\right) &= \mathrm{mod}\left(\sum_{i=1}^{n}\mathrm{mod}(x^i+t^i, na), na\right)\\ &\overset{(a)}{=} \mathrm{mod}\left(\sum_{i=1}^{n}(x^i+t^i), na\right) \overset{(b)}{=} \mathrm{mod}\left(\sum_{i=1}^{n}x^i, na\right) = \sum_{i=1}^{n}x^i. \end{aligned} \tag{5.10}$$

Recall, (a) follows from Remark 3, (b) follows from $\sum_{i=1}^{n} t^i = 0$, and final equality follows from $x^i \in [0, a)$, $\sum_{i=1}^{n} x^i \in [0, na)$ and Definition 5. We have proved TITAN computes the exact aggregate and consequently the average.

(2) The Top-k protocol involves $T = \tilde{\delta} \geq \delta(\mathfrak{G})$ iterations where nodes share the largest k values that they have encountered. From Theorem 8, Top-k converges to the largest k perturbed inputs. We need to run $\lceil n/k \rceil$ iterations of Top-k consensus. Consequently, the total iterations for complete execution is $T\lceil n/k \rceil$. □

5.6.2 Privacy Analysis

The privacy analysis presented here is similar in structure to [117, 118]. The key difference lies in the graph condition required for privacy. Similar to the work in [117] on undirected graphs, we observe that perturbations can be written using individual random numbers exchanged between the nodes, denoted by $\mathbf{r}$, and the incidence matrix B for directed graphs too. This observation helps us re-use several results from [117]. Recall, the noise shared on edge $e = (i, j) \in \mathcal{E}$, is denoted as r^{ij} and it is uniformly distributed over $[0, na)$. Perturbations t^i constructed using (5.4) can be written as $\mathbf{t} = \mathrm{mod}(B\mathbf{r}, na)$, where B is the incidence matrix of $\mathfrak{G}$ and $\mathbf{r}$ is the vector of r^{ij} ordered according to the edge ordering in columns of B. If $\mathfrak{G}$ is strongly connected then $\mathfrak{G}$ is weakly connected and $\overline{\mathfrak{G}}$ is connected.

Lemma 15. *If $\overline{\mathfrak{G}}$ is connected then $\mathbf{t}$ is uniformly distributed over the set*

$$\{\mathbf{q} | \mathbf{q} \in [0, na)^n \text{ and } \mathrm{mod}(\mathbf{1}^T\mathbf{q}, na) = 0\}.$$

Proof. We assume $\mathfrak{G}$ is weakly connected or $\overline{\mathfrak{G}}$ is connected. Consequently, $\mathfrak{G}$ has at least $n-1$ directed edges. Also recall that B is the incidence matrix of graph $\mathfrak{G}$ and rank $(B) = n - 1$ following Theorem 8.3.1, [78]. The rank condition allows us to infer – (a) $\exists$ $n-1$ columns of B that are linearly independent and (b) $B\mathbf{r}$ spans a $(n-1)-$dimensional subspace of $[0, na)^n$.

Let $\mathfrak{G}^0 = (\mathcal{V}, \mathcal{E}^0)$ be the graph with $\mathcal{V}$ nodes and $\mathcal{E}^0 \subset \mathcal{E}$ be a set of $n-1$ edges such that the columns of B-matrix corresponding to edges in $\mathcal{E}^0$ are linearly independent. Such a set $\mathcal{E}^0$ exists since rank $(B) = n-1$. Let B^0 be the incidence matrix of graph $\mathfrak{G}^0$. By construction of $\mathcal{E}^0$, rank $(B^0) = n-1$ and the columns of B^0 are linearly independent. Recall, rank (B) = rank $(B^0) = n-1$ and consequently the columns of B can

be written as a linear combination of columns of matrix B^0. We express the perturbation vector $\mathbf{t}$ as,

$$\mathbf{t} \overset{(a)}{=} \text{mod}(B\mathbf{r}, na) \tag{5.11}$$

$$\overset{(b)}{=} \text{mod}\left(\sum_{e\in\mathcal{E}} B_{*,e}\mathbf{r}_e, na\right) \tag{5.12}$$

$$\overset{(c)}{=} \text{mod}\left(\sum_{e\in\mathcal{E}^0} B_{*,e}\mathbf{r}_e + \sum_{g\in\mathcal{E}\setminus\mathcal{E}^0} B_{*,g}\mathbf{r}_g, na\right) \tag{5.13}$$

$$\overset{(d)}{=} \text{mod}\left(\sum_{e\in\mathcal{E}^0} B_{*,e}\mathbf{r}_e + \sum_{g\in\mathcal{E}\setminus\mathcal{E}^0} \left(\sum_{e\in\mathcal{E}^0} \mu_e^g B_{*,e}\right)\mathbf{r}_g, na\right) \tag{5.14}$$

$$\overset{(e)}{=} \text{mod}\left(\sum_{e\in\mathcal{E}^0} B_{*,e}\left(\mathbf{r}_e + \sum_{g\in\mathcal{E}\setminus\mathcal{E}^0} \mu_e^g\ \mathbf{r}_g\right), na\right) \tag{5.15}$$

$$\overset{(f)}{=} \text{mod}\left(\sum_{e\in\mathcal{E}^0} B_{*,e}\ \underbrace{\text{mod}\left(\mathbf{r}_e + \sum_{g\in\mathcal{E}\setminus\mathcal{E}^0} \mu_e^g\ \mathbf{r}_g, na\right)}_{\sim\ \mathcal{U}[0,na)^{n-1}}, na\right) \tag{5.16}$$

$$\overset{(g)}{=} \text{mod}\left(\sum_{e\in\mathcal{E}^0} B_{*,e}\mathbf{q}_e, na\right) \tag{5.17}$$

$$\overset{(h)}{=} \text{mod}\left(\sum_{e\in\mathcal{E}^0} B^0_{*,e}\mathbf{q}_e, na\right), \text{ where } \mathbf{q}\sim\mathcal{U}[0,na)^{n-1}. \tag{5.18}$$

Note, (a) follows from the definition of perturbations from (5.4), (b) follows directly from the definition of matrix and vector multiplication, and (c) follows from (b) by separately collecting all perturbations corresponding to edges in $\mathcal{E}^0$ and edges in $\mathcal{E}\setminus\mathcal{E}^0$.

