Efficient Methods for Distributed Machine Learning and Resource Management in the Internet-of-Things

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

Tianyi Chen

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

Advisor: Georgios B. Giannakis

June 2019

Acknowledgments

There are so many people to whom I wish to thank for making my past years at the University of Minnesota (UMN) the most enjoyable journey of my life.

My deepest gratitude goes to my advisor Prof. Georgios B. Giannakis. His advice on promising research directions, and feedback on research presentations has been extraordinary to me by all means. His invaluable guidance as well as constant encouragement through dedicating extensive amounts of time has not only made me a better researcher, but also a better person. His vision and enthusiasm about innovative research and beyond, his broad and deep knowledge, and his unbounded energy have constantly been a true inspiration for me. More importantly, he has also offered his friendship to help me build up research collaborations, which I am truly grateful.

Special thanks also go to my thesis committee members Profs. Arindam Banerjee, Mingyi Hong, Na Li, and Zhi-Li Zhang, for their patience on coordinating the date of final defense, as well as for their time of attending the defense. Their insightful comments, valuable criticism, and constant encouragement considerably improved the quality of my research. I am doubly grateful to Prof. Na Li for traveling from Boston to attend my Ph. D. thesis defense.

During my PhD studies, I had the opportunity to collaborate with several excellent individuals, and I have greatly benefited from their critical thinking, brilliant ideas, and vision. Particularly, I would like to express my gratitude to Prof. Xin Wang who was patient enough to collaborate with me during my first year at UMN. I would also like to extend due credit and warmest thanks to Prof. Sergio Barbarossa, Prof. Tamer Başar, Prof. Longbo Huang, Prof. Na Li, Prof. Qing Ling, Dr. Aryan Mokhtari, Dr. Tao Sun, Prof. Wotao Yin, Kaiqing Zhang, and Prof. Zhi-Li Zhang for their insightful input to our fruitful collaborations. The material in this thesis has also benefited from discussions with current and former members of the SPiNCOM group at UMN: Dimitris Berberidis, Dr. Jia Chen, Vassilis Ioannidis, Georgios V. Karanikolas, Donghoon Lee, Prof. Geert Leus, Bingcong Li, Meng Ma, Dr. Morteza Mardani, Prof. Antonio G. Marqués, Prof. Gonzalo Mateos, Dr. Athanasios Nikolakopoulos, Prof. Alejandro Ribeiro, Prof. Daniel Romero, Alireza

Sadeghi, Fatemeh Sheikholeslami, Dr. Yanning Shen, Panagiotis Traganitis, Dr. Liang Zhang, and Prof. Yu Zhang. I am truly grateful to these people for their continuous help.

Last but not least, I would also like to thank my family for all their unconditional support along the way. Particularly, I thank my mother, father, and grandmother for educating me to be energetic, determinant, ambitious, and persistent in pursuing those dreams deeply rooted in my soul. As a closing comment, I want to remind myself that: I have a dream; always believe in the power of having a dream; always trust the power of consistently pursuing one's dream; and never doubt that something wonderful is about to happen.

Tianyi Chen
Minneapolis, January 2019

Dedication

This dissertation is dedicated to my family for their unconditional love and support.

Abstract

Undoubtedly, this century evolves in a world of interconnected entities, where the notion of Internet-of-Things (IoT) plays a central role in the proliferation of linked devices and objects. In this context, the present dissertation deals with large-scale networked systems including IoT that consist of heterogeneous components, and can operate in unknown environments. The focus is on the theoretical and algorithmic issues at the intersection of optimization, machine learning, and networked systems. Specifically, the research objectives and innovative claims include:
(T1) Scalable distributed machine learning approaches for efficient IoT implementation; and,
(T2) Enhanced resource management policies for IoT by leveraging machine learning advances.

Conventional machine learning approaches require centralizing the users' data on one machine or in a data center. Considering the massive amount of IoT devices, centralized learning becomes computationally intractable, and rises serious privacy concerns. The widespread consensus today is that besides data centers at the cloud, future machine learning tasks have to be performed starting from the network edge, namely mobile devices. The first contribution offers innovative distributed learning methods tailored for heterogeneous IoT setups, and with reduced communication overhead. The resultant distributed algorithm can afford provably reduced communication complexity in distributed machine learning. From learning to control, reinforcement learning will play a critical role in many complex IoT tasks such as autonomous vehicles. In this context, the thesis introduces a distributed reinforcement learning approach featured with its high communication efficiency.

Optimally allocating computing and communication resources is a crucial task in IoT. The second novelty pertains to learning-aided optimization tools tailored for resource management tasks. To date, most resource management schemes are based on a pure optimization viewpoint (e.g., the dual (sub)gradient method), which incurs suboptimal performance. From the vantage point of IoT, the idea is to leverage the abundant historical data collected by devices, and formulate the resource management problem as an empirical risk minimization task — a central topic in machine learning research. By cross-fertilizing advances of optimization and learning theory, a learn-and-adapt resource management framework is developed. An upshot of the second part is its ability to account for the feedback-limited nature of tasks in IoT. Typically, solving resource allocation problems necessitates knowledge of the models that map a resource variable to its cost or utility. Targeting scenarios where models are not available, a model-free learning scheme is developed in this thesis, along with its bandit version. These algorithms come with provable performance guarantees, even when knowledge about the underlying systems is obtained only through repeated interactions with the environment.

The overarching objective of this dissertation is to wed state-of-the-art optimization and machine learning tools with the emerging IoT paradigm, in a way that they can inspire and reinforce the development of each other, with the ultimate goal of benefiting daily life.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation and context

The past decade has witnessed a proliferation of connected devices and objects, where the notion of Internet-of-Things (IoT) plays a central role in the envisioned technological advances. Conceptually speaking, IoT foresees an intelligent network infrastructure with ubiquitous smart devices - home automation, interactive healthcare, and self-driving connected vehicles, are typical in IoT [7, 177]. Today, a number of IoT applications have already brought major benefits to many aspects of our daily life. The current generation of IoT can already afford an increasing amount of real-time automation, and thus intelligence toward the vision of real-time IoT. However, despite the popularity of IoT, several critical challenges must be addressed before embracing its full potential [151, 4]. To this end, we highlight three key challenges that are arguably expected to be at the epicenter of emerging IoT research fields.

Extreme heterogeneity. The computational and communication capacities of connected devices differ due to differences in hardware (e.g., CPU frequency), communication protocol (e.g., ZigBee, WiFi), and energy availability (e.g., battery level) [176]. The tasks carried out on various devices are often considerably diverse, e.g., motion sensors monitor human behavior in a smart home [102], while cameras are responsible for recognizing a suspicious behavior in a crowded environment, or, vehicle plates in a parking garage.

Unpredictable dynamics. Unlike many existing communication, computing and networking platforms, the IoT dynamics can stem from multiple sources, where *adaptivity* is not only critical but also essential in designing hardware and management protocols. Such sources entail human-in-the-loop dynamics in addition to physical objects [102], demand response in energy systems [56], and intelligent automotive operations [101]. In these applications, IoT dynamics are intertwined with or even partially determined by human behavior [113, 121, 47] - as such, high degree of adaptivity in the algorithm and hardware design is needed.

Scalability at the core. IoT entails an intelligent network infrastructure with a massive number of devices. It is estimated that by 2020, there will be more than 50 billion devices connected through the Internet [54], which highlights *scalability* as a key challenge for IoT [151, 7]. Scalability is not only about computational efficiency, but also about lower communication overhead (e.g., how often a device needs to communicate with the remote cloud center), as well as reduced information needed (e.g., what type of information a device needs before making sensible decisions).