Recall that each element in $\mathbf{r}$ is drawn from $\mathcal{U}[0,na)$. Moreover, (d) follows the fact that columns of B that correspond to edges not in $\mathcal{E}^0$ can be expressed as a linear combination of columns of B corresponding to edges in $\mathcal{E}^0$. That is, $B_{*,g} = \sum_{e\in\mathcal{E}^0}\mu_e^g B_{*,e},\ \forall g\in\mathcal{E}\setminus\mathcal{E}^0$. (e) follows from algebraic simplification of equality (5.14). We get (f) by using Remark 3 on (5.15). Specifically we use $\text{mod}(\sum_i y^i, na) = \text{mod}(\sum_i \text{mod}(y^i, na), na)$ over terms in (5.15) to get (f). Finally, we define $\mathbf{q}_e = \text{mod}\left(\mathbf{r}_e + \sum_{g\in\mathcal{E}\setminus\mathcal{E}^0}\mu_e^g\ \mathbf{r}_g, na\right)$ for each $e\in\mathcal{E}^0$ in (5.16) to get equality (g). Moreover, we prove $\mathbf{q}\sim\mathcal{U}[0,na)^{n-1}$ next.

We use the fact that $B_{i,e}\in\{-1,0,1\}$ to observe that $\mu_e^g\in\{-1,0,1\}$. Consequently, we can show $\mathbf{r}_e + \sum_{g\in\mathcal{E}\setminus\mathcal{E}^0}\mu_e^g\ \mathbf{r}_g$ involves addition or subtraction of uniformly random variables $\mathbf{r}_e$ and $\mathbf{r}_g$ for $g\in\mathcal{E}\setminus\mathcal{E}^0$. Next we use Remark 3 to get,

$$\text{mod}\left(\mathbf{r}_e + \sum_{g\in\mathcal{E}\setminus\mathcal{E}^0}\mu_e^g\ \mathbf{r}_g, na\right) = \text{mod}\left(\text{mod}(\mathbf{r}_e, na) + \sum_{g\in\mathcal{E}\setminus\mathcal{E}^0}\text{mod}(\mu_e^g\ \mathbf{r}_g, na), na\right). \tag{5.19}$$

Note that $\mathbf{r}_g \sim \mathcal{U}[0, na)$ leading to $\text{mod}(\mathbf{r}_g, na) = \mathbf{r}_g$. In the right-hand side expression we use Remark 3 as follows, if $\mu_e^g = -1$, then

$$\text{mod}(\mu_e^g \mathbf{r}_g, na) = \begin{cases} -\mathbf{r}_g & \text{if } \mathbf{r}_g = 0, \\ na - \mathbf{r}_g & \text{if } \mathbf{r}_g \in (0, na). \end{cases}$$

Consequently, if $\mu_e^g = -1$ then $\text{mod}(\mu_e^g \mathbf{r}_g, na)$ is uniformly distributed over $\mathcal{U}[0, na)$. This is also trivially true if $\mu_e^g = 1$. The right-hand side expression in (5.19) is an addition of several uniformly distributed random variables. These random variables are uniformly distributed over the set $[0, na)$. Finally, we note that taking modulo over a sum of uniformly random variables gives a uniform random variable. This allows us to show that each $\mathbf{q}_e$ for all $e \in \mathcal{E}^0$ is uniformly distributed over $[0, na)$. Note that $\mathbf{r}_e$ is independent for each $e \in \mathcal{E}^0$ and consequently $\mathbf{q}_e$ is independent for each $e \in \mathcal{E}^0$. Consequently, $\mathbf{q} \sim \mathcal{U}[0, na)^{n-1}$.

Finally, recall $B_{*,e} = B^0_{*,e}$ for each $e \in \mathcal{E}^0$. Thus, (h) follows from the definition of B^0 matrix or $\mathcal{E}^0$ edge set. Thus we can state that perturbations used by TITAN can be written as $\mathbf{t} = \text{mod}(B^0\mathbf{q}, na)$, where $\mathbf{q} \sim [0, na)^{n-1}$.

Finally, we will show that if $\mathbf{t} = \text{mod}(B^0\mathbf{q}, na)$ with $\mathbf{q} \sim [0, na)^{n-1}$, then $\mathbf{t}$ is uniformly distributed over those vectors in $[0, na)^n$ that satisfy $\text{mod}(1^T\mathbf{t}, na) = 0$ in the following claim. This claim completes the correctness of Lemma 15.

Claim 16. *Recall the definition of $\mathcal{E}^0$ and B^0 presented at the begining of proof. If $\mathbf{t} = \text{mod}(B^0\mathbf{q}, na)$ and $\mathbf{q} \sim \mathcal{U}[0, na)^{n-1}$ then, $\mathbf{t}$ is uniformly distributed over $[0, na)^n$ subject to constraint $\text{mod}(1^T\mathbf{t}, na) = 0$.*

Proof. We will first show that, $\mathbf{t} = \text{mod}(B^0\mathbf{q}, na) = 0$ if and only if $\mathbf{q} = 0_{n-1}$. We use this to show that for each $\mathbf{q} \sim \mathcal{U}[0, na)^{n-1}$ there exists a unique $\mathbf{t} \sim \mathcal{U}[0, na)^n$ subject to $\text{mod}(1^T\mathbf{t}, na) = 0$. Consequently, as $\mathbf{q} \sim \mathcal{U}[0, na)^{n-1}$ we show that $\mathbf{t}$ is uniformly distributed over $[0, na)^n$ subject to $\text{mod}(1^T\mathbf{t}, na) = 0$.

(1) We prove - If $\mathbf{q} = 0_{n-1}$ then $\mathbf{t} = \text{mod}(B^0\mathbf{q}, na) = 0_n$.
Proof - If $\mathbf{q} = 0_{n-1}$, then $\mathbf{t} = \text{mod}(B^0\mathbf{q}, na) = \text{mod}(0_n, na) = 0$. QED.

(2) We prove - If $\mathbf{t} = \text{mod}(B^0\mathbf{q}, na) = 0_n$ then $\mathbf{q} = 0_{n-1}$.
Proof - $\mathbf{t} = \text{mod}(B^0\mathbf{q}, na) = 0_n \implies \sum_{e \in \mathcal{E}^0} B^0_{i,e}\mathbf{q}_e = k_i na$ where k_i is an integer. Recall, $B^0_{i,e} = \{-1, 0, 1\}$ and $\mathfrak{G}^0$ has $n-1$ edges and $\mathfrak{G}^0$ is acyclic graph. Consequently, $\mathbf{q}_e = 0$ for all edges e incident to or from a leaf node. We recursively move from the leaf nodes to their parent nodes. Each such parent node has on non-leaf neighbor and $\mathbf{q}_e$ corresponding to the edge corresponding to the non-leaf neighbor also has to be 0. At the root node, all incident edges (to and from) are already fixed to 0. Consequently, $\mathbf{t} = 0_n \implies \mathbf{q}_e = 0_{n-1}$. QED.

Next, we prove that $\text{mod}(B^0\mathbf{q}^1, na) = \text{mod}(B^0\mathbf{q}^2, na)$ if and only if $\mathbf{q}^1 = \mathbf{q}^2$. Observe that,

$$\begin{aligned}
\text{mod}(B^0\mathbf{q}^1, na) &= \text{mod}(B^0\mathbf{q}^2, na) \\
\text{mod}(B^0\mathbf{q}^1, na) - \text{mod}(B^0\mathbf{q}^1, na) &= \text{mod}(B^0\mathbf{q}^2, na) - \text{mod}(B^0\mathbf{q}^1, na) \\
0 &= \text{mod}(B^0\mathbf{q}^2, na) - \text{mod}(B^0\mathbf{q}^1, na) \\
0 &= \text{mod}(\text{mod}(B^0\mathbf{q}^2, na) - \text{mod}(B^0\mathbf{q}^1, na), na) \\
0 &= \text{mod}(B^0(\mathbf{q}^2 - \mathbf{q}^1), na). \qquad (5.20)
\end{aligned}$$

From the above statement (5.20) and the fact that $\mathbf{t} = 0_n \iff \mathbf{q} = 0_{n-1}$, we get, $\text{mod}(B^0\mathbf{q}^1, na) = \text{mod}(B^0\mathbf{q}^2, na)$ if and only if $\mathbf{q}^1 = \mathbf{q}^2$. As a consequence we can state that for each $\mathbf{q}$ there is a unique mapping of $\mathbf{t}$. And as $\mathbf{q}$ is uniformly distributed over $[0, na)^{n-1}$, we have $\mathbf{t}$ is uniformly distributed over $[0, na)^n$ subject to $\text{mod}(1^T\mathbf{t}, na) = 0$. This completes the proof of Claim 16. □

Claim 16 proves that $\mathbf{t}$ is uniformly distributed over $[0, na)^{n-1}$ subject to $\text{mod}(\mathbf{1}^T\mathbf{t}, na) = 0$. This concludes the proof of Lemma 15. □

Using the above property of perturbations, $\mathbf{t}$, we can show that perturbed inputs appear to be uniformly random, as described in the next lemma.

Lemma 17. *If $\overline{\mathfrak{G}}$ is connected then the effective inputs $\tilde{\mathbf{x}}$ are uniformly distributed over the set*

$$\{\mathbf{q} | \mathbf{q} \in [0, na)^n \text{ and } \text{mod}(\mathbf{1}^T\mathbf{q}, na) = \sum_{i=1}^{n} x^i\}.$$

Proof. Let $\tilde{X}, X$ and T represent the random vectors of agents obfuscated inputs, private inputs and perturbations respectively. Let $f_{\tilde{X}}, f_X$ and f_T denote the probability distribution of the respective random variables.

Recall $\tilde{x}^i = \text{mod}\left((x^i + t^i), na\right)$ and the fact that t^i and x^i are independent. We have,

$$f_{\tilde{X}}(\tilde{\mathbf{x}} | X = \mathbf{x}) = f_T(T = \mathbf{t} = \text{mod}((\tilde{\mathbf{x}} - \mathbf{x}), na)).$$

As $\overline{\mathfrak{G}}$ is connected, following Lemma 15, we have $\mathbf{t} = \text{mod}((\tilde{\mathbf{x}} - \mathbf{x}), na)$ is uniformly distributed over $[0, na)^n$, subject to $\text{mod}(\sum_{i=1}^{n} t^i, na) = 0$. That is, we have $f_T(T = \text{mod}((\tilde{\mathbf{x}} - \mathbf{x}), na)) = \text{constant}$ for any $\tilde{\mathbf{x}} \in [0, na)^n$, given $\text{mod}(1^T\tilde{\mathbf{x}}, na) = 1^T\mathbf{x}$.

We have that $f_{\tilde{X}}(\tilde{\mathbf{x}} | X = \mathbf{x}) = \text{constant}$ for all $\tilde{\mathbf{x}} \in [0, na)^n$ implying that the perturbed input appears to be uniformly distributed over $[0, na)^n$ subject to $\text{mod}(1^T\tilde{\mathbf{x}}, na) = 1^T\mathbf{x}$. □

Recall, $\mathcal{A}$ is the set of honest-but-curious adversaries with $|\mathcal{A}| \leq \tau$. Also recall, adversarial nodes observe/store all information directly received and transmitted.

Additional Notation: Let $\mathcal{H} = \mathcal{V} \setminus \mathcal{A}$ denote the set of honest nodes and $\mathcal{E}_{\mathcal{H}} \subseteq \mathcal{E}$ is the set of all edges from $\mathfrak{G}$ that are between two honest nodes. Let the incidence matrix be partitioned as $B = [B_{\mathcal{H}}\ B_{\mathcal{A}}]$, where $B_{\mathcal{H}}$ are columns of B corresponding to edges in $\mathcal{E}_{\mathcal{H}}$ and $B_{\mathcal{A}}$ are columns of B corresponding to edges in $\mathcal{E}_{\mathcal{A}} = \mathcal{E} \setminus \mathcal{E}_{\mathcal{H}}$. Let B_e represent the e^{th} column of B, $B_{i,e}$ represent the $[i,e]^{th}$ entry of matrix B and $\mathbf{r}_e$ denote the e^{th} entry of vector $\mathbf{r}$.

Proof of Theorem 10. Let $\mathfrak{G}_{\mathcal{H}} = (\mathcal{H}, \mathcal{E}_{\mathcal{H}})$ denote the subgraph induced by honest nodes. Let $B_{\mathcal{H}}$ denote the oriented incidence matrix of graph $\mathfrak{G}_{\mathcal{H}}$. If vertex connectivity $\kappa(\overline{\mathfrak{G}}) \geq \tau + 1$, implying that deleting any τ nodes from $\overline{\mathfrak{G}}$ does not disconnect the graph, then, $\overline{\mathfrak{G}}_{\mathcal{H}}$ is connected.

The information accessible to an adversary, defined as $\mathsf{View}_{\mathcal{A}}$, consists of the private inputs of corrupted agents, perturbed inputs of honest agents and the random numbers transmitted or received by corrupted nodes. We denote it as,

$$\mathsf{View}_{\mathcal{A}}(x) = \{\{\tilde{x}^i | i \in \mathcal{H}\}, \{x^i | i \in \mathcal{A}\}, \{r_e | e \in \mathcal{E}_{\mathcal{A}}\}\},$$

where $\mathcal{E}_{\mathcal{A}} = \mathcal{E} \setminus \mathcal{E}_{\mathcal{H}}$.