Faced with these major IoT challenges, innovations in theoretical foundations and algorithmic designs for machine learning and resource management tasks in IoT are desired to enable efficient large-scale operations, and seamless co-existence of humans with things [38]. Consequently, it is imperative to develop new tools for learning and management that tap into diverse inference, signal processing, communications, and networking techniques, by drawing from fields such as machine learning, and optimization. The novel expertise gleaned from these research areas, coupled with solid analytical approaches, are the best credentials for succeeding in IoT research [151].

From a network architecture perspective, to ensure the desired user experience and meet heterogeneous service requirements, IoT tasks nowadays are no longer solely supported by the cloud data centers, but also through a promising new architecture termed *edge computing*. This architecture distributes computation, communication, and storage closer to the end IoT devices and users, along the cloud-to-things continuum [38, 9, 10, 105, 167, 103]. In this way, delay-sensitive applications launched by a mobile device can be offloaded to the nearest mobile edge host, and the most popular contents can also be cached to minimize downloading time [40, 39].

From the algorithmic design perspective, a large volume of data is being generated from diverse IoT systems such as transportation, electric power, and computer networks, as well as the future urban infrastructure with ubiquitous smart devices. At the same time, the proliferation of optimization and machine learning advances motivates a systematic and efficient way to uncover “hidden insights” through learning from historical relationships and trends in massive datasets [164]. Learning from dynamic and large volumes of IoT data is expected to bring major science and engineering advances along with consequent improvements in quality of human life [110, 30].

1.2 Research overview

In this context, this dissertation is at the intersection of IoT, optimization, machine learning, and networking. The research presented in this dissertation focuses on building fundamental connections between methodologies from the optimization, machine learning and networking communities, and developing inter-disciplinary approaches for IoT.

It contributes answers to the following two intertwined questions.

(Q1) *How can we scale up machine learning approaches for efficient IoT implementation?*

(Q2) *How learning advances can be leveraged to enhance resource management for IoT?*

The overarching objective is to wed state-of-the-art optimization and machine learning tools with the emerging IoT paradigm, in a way that they can inspire and reinforce the development of each other, with the ultimate goal of benefiting our daily life.

1.2.1 Scale up machine learning approaches for IoT

It is estimated that by 2020, there will be more than 50 billion devices connected through the Internet. To tackle **(Q1)**, it is evident that *scalability* and *heterogeneity* are two key challenges for IoT [30]. Scalability is not only about computational efficiency, but also about communication overhead of running learning algorithms at the network edge; while heterogeneity comes from both the wide range of hardware devices, as well as the diversity of tasks offered by each device. The first part of the dissertation, consisting of Chapters 2 and 3, will primarily tackle these issues.

• **Federated learning at the network edge.** Conventional machine learning approaches require centralizing the users' data on one machine or in a data center. Considering the massive amount of IoT devices, centralized learning becomes computationally intractable, and rises serious privacy concerns. To date, the widespread consensus is that besides data centers at the cloud, future machine learning tasks have to be performed starting from the network edge, namely mobile devices. This is the overarching goal of *edge computing*, also known as *federated learning* [109]. Towards this goal, this research is centered on reducing the communication overhead during the federated learning processes [31], and enhancing the robustness of learning under adversarial attacks [89]. Our learning method with adaptive communication mechanism [31] has been selected as the spotlight presentation in NeurIPS, which establishes a provably reduced communication complexity in federated learning. This part of research will be presented in **Chapter 2**. Challenges of distributed learning also lie in asynchrony and delay introduced by e.g., IoT mobility and heterogeneity. In this context, we have developed algorithms for delayed online learning that can be run asynchronously on edge devices; see our recent paper [87].

• **Federated reinforcement learning over networked agents.** From learning to control, reinforcement learning (RL) will play a critical role in many complex IoT tasks. Popular RL algorithms are originally developed for the single-agent tasks, but a number of IoT tasks such as autonomous vehicles and coordination of unmanned aerial vehicles (UAV), involve multiple agents operating in a distributed fashion. Today, a group of coordinated UAVs can perform traffic control, food delivery, rescue and search tasks. To coordinate agents distributed over a network however, information exchange is necessary, which requires frequent communication among agents. For resource-limited devices (e.g., battery-powered UAVs), communication is costly and the latency caused by frequent communication becomes the bottleneck of the overall performance. In this context, we have studied the distributed RL (DRL) problem that covers multi-agent collaborative RL and parallel RL. Generalizing theory and algorithms for supervised learning, an exciting communication-efficient algorithm (LAPG) is developed for DRL [29], which builds on the policy gradient (PG) method. Remarkably, the new DRL method can achieve the same order of convergence rates as plain-vanilla policy gradient under standard conditions; and, ii) reduce the communication rounds required to achieve a targeted learning accuracy, when the distributed agents are heterogeneous. Results in this line of research have been presented as part of a tutorial we delivered at MILCOM 2018, which will be presented in **Chapter 3**.

• **Scalable function approximation with unknown dynamics.** Function approximation emerges at the core of machine learning tasks such as regression, classification, dimensionality reduction, as well as reinforcement learning. Kernel methods exhibit well-documented performance in function approximation. However, the major challenges of implementing existing methods to IoT come from two sources: i) the "curse" of dimensionality in kernel-based learning; and, ii) the need to track time-varying functions with unknown dynamics. In this context, a scalable multi-kernel learning scheme has been developed to obtain the sought nonlinear learning function 'on the fly.' To further boost performance in unknown environments, an adaptive learning scheme has been introduced, which accounts for the unknown dynamics. So far, results in this direction have appeared in [146] and [147].

1.2.2 Rethink resource management for IoT via learning

Optimally allocating limited computing and communication resources is a crucial task in IoT. The focus of the second part in this dissertation, namely Chapters 4-6, is to tackle **(Q2)** by providing affirmative answers to the following intermediate questions:

(Q2a) *can we learn from historical data to improve the existing resource management schemes*; (Q2b) *can we develop resource management schemes when the underlying models are not known?*

The key novelty here is innovative statistical and interactive learning tailored for resource management tasks in IoT.

• **Statistical learning viewpoint of resource management.** To date, most resource management schemes for IoT are based on a pure *optimization* viewpoint (e.g., the dual (sub)gradient method), which incur large queueing delays and slow convergence. From the vantage point of IoT, the fresh idea here is to leverage the abundant historical data collected by devices, and formulate the resource management problem as an empirical risk minimization (ERM) — a central topic of statistical machine learning research [164]. In this context, we have developed a fast convergent algorithm. By cross-fertilizing advances of learning theory, we have also established the sample complexity of learning a near-optimal resource management policy [26]. To boost performance in dynamic settings, we further introduced a learn-and-adapt resource management framework [32] that will be presented in **Chapter 4**, which capitalizes on the following features: (f1) it learns from historical data using advanced statistical learning tools; and, (f2) it efficiently adapts to IoT dynamics, and thus enables operational flexibility. Our proposed algorithms have been published in top signal processing and network optimization journals [32, 26], where we have analytically shown that this novel algorithmic design can provably improve the emerging performance tradeoff by an order of magnitude. To demonstrate the impact of this work, we have applied it to mobile computing and smart grid tasks [36, 88].

• **Model-free resource management for edge computing.** Typically, solving resource allocation problems necessitates knowledge of the models that map a resource allocation decision to its cost or utility; e.g., the model that maps transmit-power to the bit rate in communication systems. However, such models may not be available in IoT, because i) the utility function capturing e.g., service latency or reliability in edge computing, can be hard to model; and, ii) even if modeling is possible, IoT devices with limited resources may not afford the complexity of running sophisticated inference algorithms. Hence, another important aspects investigated in this part of the thesis is the feedback limited nature of resource allocation tasks in IoT. To account for physical constraints, we have considerably generalized the interactive learning tools for unconstrained problems to solve challenging constrained resource allocation problems [25]. Tailored for edge computing scenarios, we further developed a model-free online learning scheme [25] that will be presented in **Chapter 5**, along with its bandit version [34] that will be presented in **Chapter 6**. These algorithms come with provable performance guarantees, even when knowledge about the underlying system models can be obtained only through repeated interactions with the environment.

The dissertation is summarized, and interesting open problems are included in Chapter 7.

1.3 Notational conventions

The following notation will be used throughout the subsequent chapters. Lower- (upper-) case boldface letters denote vectors (matrices). Calligraphic letters are reserved for sets, e.g., $\mathcal{S}$. Symbol $\top$ stands for matrix/vector transposition. For vectors, $\|\cdot\|_2$ or $\|\cdot\|$ represents the Euclidean norm, while $\|\cdot\|_0$ denotes the ℓ_0 pseudo-norm counting the number of nonzero entries. The floor (ceiling) operation $\lfloor c \rfloor$ ($\lceil c \rceil$) denotes the largest integer no greater (the smallest integer but no smaller) than the given number $c > 0$; $|\mathcal{S}|$ counts the number of entries in $\mathcal{S}$. Let $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ be the vector Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$.

Chapter 2

Federated learning at the network edge

2.1 Introduction

In this paper, we develop communication-efficient algorithms to solve the following problem

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^d} \mathcal{L}(\boldsymbol{\theta}) \quad \text{with} \quad \mathcal{L}(\boldsymbol{\theta}) := \sum_{m\in\mathcal{M}} \mathcal{L}_m(\boldsymbol{\theta}) \tag{2.1}$$

where $\boldsymbol{\theta} \in \mathbb{R}^d$ is the unknown vector, $\mathcal{L}$ and $\{\mathcal{L}_m, m\in\mathcal{M}\}$ are smooth (but not necessarily convex) functions with $\mathcal{M} := \{1, \ldots, M\}$. Problem (2.1) naturally arises in a number of areas, such as multi-agent optimization [115], distributed signal processing [57, 136], and distributed machine learning [44]. Considering the distributed machine learning paradigm, each $\mathcal{L}_m$ is also a sum of functions, e.g., $\mathcal{L}_m(\boldsymbol{\theta}) := \sum_{n\in\mathcal{N}_m} \ell_n(\boldsymbol{\theta})$, where ℓ_n is the loss function (e.g., square or the logistic loss) with respect to the vector $\boldsymbol{\theta}$ (describing the model) evaluated at the training sample $\mathbf{x}_n$; that is, $\ell_n(\boldsymbol{\theta}) := \ell(\boldsymbol{\theta}; \mathbf{x}_n)$. While machine learning tasks are traditionally carried out at a single server, for datasets with massive samples $\{\mathbf{x}_n\}$, running gradient-based iterative algorithms at a single server can be prohibitively slow; e.g., the server needs to sequentially compute gradient components given limited processors. A simple yet popular solution in recent years is to parallelize the training across multiple computing units (a.k.a. workers) [44]. Specifically, assuming batch samples distributedly stored in a total of M workers with the worker $m \in \mathcal{M}$ associated with samples $\{\mathbf{x}_n, n \in \mathcal{N}_m\}$, a globally shared model $\boldsymbol{\theta}$ will be updated at the central server by aggregating gradients computed by workers. Due to bandwidth and privacy concerns, each worker m will not upload its data $\{\mathbf{x}_n, n \in \mathcal{N}_m\}$ to the server, thus the learning task needs to be performed by iteratively communicating with the server.

We are particularly interested in the scenarios where communication between the central server and the local workers is costly, as is the case with the Federated Learning paradigm [109, 150], and the cloud-edge AI systems [153]. In those cases, communication latency is the bottleneck of overall performance. More precisely, the communication latency is a result of initiating communication links, queueing and propagating the message. For sending small messages, e.g., the d-dimensional model $\boldsymbol{\theta}$ or aggregated gradient, this latency dominates the message size-dependent transmission latency. Therefore, it is important to reduce the number of communication rounds, even more so than the bits per round. In short, **our goal** is to find $\boldsymbol{\theta}$ that minimizes (2.1) using as low communication overhead as possible.

2.1.1 Prior art

To put our work in context, we review prior contributions that we group in two categories.

Large-scale machine learning. Solving (2.1) at a single server has been extensively studied for large-scale learning tasks, where the "workhorse approach" is the simple yet efficient stochastic gradient descent (SGD) [131, 18, 19]. For learning beyond a single server, distributed parallel machine learning is an attractive solution to tackle large-scale learning tasks, where the parameter server architecture is the most commonly used one [44, 91]. Different from the single server case, parallel implementation of the batch gradient descent (GD) is a popular choice, since SGD that has low complexity per iteration requires a large number of iterations thus communication rounds [110]. For traditional parallel learning algorithms however, latency, bandwidth limits, and unexpected drain on resources, that delay the update of even a single worker will slow down the entire system operation. Recent research efforts in this line have been centered on understanding asynchronous-parallel algorithms to speed up machine learning by eliminating costly synchronization; e.g., [23, 156, 126, 129, 96].

Communication-efficient learning. Going beyond single-server learning, the high communication overhead becomes the bottleneck of the overall system performance [110]. Communication-efficient learning algorithms have gained popularity [73, 182]. Distributed learning approaches have been developed based on quantized (gradient) information, e.g., [157], but they only reduce the required bandwidth per communication, not the rounds. For machine learning tasks where the loss function is convex and its conjugate dual is expressible, the dual coordinate ascent-based approaches have been demonstrated to yield impressive empirical performance [150, 72, 104]. But these algorithms run in a double-loop manner, and the communication reduction has not been formally quantified. To reduce communication by accelerating convergence, approaches leveraging (inexact) second-order information have been studied in [144, 183]. Roughly speaking, algorithms in [150, 72, 104, 144, 183] reduce communication by increasing local computation (relative to GD), while our method does not increase local computation. In settings *different* from the one considered in this paper, communication-efficient approaches have been recently studied with triggered communication protocols [98, 83]. Except for convergence guarantees however, no theoretical justification for communication reduction has been established in [98]. While a sublinear convergence rate can be achieved by algorithms in [83], the proposed gradient selection rule is nonadaptive and requires double-loop iterations.

2.1.2 Our contributions

Before introducing our approach, we revisit the popular GD method for (2.1) in the setting of one parameter server and M workers: At iteration k, the server broadcasts the current model $\boldsymbol{\theta}^k$ to *all* the workers; every worker $m \in \mathcal{M}$ computes $\nabla \mathcal{L}_m(\boldsymbol{\theta}^k)$ and uploads it to the server; and once receiving gradients from all workers, the server updates the model parameters via

GD iteration
$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \alpha \nabla_{\text{GD}}^k \quad \text{with} \quad \nabla_{\text{GD}}^k := \sum_{m \in \mathcal{M}} \nabla \mathcal{L}_m(\boldsymbol{\theta}^k) \tag{2.2}$$

where α is a stepsize, and ∇_{GD}^k is an aggregated gradient that summarizes the model change. To implement (2.2), the server has to communicate with *all* workers to obtain fresh $\{\nabla \mathcal{L}_m(\boldsymbol{\theta}^k)\}$.