In order to prove privacy as per Definition 3, we intend to prove that the probability distributions of $\mathsf{View}_{\mathcal{A}}(x)$ and $\mathsf{View}_{\mathcal{A}}(y)$ are identical, for any two inputs x, y such that $x^i = y^i$ for all $i \in \mathcal{A}$ and $\sum_{i\in\mathcal{V}} x^i = \sum_{i\in\mathcal{V}} y^i$.

First, we note that the perturbations can be expressed as, $t^i = \mathrm{mod}\left(\sum_{e\in\mathcal{E}_{\mathcal{H}}\cup\mathcal{E}_{\mathcal{A}}} B_{i,e}\mathbf{r}_e, na\right)$. Using Remark 3 we can equivalently write,

$$t^i = \mathrm{mod}(t^i_{\mathcal{H}} + t^i_{\mathcal{A}}, na), \tag{5.21}$$

where $t^i_{\mathcal{H}} = \mathrm{mod}\left(\sum_{e\in\mathcal{E}_{\mathcal{H}}} B_{i,e}\mathbf{r}_e, na\right)$ and $t^i_{\mathcal{A}} = \mathrm{mod}\left(\sum_{e\in\mathcal{E}_{\mathcal{A}}} B_{i,e}\mathbf{r}_e, na\right)$.

Next we use Lemma 15. As $\overline{\mathfrak{G}}_{\mathcal{H}}$ is connected, we have, $\{t^i_{\mathcal{H}}\}_{i\in\mathcal{H}}$ is uniformly distributed over $[0, na)^{|\mathcal{H}|}$ subject to

$$\mathrm{mod}\left(\sum_{i\in\mathcal{H}} t^i_{\mathcal{H}}, na\right) = 0. \tag{5.22}$$

We can equivalently write $\{t^i_{\mathcal{H}}\}_{i\in\mathcal{H}}$ is uniformly distributed over $[0, na)^{|\mathcal{H}|}$ and subject to

$$\mathrm{mod}(\sum_{i\in\mathcal{H}}(\sum_{e\in\mathcal{E}_{\mathcal{H}}} B_{i,e}\mathbf{r}_e), na) = 0.$$

We also use Lemma 15 on graph $\mathfrak{G}$. As $\overline{\mathfrak{G}}$ is connected, we have, $\{t^i\}_{i\in\mathcal{V}}$ is uniformly distributed over

$[0, na)^n$ subject to,

$$\mod\left(\sum_{i\in\mathcal{H}\cup\mathcal{A}} t^i, na\right) = 0. \tag{5.23}$$

Using (5.21), (5.22), and (5.23) we get,

$$0 = \mod\left(\sum_{i\in\mathcal{H}\cup\mathcal{A}} t^i, na\right) \tag{5.24}$$

$$= \mod\left(\mod\left(\sum_{i\in\mathcal{H}} t^i_{\mathcal{H}} + t^i_{\mathcal{A}}, na\right) + \mod\left(\sum_{i\in\mathcal{A}} t^i_{\mathcal{H}} + t^i_{\mathcal{A}}, na\right), na\right), \tag{5.25}$$

Thus, we can state that the perturbations $\{t^i\}_{i\in\mathcal{H}}$ are uniformly distributed over $[0, na)^{|\mathcal{H}|}$ subject to $\mod(\sum_{i\in\mathcal{H}} t^i, na) = -\mod(\sum_{i\in\mathcal{A}} t^i, na)$. Moreover, recall that $\mathbf{r}_e \sim \mathcal{U}[0, na)$ for each $e \in \mathcal{E}$. Also note that adversary observes $t^i = \mod(\sum_{e\in\mathcal{E}_\mathcal{A}} B_{i,e}\mathbf{r}_e, na)$ for each $i \in \mathcal{A}$.

Using Lemma 17, if $\overline{\mathfrak{G}}_\mathcal{H}$ is connected, we get,

$$f_{\tilde{X}_\mathcal{H}}(\tilde{\mathbf{x}}_\mathcal{H}|\mathbf{x}_\mathcal{H}, \{\mathbf{r}_e|e \in \mathcal{E}_\mathcal{A}\}) = \text{constant},$$

for all $\tilde{\mathbf{x}}_\mathcal{H} \in [0, na)^{|\mathcal{H}|}$ that satisfies $\mod(\sum_{i\in\mathcal{H}} \tilde{\mathbf{x}}_i, ma) = \mod(\sum_{i\in\mathcal{H}} \mathbf{x}_i, na)$.

If $x^i = y^i$ for $i \in \mathcal{A}$ and $\sum_{i\in\mathcal{V}} x^i = \sum_{i\in\mathcal{V}} y^i$, then we can write, using Lemma 17, if $\overline{\mathfrak{G}}_\mathcal{H}$ is connected, we get,

$$f_{\tilde{X}_\mathcal{H}}(\tilde{\mathbf{x}}_\mathcal{H}|\mathbf{y}_\mathcal{H}, \{\mathbf{r}_e|e \in \mathcal{E}_\mathcal{A}\}) = \text{constant},$$

for all $\tilde{\mathbf{x}}_\mathcal{H} \in [0, na)^{|\mathcal{H}|}$ that satisfies $\mod(\sum_{i\in\mathcal{H}} \tilde{x}^i, na) = \mod(\sum_{i\in\mathcal{H}} y^i, na)$. That is,

$$f_{\tilde{X}_\mathcal{H}}(\tilde{\mathbf{x}}_\mathcal{H}|\mathbf{x}_\mathcal{H}, \{\mathbf{r}_e|e \in \mathcal{E}_\mathcal{A}\}) = f_{\tilde{X}_\mathcal{H}}(\tilde{\mathbf{x}}_\mathcal{H}|\mathbf{y}_\mathcal{H}, \{\mathbf{r}_e|e \in \mathcal{E}_\mathcal{A}\}),$$

and equivalently,

$$f_{\mathsf{View}_\mathcal{A}(\mathbf{x})}(\{\tilde{\mathbf{x}}_\mathcal{H}, \{\mathbf{r}_e|e \in \mathcal{E}_\mathcal{A}\}\}) = f_{\mathsf{View}_\mathcal{A}(\mathbf{y})}(\{\tilde{\mathbf{x}}_\mathcal{H}, \{\mathbf{r}_e|e \in \mathcal{E}_\mathcal{A}\}\}).$$

□

5.7 Conclusion

We presented TITAN, a finite-time, private algorithm for solving distributed average consensus. We show that TITAN converges to the average in finite-time that is dependent only on graph diameter and number of agents/nodes. It also protects statistical privacy of inputs against an honest-but-curious adversary that

corrupts at most τ nodes in the network, provided weak vertex-connectivity of graph is at least $\tau + 1$. We use TITAN to solve a horizontally partitioned system of linear equations in finite-time, while, protecting statistical privacy of local equations against an honest-but-curious adversary.