In this context, the present paper puts forward a new batch gradient method (as simple as GD) that can *skip* communication at certain rounds, which justifies the term **L**azily **A**ggregated **G**radient (**LAG**). With its derivations deferred to Section 2.2, LAG resembles (2.2), given by

LAG iteration
$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \alpha \nabla^k \quad \text{with} \quad \nabla^k := \sum_{m \in \mathcal{M}} \nabla \mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^k) \tag{2.3}$$

where each $\nabla \mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^k)$ is either $\nabla \mathcal{L}_m(\boldsymbol{\theta}^k)$, when $\hat{\boldsymbol{\theta}}_m^k = \boldsymbol{\theta}^k$, or an outdated gradient that has been computed using an old copy $\hat{\boldsymbol{\theta}}_m^k \neq \boldsymbol{\theta}^k$. Instead of requesting fresh gradient from every worker in (2.2), the twist is to obtain ∇^k by refining the previous aggregated gradient ∇^{k-1}; that is, using only the new gradients from the *selected* workers in $\mathcal{M}^k$, while reusing the outdated gradients from the rest of workers. Therefore, with $\hat{\boldsymbol{\theta}}_m^k := \boldsymbol{\theta}^k, \forall m \in \mathcal{M}^k, \ \hat{\boldsymbol{\theta}}_m^k := \hat{\boldsymbol{\theta}}_m^{k-1}, \forall m \notin \mathcal{M}^k$, LAG in

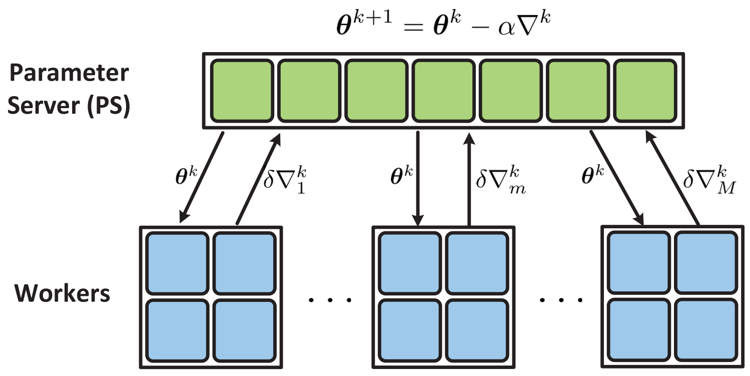

Figure 2.1: LAG for distributed machine learning in a parameter server setup.

(2.3) is equivalent to

LAG iteration
$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \alpha\nabla^k \quad \text{with} \quad \nabla^k = \nabla^{k-1} + \sum_{m\in\mathcal{M}^k} \delta\nabla_m^k \tag{2.4}$$

where $\delta\nabla_m^k := \nabla\mathcal{L}_m(\boldsymbol{\theta}^k) - \nabla\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^{k-1})$ is the difference between two evaluations of $\nabla\mathcal{L}_m$ at the current iterate $\boldsymbol{\theta}^k$ and the old copy $\hat{\boldsymbol{\theta}}_m^{k-1}$. If ∇^{k-1} is stored in the server, this simple modification scales down the number of communication rounds from GD's M to LAG's $|\mathcal{M}^k|$.

We develop two different rules to select $\mathcal{M}^k$. The first rule is adopted by the parameter server (PS), and the second one by every worker (WK). At iteration k,

LAG-PS: the server determines $\mathcal{M}^k$ and sends $\boldsymbol{\theta}^k$ to the workers in $\mathcal{M}^k$; each worker $m \in \mathcal{M}^k$ computes $\nabla\mathcal{L}_m(\boldsymbol{\theta}^k)$ and uploads $\delta\nabla_m^k$; workers in $\mathcal{M}^k$ do nothing; the server updates via (2.4);

LAG-WK: the server broadcasts $\boldsymbol{\theta}^k$ to all workers; every worker computes $\nabla\mathcal{L}_m(\boldsymbol{\theta}^k)$, and checks if it belongs to $\mathcal{M}^k$; only the workers in $\mathcal{M}^k$ upload $\delta\nabla_m^k$; the server updates via (2.4).

See a comparison of two LAG variants with GD in Table 2.1.

Naively reusing outdated gradients, while saving communication per iteration, can increase the total number of iterations. To keep this number in control, we judiciously design our simple trigger rules so that LAG can: i) achieve the *same* order of convergence rates (thus iteration complexities) as batch GD under strongly-convex, convex, and nonconvex smooth cases; and, ii) require *reduced* communication to achieve a targeted learning accuracy, when the distributed datasets are heterogeneous (measured by certain quantity specified later). In certain learning settings, LAG requires only $\mathcal{O}(1/M)$ communication of GD. Empirically, we found that LAG can reduce the communication required by GD and other distributed parallel learning methods by

Metric	Communication		Computation		Memory	
Algorithm	**PS$\rightarrow$WK** m	**WK** m $\rightarrow$**PS**	**PS**	**WK** m	**PS**	**WK** m
GD	$\boldsymbol{\theta}^k$	$\nabla\mathcal{L}_m$	(2.2)	$\nabla\mathcal{L}_m$	$\boldsymbol{\theta}^k$	/
LAG-PS	$\boldsymbol{\theta}^k$, if $m\in\mathcal{M}^k$	$\delta\nabla_m^k$, if $m\in\mathcal{M}^k$	(2.4), (2.15b)	$\nabla\mathcal{L}_m$, if $m\in\mathcal{M}^k$	$\boldsymbol{\theta}^k$, ∇^k, $\{\hat{\boldsymbol{\theta}}_m^k\}$	$\nabla\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^k)$
LAG-WK	$\boldsymbol{\theta}^k$	$\delta\nabla_m^k$, if $m\in\mathcal{M}^k$	(2.4)	$\nabla\mathcal{L}_m$, (2.15a)	$\boldsymbol{\theta}^k, \nabla^k$	$\nabla\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^k)$

Table 2.1: A comparison of communication, computation and memory requirements. **PS** denotes the parameter server, **WK** denotes the worker, **PS$\rightarrow$WK** m is the communication link from the server to worker m, and **WK** $m \rightarrow$ **PS** is the communication link from worker m to the server.

several orders of magnitude.

Notation. Bold lowercase letters denote column vectors, which are transposed by $(\cdot)^\top$. And $\|\mathbf{x}\|$ denotes the ℓ_2-norm of $\mathbf{x}$. Inequalities for vectors $\mathbf{x} > \mathbf{0}$ is defined entrywise.

2.2 LAG: Lazily aggregated gradient approach

In this section, we formally develop our LAG method, and present the intuition and basic principles behind its design. The original idea of LAG comes from a simple rewriting of the GD iteration (2.2) as

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \alpha \sum_{m\in\mathcal{M}} \nabla\mathcal{L}_m(\boldsymbol{\theta}^{k-1}) - \alpha \sum_{m\in\mathcal{M}} \Big(\nabla\mathcal{L}_m\big(\boldsymbol{\theta}^k\big) - \nabla\mathcal{L}_m\big(\boldsymbol{\theta}^{k-1}\big)\Big). \tag{2.5}$$

Let us view $\nabla\mathcal{L}_m(\boldsymbol{\theta}^k)-\nabla\mathcal{L}_m(\boldsymbol{\theta}^{k-1})$ as a refinement to $\nabla\mathcal{L}_m(\boldsymbol{\theta}^{k-1})$, and recall that obtaining this refinement requires a round of communication between the server and the worker m. Therefore, to save communication, we can skip the server's communication with the worker m if this refinement is small compared to the old gradient; that is, $\|\nabla\mathcal{L}_m(\boldsymbol{\theta}^k) - \nabla\mathcal{L}_m(\boldsymbol{\theta}^{k-1})\| \ll \|\sum_{m\in\mathcal{M}} \nabla\mathcal{L}_m(\boldsymbol{\theta}^{k-1})\|$.

Generalizing on this intuition, given the generic outdated gradient components $\{\nabla\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^{k-1})\}$ with $\hat{\boldsymbol{\theta}}_m^{k-1}{=}\boldsymbol{\theta}_m^{k-1-\tau_m^{k-1}}$for a certain $\tau_m^{k-1}\geq 0$, if communicating with some workers will bring only small gradient refinements, we skip those communications (contained in set $\mathcal{M}_c^k$) and end up with

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \alpha \sum_{m\in\mathcal{M}} \nabla\mathcal{L}_m\big(\hat{\boldsymbol{\theta}}_m^{k-1}\big) - \alpha \sum_{m\in\mathcal{M}^k} \Big(\nabla\mathcal{L}_m\big(\boldsymbol{\theta}^k\big) - \nabla\mathcal{L}_m\big(\hat{\boldsymbol{\theta}}_m^{k-1}\big)\Big) \tag{2.6a}$$

$$= \boldsymbol{\theta}^k - \alpha\nabla\mathcal{L}(\boldsymbol{\theta}^k) - \alpha \sum_{m\in\mathcal{M}_c^k} \Big(\nabla\mathcal{L}_m\big(\hat{\boldsymbol{\theta}}_m^{k-1}\big) - \nabla\mathcal{L}_m\big(\boldsymbol{\theta}^k\big)\Big) \tag{2.6b}$$

where $\mathcal{M}^k$ and $\mathcal{M}_c^k$ are the sets of workers that *do* and *do not* communicate with the server, respectively. It is easy to verify that (2.6) is identical to (2.3) and (2.4). Comparing (2.2) with (2.6b), when $\mathcal{M}_c^k$ includes more workers, more communication is saved, but $\boldsymbol{\theta}^k$ is updated by a coarser gradient.

Key to addressing this communication vs accuracy tradeoff is a principled criterion to select a subset of workers $\mathcal{M}_c^k$ that do not communicate with the server at each round. To achieve this "sweet spot," we will rely on the fundamental descent lemma. For GD, it is given as follows [119].

Lemma 1 (GD descent in objective)**.** *Suppose $\mathcal{L}(\boldsymbol{\theta})$ is L-smooth, and $\bar{\boldsymbol{\theta}}^{k+1}$ is generated by running one-step GD iteration* (2.2) *given $\boldsymbol{\theta}^k$ and stepsize α. Then the objective values satisfy*

$$\mathcal{L}(\bar{\boldsymbol{\theta}}^{k+1}) - \mathcal{L}(\boldsymbol{\theta}^k) \leq -\left(\alpha - \frac{\alpha^2 L}{2}\right)\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\|^2 := \Delta_{\mathrm{GD}}^k(\boldsymbol{\theta}^k). \tag{2.7}$$

Likewise, for our wanted iteration (2.6), the following holds; its proof is given in the Supplement.

Lemma 2 (LAG descent in objective)**.** *Suppose $\mathcal{L}(\boldsymbol{\theta})$ is L-smooth, and $\boldsymbol{\theta}^{k+1}$ is generated by running one-step LAG iteration* (2.4) *given $\boldsymbol{\theta}^k$. The objective values satisfy (cf. $\delta\nabla_m^k$ in* (2.4)*)*

$$\mathcal{L}(\boldsymbol{\theta}^{k+1}) - \mathcal{L}(\boldsymbol{\theta}^k) \leq -\frac{\alpha}{2}\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + \frac{\alpha}{2}\left\|\sum_{m\in\mathcal{M}_c^k}\delta\nabla_m^k\right\|^2 + \left(\frac{L}{2} - \frac{1}{2\alpha}\right)\left\|\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k\right\|^2 := \Delta_{\mathrm{LAG}}^k(\boldsymbol{\theta}^k). \tag{2.8}$$

Lemmas 1 and 2 estimate the objective value descent by performing one-iteration of the GD and LAG methods, respectively, conditioned on a common iterate $\boldsymbol{\theta}^k$. GD finds $\Delta_{\mathrm{GD}}^k(\boldsymbol{\theta}^k)$ by performing M rounds of communication with all the workers, while LAG yields $\Delta_{\mathrm{LAG}}^k(\boldsymbol{\theta}^k)$ by performing only $|\mathcal{M}^k|$ rounds of communication with a selected subset of workers. Our pursuit is to select $\mathcal{M}^k$ to ensure that *LAG enjoys larger per-communication descent than GD*; that is

$$\frac{\Delta_{\mathrm{LAG}}^k(\boldsymbol{\theta}^k)}{|\mathcal{M}^k|} \leq \frac{\Delta_{\mathrm{GD}}^k(\boldsymbol{\theta}^k)}{M}. \tag{2.9}$$

If we choose the standard $\alpha = 1/L$ in Lemmas 1 and 2, it follows that

$$\boxed{\begin{aligned} &\Delta_{\mathrm{GD}}^k(\boldsymbol{\theta}^k) := -\frac{1}{2L}\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 && \text{(2.10a)}\\ &\Delta_{\mathrm{LAG}}^k(\boldsymbol{\theta}^k) := -\frac{1}{2L}\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + \frac{1}{2L}\left\|\sum_{m\in\mathcal{M}_c^k}\left(\nabla\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^{k-1}) - \nabla\mathcal{L}_m(\boldsymbol{\theta}^k)\right)\right\|^2. && \text{(2.10b)}\end{aligned}}$$

Plugging (2.10) into (2.9), and rearranging terms, (2.9) is equivalent to

$$\left\|\sum_{m\in\mathcal{M}_c^k}\left(\nabla\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^{k-1})-\nabla\mathcal{L}_m(\boldsymbol{\theta}^k)\right)\right\|^2 \leq \left|\mathcal{M}_c^k\right|\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2\Big/M. \tag{2.11}$$

Note that since we have

$$\left\|\sum_{m\in\mathcal{M}_c^k}\left(\nabla\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^{k-1})-\nabla\mathcal{L}_m(\boldsymbol{\theta}^k)\right)\right\|^2 \leq \left|\mathcal{M}_c^k\right|\sum_{m\in\mathcal{M}_c^k}\left\|\nabla\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^{k-1})-\nabla\mathcal{L}_m(\boldsymbol{\theta}^k)\right\|^2 \tag{2.12}$$

if we can further show that

$$\left\|\nabla\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^{k-1})-\nabla\mathcal{L}_m(\boldsymbol{\theta}^k)\right\|^2 \leq \left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2\Big/M^2, \quad \forall m\in\mathcal{M}_c^k. \tag{2.13}$$

then we can prove that (2.11) holds thus (2.9) also holds.

However, directly checking (2.13) at each worker is expensive since i) obtaining $\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\|^2$ requires information from all the workers; and ii) each worker does not know $\mathcal{M}_c^k$. Instead, we approximate $\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\|^2$ in (2.13) by

$$\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 \approx \frac{1}{\alpha^2}\sum_{d=1}^{D}\xi_d\left\|\boldsymbol{\theta}^{k+1-d}-\boldsymbol{\theta}^{k-d}\right\|^2 \tag{2.14}$$

where $\{\xi_d\}_{d=1}^D$ are constant weights. The rationale here is that, as $\mathcal{L}$ is smooth, $\nabla\mathcal{L}(\boldsymbol{\theta}^k)$ cannot be very different from the recent gradients or the recent iterate *lags*.