5.8 Proof of Remark 3

Proof. 1. For each $i = 1, \ldots, q$, we have $\text{mod}(y_i, a) = y_i - ak_i$, where k_i is an integer. This follows directly from the definition of modular arithmetic over real numbers (Definition 5). We use this relation to get,

$$\begin{aligned}
\text{mod}\left(\sum_{i=1}^{q} \text{mod}(y_i, a), a\right) &= \text{mod}\left(\sum_{i=1}^{q} (y_i - ak_i), a\right) \\
&= \text{mod}\left(\sum_{i=1}^{q} y_i - a\left(\sum_{i=1}^{q} k_i\right), a\right) \\
&\overset{(a)}{=} \text{mod}\left(\sum_{i=1}^{q} y_i, a\right).
\end{aligned}$$

We get (a) following Definition 5 to get $\text{mod}(x + ka, a) = \text{mod}(x, a)$ for all integers k.

2. Observe that the modulo operator is defined as $\text{mod}(y, a) = y - a\lfloor y/a \rfloor$. Next, we observe the following,

$$\begin{aligned}
\text{mod}(y, a) + \text{mod}(-y, a) &= y - a\lfloor y/a \rfloor + (-y - a\lfloor -y/a \rfloor) \\
&= -a\left(\lfloor y/a \rfloor + \lfloor -y/a \rfloor\right) \\
&\overset{(a)}{=} -a\left(\lfloor y/a \rfloor - (\lfloor y/a \rfloor + 1)\right) \\
&= a,
\end{aligned}$$

where (a) follows from the property of floor functions $(\lfloor \cdot \rfloor)$ and the fact that $y \in (0, a)$. This gives us, $\text{mod}(-y, a) = a - \text{mod}(y, a)$ whenever $y \in (0, a)$.

Quite trivially, we also observe, if $y = 0$, then $\text{mod}(-y, a) = -\text{mod}(y, a) = 0$. □

CHAPTER 6

SUMMARY AND FUTURE RESEARCH DIRECTIONS

6.1 Dissertation Summary

In this dissertation, we explore privacy issues in both peer-to-peer network model and multiple parameter server model for a host of distributed computing problems. Specifically, we focus on algorithms that leverage network architecture and correlated random perturbations to simultaneously guarantee privacy and accuracy.

Distributed Optimization: We consider a distributed optimization problem. Our proposed solution involves a pre-processing step that leverages correlated random functions to obfuscate private objective functions, followed by, a standard distributed optimization problem. We propose FS-GP and FS-DGD for privacy preserving distributed optimization over directed and undirected graphs respectively. We prove deterministic and asymptotic convergence of both algorithms to the exact solution. We also characterize a finite-time rate degradation due to the pre-processing (obfuscation) step. We show that weak vertex connectivity of at least $\tau + 1$ (directed graphs) and vertex connectivity of at least $\tau + 1$ (undirected graphs) are necessary and sufficient graph condition for privacy via non-identifiability of local objective functions.

Network Aggregate Games: We propose a solution that uses locally balanced (correlated) random perturbations to hide private actions/decisions before sharing them with neighbors. Our algorithm converges to the unique Nash equilibrium of the network aggregate game asymptotically. We show that if the graph formed by non-corrupted nodes connected and non-bipartite then any permutation of private cost functions appears the same to an adversary (non-identifiability).

System of Linear Equations: We first propose TITAN, an average consensus algorithm with finite-time convergence and information-theoretic privacy guarantees. Our solver for distributed system of linear equations requires each agent to compute an update based on local equations followed by private aggregation using TITAN. We show that our solution converges in finite time that depends only on the total number of agents and graph diameter.

6.2 Future Research Directions

The Function Sharing algorithms (FS-GP and FS-DGD) proposed in Chapter 2 can guarantee privacy of local objective functions. However, it requires additional work to characterize right set of perturbation or noise functions for highly non-convex optimization problems such as deep learning and reinforcement learning.

Distributed recovery based average consensus algorithms, such as TITAN, provide finite-time convergence over fixed graphs. However, design of recovery based average consensus algorithms over time-varying graphs is not straight forward and needs additional work. Several application areas involve time-varying communication topologies and we need to extend our work to allow for distributed computing over time-varying directed graphs.

REFERENCES

[1] R. Roman, J. Zhou, and J. Lopez, "On the features and challenges of security and privacy in distributed internet of things," *Computer Networks*, vol. 57, no. 10, pp. 2266–2279, 2013.

[2] P. Hallgren, C. Orlandi, and A. Sabelfeld, "Privatepool: Privacy-preserving ridesharing," in *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*. IEEE, 2017, pp. 276–291.

[3] G. A. Kaissis, M. R. Makowski, D. Rückert, and R. F. Braren, "Secure, privacy-preserving and federated machine learning in medical imaging," *Nature Machine Intelligence*, pp. 1–7, 2020.

[4] Y. Wang, S. Adams, P. Beling, S. Greenspan, S. Rajagopalan, M. Velez-Rojas, S. Mankovski, S. Boker, and D. Brown, "Privacy preserving distributed deep learning and its application in credit card fraud detection," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 2018, pp. 1070–1078.

[5] N. Papernot, P. McDaniel, A. Sinha, and M. P. Wellman, "SoK: Security and privacy in machine learning," in *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2018, pp. 399–414.

[6] M. Al-Rubaie and J. M. Chang, "Privacy-preserving machine learning: Threats and solutions," *IEEE Security & Privacy*, vol. 17, no. 2, pp. 49–58, 2019.

[7] D. Fudenberg and J. Tirole, *Game Theory*. MIT Press, 1991.

[8] A. Nedić and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[9] A. Nedić and A. Olshevsky, "Distributed optimization over time-varying directed graphs," *IEEE Transactions on Automatic Control*, vol. 60, no. 3, pp. 601–615, 2015.

[10] S. Mou, J. Liu, and A. S. Morse, "A distributed algorithm for solving a linear algebraic equation," *IEEE Transactions on Automatic Control*, vol. 60, no. 11, pp. 2863–2878, 2015.

[11] J. Liu, S. Mou, and A. S. Morse, "Asynchronous distributed algorithms for solving linear algebraic equations," *IEEE Transactions on Automatic Control*, vol. 63, no. 2, pp. 372–385, 2017.

[12] F. Salehisadaghiani and L. Pavel, "Nash equilibrium seeking by a gossip-based algorithm," in *53rd IEEE Conference on Decision and Control*. IEEE, 2014, pp. 1155–1160.

[13] J. Koshal, A. Nedić, and U. V. Shanbhag, "Distributed algorithms for aggregative games on graphs," *Operations Research*, vol. 64, no. 3, pp. 680–704, 2016.

[14] Q. Ling, Y. Xu, W. Yin, and Z. Wen, "Decentralized low-rank matrix completion," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 2925–2928.

[15] C. Chen, Z. Liu, P. Zhao, J. Zhou, and X. Li, "Privacy preserving point-of-interest recommendation using decentralized matrix factorization," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[16] A.-Y. Lin and Q. Ling, "Decentralized and privacy-preserving low-rank matrix completion," *Journal of the Operations Research Society of China*, vol. 3, no. 2, pp. 189–205, 2015.

[17] H. Zhou, X.-Y. Liu, C. Fu, C. Shang, and X. Chang, "Differentially private matrix completion via distributed matrix factorization," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 2018, pp. 1628–1631.

[18] C. Dwork, "Differential privacy: A survey of results," in *International Conference on Theory and Applications of Models of Computation*. Springer, 2008, pp. 1–19.

[19] Ú. Erlingsson, V. Pihur, and A. Korolova, "Rappor: Randomized aggregatable privacy-preserving ordinal response," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014, pp. 1054–1067.

[20] S. Gade and N. H. Vaidya, "Private optimization on networks," in *2018 Annual American Control Conference (ACC)*. IEEE, 2018, pp. 1402–1409.

[21] N. Gupta, S. Gade, N. Chopra, and N. H. Vaidya, "Preserving statistical privacy in distributed optimization," *IEEE Control Systems Letters*, vol. 5, no. 3, pp. 779–784, 2021.

[22] S. Gade, A. Winnicki, and S. Bose, "On privatizing equilibrium computation in aggregate games over networks," *arXiv preprint arXiv:1912.06296*, 2019.

[23] S. Gade, J. Liu, and N. H. Vaidya, "A private and finite-time algorithm for solving a distributed system of linear equations," *arXiv preprint arXiv:2004.04680*, 2020.

[24] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[25] M. Li, D. G. Andersen, A. Smola, and K. Yu, "Communication efficient distributed machine learning with the parameter server," in *NIPS*, 2014, pp. 1–9.

[26] I. Cano, M. Weimer, D. Mahajan, C. Curino, and G. M. Fumarola, "Towards geo-distributed machine learning," *arXiv preprint arXiv:1603.09035*, 2016.

[27] S. Gade and N. H. Vaidya, "Privacy-preserving distributed learning via obfuscated stochastic gradients," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 184–191.

[28] T. T. Doan and A. Olshevsky, "Distributed resource allocation on dynamic networks in quadratic time," *Systems & Control Letters*, vol. 99, pp. 57–63, 2017.

[29] L. Xiao, S. Boyd, and S.-J. Kim, "Distributed average consensus with least-mean-square deviation," *Journal of Parallel and Distributed Computing*, vol. 67, no. 1, pp. 33–46, 2007.

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

[31] H. Khurana, M. Hadley, N. Lu, and D. A. Frincke, "Smart-grid security issues," *IEEE Security & Privacy*, vol. 8, no. 1, 2010.

[32] T. Kraska, A. Talwalkar, J. C. Duchi, R. Griffith, M. J. Franklin, and M. I. Jordan, "MLbase: A distributed machine-learning system." in *CIDR*, vol. 1, 2013.

[33] M. Li, D. G. Andersen, A. J. Smola, and K. Yu, "Communication efficient distributed machine learning with the parameter server," in *NIPS*, 2014, pp. 19–27.

[34] K. Hsieh, A. Harlap, N. Vijaykumar, D. Konomis, G. R. Ganger, P. B. Gibbons, and O. Mutlu, "Gaia: Geo-distributed machine learning approaching lan speeds." in *NSDI*, 2017, pp. 629–647.

[35] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 1310–1321.

[36] S. Gade and N. H. Vaidya, "Private learning on networks," *arXiv preprint arXiv:1612.05236*, 2016.

[37] A. Nedić and A. Ozdaglar, "On the rate of convergence of distributed subgradient methods for multi-agent optimization," *Proceedings of the IEEE Conference on Decision and Control*, vol. 54, no. 1, pp. 4711–4716, 2007.

[38] S. S. Ram, A. Nedić, and V. V. Veeravalli, "Distributed stochastic subgradient projection algorithms for convex optimization," *Journal of Optimization Theory and Applications*, vol. 147, no. 3, pp. 516–545, 2010.

[39] J. C. Duchi, A. Agarwal, and M. J. Wainwright, "Dual averaging for distributed optimization: Convergence analysis and network scaling," *IEEE Transactions on Automatic Control*, vol. 57, no. 3, pp. 592–606, 2012.

[40] S. S. Ram, A. Nedić, and V. V. Veeravalli, "Incremental stochastic subgradient algorithms for convex optimization," *SIAM Journal on Optimization*, vol. 20, no. 2, pp. 691–717, 2009.

[41] D. Jakovetic, J. Xavier, and J. M. Moura, "Fast distributed gradient methods," *IEEE Transactions on Automatic Control*, vol. 59, no. 5, pp. 1131–1146, 2014.

[42] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, "On the linear convergence of the ADMM in decentralized consensus optimization," *IEEE Transactions on Signal Processing*, vol. 62, no. 7, pp. 1750–1761, 2014.

[43] W. Shi, Q. Ling, G. Wu, and W. Yin, "Extra: An exact first-order algorithm for decentralized consensus optimization," *SIAM Journal on Optimization*, vol. 25, no. 2, pp. 944–966, 2015.

[44] C. Xi and U. A. Khan, "On the linear convergence of distributed optimization over directed graphs," *arXiv:1510.02149*, 2015.

[45] C. N. Hadjicostis, N. H. Vaidya, and A. D. Domínguez-García, "Robust distributed average consensus via exchange of running sums," *IEEE Transactions on Automatic Control*, vol. 61, no. 6, pp. 1492–1507, 2016.

[46] A. Nedić, "Asynchronous broadcast-based convex optimization over a network," *IEEE Transactions on Automatic Control*, vol. 56, no. 6, pp. 1337–1351, 2011.

[47] J. Liu and S. J. Wright, "Asynchronous stochastic coordinate descent: Parallelism and convergence properties," *SIAM Journal on Optimization*, vol. 25, no. 1, pp. 351–376, 2015.

[48] E. Wei and A. Ozdaglar, "On the O(1/k) convergence of asynchronous distributed alternating direction method of multipliers," in *2013 Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE, 2013, pp. 551–554.

[49] L. Su and N. H. Vaidya, "Fault-tolerant multi-agent optimization: Optimal iterative distributed algorithms," in *Proceedings of the ACM Symposium on Principles of Distributed Computing*, 2016, pp. 425–434.

[50] J. Vaidya, "Privacy-preserving linear programming," in *Proceedings of the 2009 ACM symposium on Applied Computing*. ACM, 2009, pp. 2002–2007.