Building upon (2.13) and (2.14), we will include worker m in $\mathcal{M}_c^k$ of (2.6) if it satisfies

LAG-WK condition $$\left\|\nabla\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^{k-1})-\nabla\mathcal{L}_m(\boldsymbol{\theta}^k)\right\|^2 \leq \frac{1}{\alpha^2M^2}\sum_{d=1}^{D}\xi_d\left\|\boldsymbol{\theta}^{k+1-d}-\boldsymbol{\theta}^{k-d}\right\|^2. \tag{2.15a}$$

Condition (2.15a) is checked at *the worker side* after each worker receives $\boldsymbol{\theta}^k$ from the server and computes its $\nabla\mathcal{L}_m(\boldsymbol{\theta}^k)$. If broadcasting is also costly, we can resort to the following *server side* rule:

LAG-PS condition $$L_m^2\left\|\hat{\boldsymbol{\theta}}_m^{k-1}-\boldsymbol{\theta}^k\right\|^2 \leq \frac{1}{\alpha^2M^2}\sum_{d=1}^{D}\xi_d\left\|\boldsymbol{\theta}^{k+1-d}-\boldsymbol{\theta}^{k-d}\right\|^2. \tag{2.15b}$$

Algorithm 1 LAG-WK

```
1:  **Input:** Stepsize $\alpha > 0$, and $\{\xi_d\}$.
2:  **Initialize:** $\boldsymbol{\theta}^1$, $\{\nabla\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^0), \forall m\}$.
3:  **for** $k = 1, 2, \dots, K$ **do**
4:      Server **broadcasts** $\boldsymbol{\theta}^k$ to all workers.
5:      **for** worker $m = 1, \dots, M$ **do**
6:          Worker $m$ **computes** $\nabla\mathcal{L}_m(\boldsymbol{\theta}^k)$.
7:          Worker $m$ **checks** condition (2.15a).
8:          **if** worker $m$ violates (2.15a) **then**
9:              Worker $m$ **uploads** $\delta\nabla_m^k$.
10:                 ▷ Save $\nabla\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^k) = \nabla\mathcal{L}_m(\boldsymbol{\theta}^k)$
11:         **else**
12:             Worker $m$ uploads nothing.
13:         **end if**
14:     **end for**
15:     Server **updates** via (2.4).
16: **end for**
```

Algorithm 2 LAG-PS

```
1:  **Input:** Stepsize $\alpha > 0$, $\{\xi_d\}$, and $L_m$, $\forall m$.
2:  **Initialize:** $\boldsymbol{\theta}^1$, $\{\hat{\boldsymbol{\theta}}_m^0, \nabla\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^0), \forall m\}$.
3:  **for** $k = 1, 2, \dots, K$ **do**
4:      **for** worker $m = 1, \dots, M$ **do**
5:          Server **checks** condition (2.15b).
6:          **if** worker $m$ violates (2.15b) **then**
7:              Server **sends** $\boldsymbol{\theta}^k$ to worker $m$.
8:                  ▷ Save $\hat{\boldsymbol{\theta}}_m^k = \boldsymbol{\theta}^k$ at server
9:              Worker $m$ **computes** $\nabla\mathcal{L}_m(\boldsymbol{\theta}^k)$.
10:             Worker $m$ **uploads** $\delta\nabla_m^k$.
11:         **else**
12:             No actions at server and worker $m$.
13:         **end if**
14:     **end for**
15:     Server **updates** via (2.4).
16: **end for**
```

Table 2.2: A comparison of LAG-WK and LAG-PS.

The values of $\{\xi_d\}$ and D admit simple choices, e.g., $\xi_d = 1/D$, $\forall d$ with $D = 10$ used in the simulations.

LAG-WK vs LAG-PS. To perform (2.15a), the server needs to broadcast the current model $\boldsymbol{\theta}^k$, and all the workers need to compute the gradient; while performing (2.15b), the server needs the estimated smoothness constant L_m for all the local functions. On the other hand, as it will be shown in Section 2.3, (2.15a) and (2.15b) lead to the same worst-case convergence guarantees. In practice, however, the server-side condition is more conservative than the worker-side one at communication reduction, because the smoothness of $\mathcal{L}_m$ readily implies that satisfying (2.15b) will necessarily satisfy (2.15a), but not vice versa. Empirically, (2.15a) will lead to a larger $\mathcal{M}_c^k$ than that of (2.15b), and thus extra communication overhead will be saved. Hence, (2.15a) and (2.15b) can be chosen according to users' preferences. LAG-WK and LAG-PS are summarized as Algorithms 1 and 2.

Regarding our proposed LAG method, two remarks are in order.

R1) With recursive update of the lagged gradients in (2.4) and the lagged iterates in (2.15), implementing LAG is as simple as GD; see Table 2.1. Both empirically and theoretically, we will further demonstrate that using lagged gradients even reduces the overall delay by cutting down costly communication.

R2) Compared with existing efforts for communication-efficient learning such as quantized gradient, Nesterov's acceleration, dual coordinate ascent and second-order methods, LAG is not orthogonal to all of them. Instead, LAG can be combined with these methods to develop even more powerful learning schemes. Extension to the proximal LAG is also possible to cover nonsmooth regularizers.

2.3 Iteration and communication complexity

In this section, we establish the convergence of LAG, under the following standard conditions.

Assumption 1: Loss function $\mathcal{L}_m(\boldsymbol{\theta})$ *is* L_m*-smooth, and* $\mathcal{L}(\boldsymbol{\theta})$ *is* L*-smooth.*

Assumption 2: $\mathcal{L}(\boldsymbol{\theta})$ *is convex and coercive.*

Assumption 3: $\mathcal{L}(\boldsymbol{\theta})$ *is* μ*-strongly convex, or generally, satisfies the Polyak-Łojasiewicz (PL) condition with the constant* μ*; that is,* $2\mu(\mathcal{L}(\boldsymbol{\theta}^k) - \mathcal{L}(\boldsymbol{\theta}^*)) \leq \|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\|^2$.

Note that the PL condition in Assumption 3 is strictly weaker than the strongly convexity (or even convexity), and it is satisfied by a wider range of machine learning problems such as least squares for underdetermined linear systems and logistic regression; see details in [77]. While the PL condition is sufficient for the subsequent linear convergence analysis, we will still use the strong convexity for the ease of understanding by a wide audience.

The subsequent analysis critically builds on the following **Lyapunov function**:

$$\mathbb{V}^k := \mathcal{L}(\boldsymbol{\theta}^k) - \mathcal{L}(\boldsymbol{\theta}^*) + \sum_{d=1}^{D} \beta_d \left\|\boldsymbol{\theta}^{k+1-d} - \boldsymbol{\theta}^{k-d}\right\|^2 \tag{2.16}$$

where $\boldsymbol{\theta}^*$ is the minimizer of (2.1), and $\{\beta_d\}$ are constants that will be determined later.

We will start with the sufficient descent of our $\mathbb{V}^k$ in (2.16).

Lemma 3 (descent lemma)**.** *Under Assumption 1, if* α *and* $\{\xi_d\}$ *are chosen properly, there exist constants* $c_0, \cdots, c_D \geq 0$ *such that the Lyapunov function in* (2.16) *satisfies*

$$\mathbb{V}^{k+1} - \mathbb{V}^k \leq -c_0 \left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 - \sum_{d=1}^{D} c_d \left\|\boldsymbol{\theta}^{k+1-d} - \boldsymbol{\theta}^{k-d}\right\|^2 \tag{2.17}$$

which implies the descent in our Lyapunov function, that is, $\mathbb{V}^{k+1} \leq \mathbb{V}^k$.