[51] B. Pinkas, "Cryptographic techniques for privacy-preserving data mining," *ACM SIGKDD Expl. Newsletter*, vol. 4, no. 2, pp. 12–19, 2002.

[52] Y. Lu and M. Zhu, "Privacy preserving distributed optimization using homomorphic encryption," *arXiv preprint arXiv:1805.00572*, 2018.

[53] A. B. Alexandru, K. Gatsis, and G. J. Pappas, "Privacy preserving cloud-based quadratic optimization," in *2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2017, pp. 1168–1175.

[54] A. B. Alexandru, K. Gatsis, Y. Shoukry, S. A. Seshia, P. Tabuada, and G. J. Pappas, "Cloud-based quadratic optimization with partially homomorphic encryption," *arXiv preprint arXiv:1809.02267*, 2018.

[55] C. Zhang, M. Ahmad, and Y. Wang, "ADMM based privacy-preserving decentralized optimization," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 3, pp. 565–580, 2019.

[56] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," *arXiv preprint arXiv:1607.00133*, 2016.

[57] B. Ding, J. Kulkarni, and S. Yekhanin, "Collecting telemetry data privately," in *Advances in Neural Information Processing Systems*, 2017, pp. 3571–3580.

[58] Z. Huang, S. Mitra, and N. Vaidya, "Differentially private distributed optimization," in *Proceedings of the 2015 International Conference on Distributed Computing and Networking*. ACM, 2015, p. 4.

[59] M. T. Hale and M. Egerstedt, "Cloud-enabled differentially private multiagent optimization with constraints," *IEEE Transactions on Control of Network Systems*, vol. 5, no. 4, pp. 1693–1706, 2017.

[60] E. Nozari, P. Tallapragada, and J. Cortés, "Differentially private distributed convex optimization via functional perturbation," *IEEE Transactions on Control of Network Systems*, vol. 5, no. 1, pp. 395–408, 2018.

[61] S. Han, U. Topcu, and G. J. Pappas, "Differentially private distributed constrained optimization," *IEEE Transactions on Automatic Control*, vol. 62, no. 1, pp. 50–64, 2016.

[62] J. Hamm, Y. Cao, and M. Belkin, "Learning privately from multiparty data," in *Proceedings of The 33rd International Conference on Machine Learning*, 2016, pp. 555–563.

[63] J. He, L. Cai, C. Zhao, P. Cheng, and X. Guan, "Privacy-preserving average consensus: Privacy analysis and algorithm design," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 5, no. 1, pp. 127–138, 2018.

[64] Y. Mo and R. M. Murray, "Privacy preserving average consensus," *IEEE Transactions on Automatic Control*, vol. 62, no. 2, pp. 753–765, 2017.

[65] O. L. Mangasarian, "Privacy-preserving horizontally partitioned linear programs," *Optimization Letters*, vol. 6, no. 3, pp. 431–436, 2012.

[66] C. Wang, K. Ren, and J. Wang, "Secure and practical outsourcing of linear programming in cloud computing," in *2011 Proceedings of INFOCOM*. IEEE, 2011, pp. 820–828.

[67] S. Gade and N. H. Vaidya, "Distributed optimization of convex sum of non-convex functions," *arXiv preprint arXiv:1608.05401*, 2016.

[68] K. Kvaternik and L. Pavel, "Lyapunov analysis of a distributed optimization scheme," in *NetGCooP-2011*. IEEE, 2011, pp. 1–5.

[69] S. Gade and N. H. Vaidya, "Private learning on networks: Part ii," *arXiv preprint arXiv:1703.09185*, 2017.

[70] N. Gupta, "Privacy in distributed multi-agent collaboration: Consensus and optimization," Ph.D. dissertation, University of Maryland, College Park, 2018.

[71] Y. Liu, J. Wu, I. R. Manchester, and G. Shi, "Gossip algorithms that preserve privacy for distributed computation part i: The algorithms and convergence conditions," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 4499–4504.

[72] Y. Liu, J. Wu, I. R. Manchester, and G. Shi, "Gossip algorithms that preserve privacy for distributed computation part ii: Performance against eavesdroppers," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 5346–5351.

[73] E. A. Abbe, A. E. Khandani, and A. W. Lo, "Privacy-preserving methods for sharing financial risk exposures," *The American Economic Review*, vol. 102, no. 3, pp. 65–70, 2012.

[74] T. W. Judson, *Abstract Algebra*. Stephen F. Austin State University, 2010.

[75] H. Robbins and D. Siegmund, "A convergence theorem for non negative almost supermartingales and some applications," in *Herbert Robbins Selected Papers*. Springer, 1985, pp. 111–135.

[76] S. Abbott et al., *Understanding Analysis*. Springer, 2015.

[77] L. Xiao, S. Boyd, and S. Lall, "Distributed average consensus with time-varying metropolis weights," *Automatica*, 2006.

[78] C. Godsil and G. F. Royle, *Algebraic Graph Theory*. Springer Science & Business Media, 2013, vol. 207.

[79] R. Diestel, *Graph Theory*. Springer, 2005, vol. 101.

[80] R. B. Bapat, *Graphs and Matrices*. Springer, 2010.

[81] D. Zelazo, A. Rahmani, and M. Mesbahi, "Agreement via the edge Laplacian," in *2007 46th IEEE Conference on Decision and Control*. IEEE, 2007, pp. 2309–2314.

[82] A. Nedic, A. Olshevsky, A. Ozdaglar, and J. N. Tsitsiklis, "Distributed subgradient methods and quantization effects," in *47th IEEE Conference on Decision and Control (CDC)*. IEEE, 2008, pp. 4177–4184.

[83] J. A. Fax and R. M. Murray, "Information flow and cooperative control of vehicle formations," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1465–1476, 2004.

[84] J. Konecny, B. McMahan, and D. Ramage, "Federated optimization: Distributed optimization beyond the datacenter," *arXiv preprint arXiv:1511.03575*, 2015.

[85] S. Gade and N. H. Vaidya, "Distributed optimization for client-server architecture with negative gradient weights," *arXiv preprint arXiv:1608.03866*, 2016.

[86] S. Gade and N. H. Vaidya, "Privacy-preserving distributed learning via obfuscated stochastic gradients," University of Illinois at Urbana-Champaign, Tech. Rep., 2018. [Online]. Available: http://publish.illinois.edu/shripadgade/files/2020/03/TR5.pdf

[87] D. P. Bertsekas, A. Nedić, and A. E. Ozdaglar, *Convex Analysis and Optimization*. Athena Scientific, 2003.

[88] L. Xiao and S. Boyd, "Optimal scaling of a gradient method for distributed resource allocation," *Journal of Optimization Theory and Applications*, vol. 129, no. 3, pp. 469–488, 2006.