Lemma 3 is a generalization of GD's descent lemma. As specified in the supplementary material, under properly chosen $\{\xi_d\}$, the stepsize $\alpha \in (0, 2/L)$ including $\alpha = 1/L$ guarantees

(2.17), matching the stepsize region of GD. With $\mathcal{M}^k = \mathcal{M}$ and $\beta_d = 0$, $\forall d$ in (2.16), Lemma 3 reduces to Lemma 1.

2.3.1 Convergence in strongly convex case

We first present the convergence under the smooth and strongly convex condition.

Theorem 1 (strongly convex case)**.** *Under Assumptions 1 and 3, the iterates* $\{\boldsymbol{\theta}^k\}$ *generated by LAG-WK or LAG-PS satisfy*

$$\mathcal{L}(\boldsymbol{\theta}^K) - \mathcal{L}(\boldsymbol{\theta}^*) \leq \left(1 - c(\alpha; \{\xi_d\})\right)^K \mathbb{V}^0 \tag{2.18}$$

where $\boldsymbol{\theta}^*$ *is the minimizer of* $\mathcal{L}(\boldsymbol{\theta})$ *in* (2.1)*, and* $c(\alpha; \{\xi_d\}) \in (0, 1)$ *is a constant depending on* α, $\{\xi_d\}$ *and* $\{\beta_d\}$ *as well as the condition number* $\kappa := L/\mu$ *that are specified in the supplementary material.*

Iteration complexity. The iteration complexity in its generic form is complicated since $c(\alpha; \{\xi_d\})$ depends on the choice of several parameters. Specifically, if we choose the parameters as follows

$$\xi_1 = \cdots = \xi_D := \xi < \frac{1}{D}, \quad \alpha := \frac{1 - \sqrt{D\xi}}{L}, \quad \beta_1 = \cdots = \beta_D := \frac{D - d + 1}{2\alpha\sqrt{D/\xi}} \tag{2.19}$$

then, following Theorem 1, the iteration complexity of LAG in this case is

$$\mathbb{I}_{\text{LAG}}(\epsilon) = \frac{\kappa}{1 - \sqrt{D\xi}} \log\left(\epsilon^{-1}\right). \tag{2.20}$$

The iteration complexity in (2.20) is on the same order of GD's iteration complexity $\kappa \log(\epsilon^{-1})$, but has a worse constant. This is the consequence of using a smaller stepsize in (2.19) (relative to $\alpha = 1/L$ in GD) to simplify the choice of other parameters. Empirically, LAG with $\alpha = 1/L$ can achieve almost the same empirical iteration complexity as GD; see Section 2.4. Building on the iteration complexity, we study next the *communication complexity* of LAG. In the setting of our interest, we define the communication complexity as the *total number of uploads* over all the workers needed to achieve accuracy ϵ. While the accuracy refers to the objective optimality error in the strongly convex case, it is considered as the gradient norm in general (non)convex cases.

The power of LAG is best illustrated by numerical examples; see an example of LAG-WK in Figure 2.2. Clearly, workers with a small smoothness constant communicate with the server less frequently. This intuition will be formally treated in the next lemma.

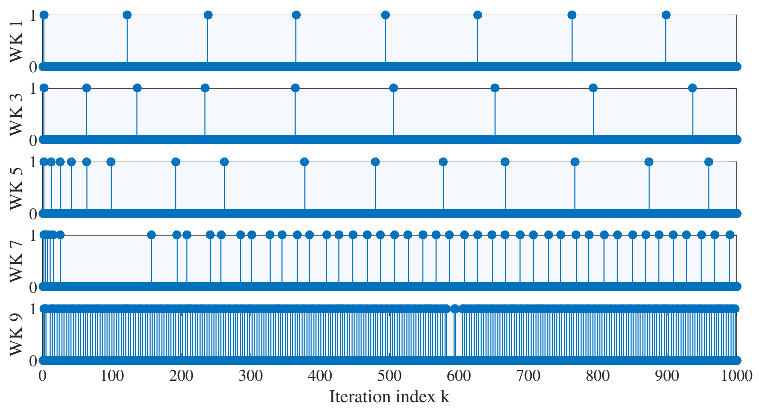

Figure 2.2: Communication events of workers $1, 3, 5, 7, 9$ over $1,000$ iterations. Each stick is an upload. An example with $L_1 < \ldots < L_9$.

Lemma 4 (lazy communication)**.** *Define the importance factor of every worker m as $\mathbb{H}(m) := L_m/L$. If the stepsize α and the constants $\{\xi_d\}$ in* (2.15) *satisfy $\xi_D \leq \cdots \leq \xi_d \leq \cdots \leq \xi_1$ and worker m satisfies*

$$\mathbb{H}^2(m) \leq \xi_d/(d\alpha^2 L^2 M^2) := \gamma_d \tag{2.21}$$

then, until iteration k, worker m communicates with the server at most $k/(d+1)$ rounds.

Lemma 4 asserts that if the worker m has a small L_m (a close-to-linear loss function) such that $\mathbb{H}^2(m) \leq \gamma_d$, then under LAG, it only communicates with the server at most $k/(d+1)$ rounds. This is in contrast to the total of k communication rounds involved per worker under GD. Ideally, we want as many workers satisfying (2.21) as possible, especially when d is large.

To quantify the overall communication reduction, we will rely on what we term the **heterogeneity score function**, given by

$$h(\gamma) := \frac{1}{M} \sum_{m \in \mathcal{M}} \mathbb{1}(\mathbb{H}^2(m) \leq \gamma) \tag{2.22}$$

where the indicator $\mathbb{1}$ equals 1 when $\mathbb{H}^2(m) \leq \gamma$ holds, and 0 otherwise. Clearly, $h(\gamma)$ is a nondecreasing function of γ, that depends on the distribution of smoothness constants $L_1, L_2, \ldots, L_M$. It is also instructive to view it as the cumulative distribution function of the *deterministic* quantity $\mathbb{H}^2(m)$, implying $h(\gamma) \in [0, 1]$. Putting it in our context, the critical quantity $h(\gamma_d)$ lower bounds the fraction of workers that communicate with the server at most $k/(d+1)$ rounds until the k-th iteration.

We are now ready to present the communication complexity.

Proposition 1 (communication complexity)**.** *Under the same conditions as those in Theorem 1, with γ_d defined in* (2.21) *and the function $h(\gamma)$ defined in* (2.22)*, the communication complexity of LAG denoted as $\mathbb{C}_{\text{LAG}}(\epsilon)$ is bounded by*

$$\mathbb{C}_{\text{LAG}}(\epsilon) \leq \left(1 - \sum_{d=1}^{D}\left(\frac{1}{d} - \frac{1}{d+1}\right) h\left(\gamma_d\right)\right) M\, \mathbb{I}_{\text{LAG}}(\epsilon) := \left(1 - \Delta\bar{\mathbb{C}}(h; \{\gamma_d\})\right) M\, \mathbb{I}_{\text{LAG}}(\epsilon) \tag{2.23}$$

where the constant is defined as $\Delta\bar{\mathbb{C}}(h; \{\gamma_d\}) := \sum_{d=1}^{D}\left(\frac{1}{d} - \frac{1}{d+1}\right) h\left(\gamma_d\right)$.