[89] W. Novshek, "On the existence of Cournot equilibrium," *The Review of Economic Studies*, vol. 52, no. 1, pp. 85–98, 1985.

[90] M. K. Jensen, "Aggregative games and best-reply potentials," *Economic Theory*, vol. 43, no. 1, pp. 45–66, 2010.

[91] B. Willems, I. Rumiantseva, and H. Weigt, "Cournot versus supply functions: What does the data tell us?" *Energy Economics*, vol. 31, no. 1, pp. 38–47, 2009.

[92] D. Cai, S. Bose, and A. Wierman, "On the role of a market maker in networked Cournot competition," *Mathematics of Operations Research*, vol. 44, no. 3, pp. 1122–1144, 2019.

[93] A. Cherukuri and J. Cortés, "Iterative bidding in electricity markets: Rationality and robustness," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 3, pp. 1265–1281, 2020.

[94] Z.-j. Teng, L.-y. Xie, H.-l. Chen, L.-x. Teng, and H.-b. Li, "Application research of game theory in cognitive radio spectrum allocation," *Wireless Networks*, vol. 25, no. 7, pp. 4275–4286, 2019.

[95] S. Koskie and Z. Gajic, "A Nash game algorithm for SIR-based power control in 3G wireless CDMA networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 13, no. 5, pp. 1017–1026, 2005.

[96] D. Martimort and L. Stole, "Aggregate representations of aggregate games," 2011.

[97] F. Salehisadaghiani and L. Pavel, "Distributed Nash equilibrium seeking in networked graphical games," *Automatica*, vol. 87, pp. 17–24, 2018.

[98] M. Ye and G. Hu, "Distributed Nash equilibrium seeking by a consensus based approach," *IEEE Transactions on Automatic Control*, vol. 62, no. 9, pp. 4811–4818, 2017.

[99] T. Tatarenko, W. Shi, and A. Nedić, "Accelerated gradient play algorithm for distributed Nash equilibrium seeking," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 3561–3566.

[100] F. Parise, B. Gentile, S. Grammatico, and J. Lygeros, "Network aggregative games: Distributed convergence to Nash equilibria," in *2015 54th IEEE Conference on Decision and Control (CDC)*. IEEE, 2015, pp. 2295–2300.

[101] F. Facchinei and J.-S. Pang, *Finite-dimensional Variational Inequalities and Complementarity Problems*. Springer Science & Business Media, 2007.

[102] R. Dong, W. Krichene, A. M. Bayen, and S. S. Sastry, "Differential privacy of populations in routing games," in *2015 54th IEEE Conference on Decision and Control (CDC)*. IEEE, 2015, pp. 2798–2803.

[103] R. Cummings, M. Kearns, A. Roth, and Z. S. Wu, "Privacy and truthful equilibrium selection for aggregative games," in *International Conference on Web and Internet Economics*. Springer, 2015, pp. 286–299.

[104] Y. Lu and M. Zhu, "Game-theoretic distributed control with information-theoretic security guarantees," *IFAC-PapersOnLine*, vol. 48, no. 22, pp. 264–269, 2015.

[105] F. R. Chung and F. C. Graham, *Spectral Graph Theory*. American Mathematical Soc., 1997, no. 92.

[106] K. A. Ross, *Elementary Analysis: The Theory of Calculus*. Springer Science & Business Media, 2013.

[107] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *Fourth International Symposium on Information Processing in Sensor Networks*. IEEE, 2005, pp. 63–70.

[108] S. Kar, J. M. Moura, and K. Ramanan, "Distributed parameter estimation in sensor networks: Nonlinear observation models and imperfect communication," *IEEE Transactions on Information Theory*, vol. 58, no. 6, pp. 3575–3605, 2012.

[109] G. Williams, *Linear Algebra with Applications*. Jones & Bartlett Learning, 2017.

[110] J. Liu, A. S. Morse, A. Nedić, and T. Başar, "Exponential convergence of a distributed algorithm for solving linear algebraic equations," *Automatica*, vol. 83, pp. 37–46, 2017.

[111] T. Yang, J. George, J. Qin, X. Yi, and J. Wu, "Distributed least squares solver for network linear equations," *Automatica*, vol. 113, pp. 1087–98, 2020.

[112] D. Jakovetic, N. Krejic, N. K. Jerinkic, G. Malaspina, and A. Micheletti, "Distributed fixed point method for solving systems of linear algebraic equations," *arXiv preprint arXiv:2001.03968*, 2020.

[113] P. Wang, S. Mou, J. Lian, and W. Ren, "Solving a system of linear equations: From centralized to distributed algorithms," *Annual Reviews in Control*, 2019.

[114] S. S. Alaviani and N. Elia, "A distributed algorithm for solving linear algebraic equations over random networks," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 83–88.

[115] A. Nedić, A. Ozdaglar, and P. A. Parrilo, "Constrained consensus and optimization in multi-agent networks," *IEEE Transactions on Automatic Control*, vol. 55, no. 4, pp. 922–938, 2010.

[116] A. Gascón, P. Schoppmann, B. Balle, M. Raykova, J. Doerner, S. Zahur, and D. Evans, "Privacy-preserving distributed linear regression on high-dimensional data," *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 4, pp. 345–364, 2017.

[117] N. Gupta, J. Kat, and N. Chopra, "Statistical privacy in distributed average consensus on bounded real inputs," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 1836–1841.

[118] N. Gupta, J. Katz, and N. Chopra, "Privacy in distributed average consensus," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 9515–9520, 2017.

[119] C. N. Hadjicostis, N. H. Vaidya, and A. D. Domínguez-García, "Robust distributed average consensus via exchange of running sums," *IEEE Transactions on Automatic Control*, vol. 61, no. 6, pp. 1492–1507, 2016.

[120] J. Tsitsiklis, "Problems in decentralized decision making and computation," Ph.D. dissertation, Massachusetts Institute of Technology, 1984.

[121] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *44th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, 2003, pp. 482–491.

[122] G. Oliva, R. Setola, and C. N. Hadjicostis, "Distributed finite-time average-consensus with limited computational and storage capability," *IEEE Transactions on Control of Network Systems*, vol. 4, no. 2, pp. 380–391, 2016.

[123] A. Nedić, A. Olshevsky, and W. Shi, "Achieving geometric convergence for distributed optimization over time-varying graphs," *SIAM Journal on Optimization*, vol. 27, no. 4, pp. 2597–2633, 2017.

[124] Y. Sun, A. Daneshmand, and G. Scutari, "Convergence rate of distributed optimization algorithms based on gradient tracking," *arXiv preprint arXiv:1905.02637*, 2019.