The communication complexity in (2.23) crucially depends on the iteration complexity $\mathbb{I}_{\text{LAG}}(\epsilon)$ as well as what we call the **fraction of reduced communication per iteration** $\Delta\bar{\mathbb{C}}(h; \{\gamma_d\})$. Simply choosing the parameters as (2.19), it follows from (2.20) and (2.23) that (cf. $\gamma_d = \xi(1 - \sqrt{D\xi})^{-2} M^{-2} d^{-1}$)

$$\mathbb{C}_{\text{LAG}}(\epsilon) \leq \left(1 - \Delta\bar{\mathbb{C}}(h; \xi)\right) \mathbb{C}_{\text{GD}}(\epsilon) / \left(1 - \sqrt{D\xi}\right). \tag{2.24}$$

where the GD's complexity is $\mathbb{C}_{\text{GD}}(\epsilon) = M\kappa \log(\epsilon^{-1})$. In (2.24), due to the nondecreasing property of $h(\gamma)$, increasing the constant ξ yields a smaller fraction of workers $1 - \Delta\bar{\mathbb{C}}(h; \xi)$ that are communicating per iteration, yet with a larger number of iterations (cf. (2.20)). The key enabler of LAG's communication reduction is a heterogeneous environment associated with a favorable $h(\gamma)$ ensuring that the benefit of increasing ξ is more significant than its effect on increasing iteration complexity. More precisely, for a given ξ, if $h(\gamma)$ guarantees $\Delta\bar{\mathbb{C}}(h; \xi) > \sqrt{D\xi}$, then we have $\mathbb{C}_{\text{LAG}}(\epsilon) < \mathbb{C}_{\text{GD}}(\epsilon)$. Intuitively speaking, if there is a large fraction of workers with small L_m, LAG has lower communication complexity than GD. An example follows to illustrate this reduction.

Example. Consider $L_m = 1$, $m \neq M$, and $L_M = L \geq M^2 \gg 1$, where we have $\mathbb{H}(m) = 1/L, m \neq M$, $\mathbb{H}(M) = 1$, implying that $h(\gamma) \geq 1 - \frac{1}{M}$, if $\gamma \geq 1/L^2$. Choosing $D \geq M$ and $\xi = M^2 D / L^2 < 1/D$ in (2.19) such that $\gamma_D \geq 1/L^2$ in (2.21), we have (cf. (2.24))

$$\mathbb{C}_{\text{LAG}}(\epsilon)/\mathbb{C}_{\text{GD}}(\epsilon) \leq \left[1 - \left(1 - \frac{1}{D+1}\right)\left(1 - \frac{1}{M}\right)\right] \Big/ \left(1 - MD/L\right) \approx \frac{M+D}{M(D+1)} \approx \frac{2}{M}. \tag{2.25}$$

Due to technical issues in the convergence analysis, the current condition on $h(\gamma)$ to ensure LAG's communication reduction is relatively restrictive. Establishing communication reduction on a broader learning setting that matches the LAG's intriguing empirical performance is in our research agenda.

2.3.2 Convergence in (non)convex case

LAG's convergence and communication reduction guarantees go beyond the strongly-convex case. We next establish the convergence of LAG for general convex functions.

Theorem 2 (convex case)**.** *Under Assumptions 1 and 2, if α and $\{\xi_d\}$ are chosen properly, then the iterates $\{\boldsymbol{\theta}^k\}$ generated by LAG-WK or LAG-PS satisfy*

$$\mathcal{L}(\boldsymbol{\theta}^K) - \mathcal{L}(\boldsymbol{\theta}^*) = \mathcal{O}\left(1/K\right). \tag{2.26}$$

For nonconvex objective functions, LAG can guarantee the following convergence result.

Theorem 3 (nonconvex case)**.** *Under Assumption 1, if α and $\{\xi_d\}$ are chosen properly, then the iterates $\{\boldsymbol{\theta}^k\}$ generated by LAG-WK or LAG-PS satisfy*

$$\min_{1\leq k\leq K} \left\|\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k\right\|^2 = o\left(1/K\right) \quad \text{and} \quad \min_{1\leq k\leq K} \left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 = o\left(1/K\right). \tag{2.27}$$

Theorems 2 and 3 assert that with the judiciously designed lazy gradient aggregation rules, LAG can achieve order of convergence rate identical to GD for general convex and nonconvex smooth objective functions. Furthermore, we next show that in these general cases, LAG still requires fewer communication rounds than GD, under certain conditions on the heterogeneity function $h(\gamma)$.

In the general smooth (possibly nonconvex) case however, we define the communication complexity in terms of achieving ϵ-gradient error; e.g., $\min_{k=1,\cdots,K} \|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\|^2 \leq \epsilon$. Similar to Proposition 1, we present the communication complexity as follows.

Proposition 2. *2[communication complexity] Under Assumption 1, with $\Delta\bar{\mathbb{C}}(h;\{\gamma_d\})$ defined as in Proposition 1, the communication complexity of LAG denoted as $\mathbb{C}_{\mathrm{N-LAG}}(\epsilon)$ is bounded by*

$$\mathbb{C}_{\mathrm{N-LAG}}(\epsilon) \leq \left(1 - \Delta\bar{\mathbb{C}}(h;\{\gamma_d\})\right)\frac{\mathbb{C}_{\mathrm{N-GD}}(\epsilon)}{(1 - \sum_{d=1}^{D}\xi_d)} \tag{2.28}$$

where $\mathbb{C}_{\mathrm{N-GD}}(\epsilon)$ is the communication complexity of GD. Choosing the parameters as (2.19)*, if the heterogeneity function $h(\gamma)$ satisfies that there exists γ' such that $\gamma' < \frac{h(\gamma')}{(D+1)DM^2}$, then we have that*

$$\mathbb{C}_{\mathrm{N-LAG}}(\epsilon) < \mathbb{C}_{\mathrm{N-GD}}(\epsilon). \tag{2.29}$$

Along with Proposition 1, we have shown that for strongly convex, convex, and nonconvex smooth objective functions, LAG enjoys provably lower communication overhead relative to GD in certain heterogeneous learning settings. In fact, the LAG's empirical performance gain over GD goes far beyond the above worst-case theoretical analysis, and lies in a much broader distributed learning setting, which is confirmed by the subsequent numerical tests.

2.4 Numerical tests

To validate the theoretical results, this section evaluates the empirical performance of LAG in linear and logistic regression tasks. All experiments were performed using MATLAB on an Intel CPU @ 3.4 GHz (32 GB RAM) desktop.

For linear regression task, consider the square loss function at worker m as

$$\mathcal{L}_m(\boldsymbol{\theta}) := \sum_{n \in \mathcal{N}_m} \left(y_n - \mathbf{x}_n^\top \boldsymbol{\theta} \right)^2 \tag{2.30}$$

where $\{\mathbf{x}_n, y_n, \forall n \in \mathcal{N}_m\}$ are data at worker m.

Real datasets. Performance is tested on the following benchmark datasets [93]; see Table 2.3.

• **Housing** dataset [63] contains 506 samples $(\mathbf{x}_n, y_n)$ with y_n representing the median value of house price, which is affected by features in $\mathbf{x}_n$ such as per capita crime rate and weighted distances to five Boston employment centers.

• **Body fat** dataset contains 252 samples $(\mathbf{x}_n, y_n)$ with y_n describing the percentage of body fat, which is determined by underwater weighing and various body measurements in $\mathbf{x}_n$.

• **Abalone** dataset contains 417 samples $(\mathbf{x}_n, y_n)$ with y_n for the age of abalone and $\mathbf{x}_n$ for the physical measurements of abalone, e.g., sex, height, and shell weight.

Dataset	**# features** (d)	**# samples** (N)	**worker index**
Housing	13	506	1,2,3
Body fat	14	252	4,5,6
Abalone	8	417	7,8,9

Table 2.3: A summary of real datasets used in the linear regression tests.

For logistic regression, consider the binary logistic regression problem

$$\mathcal{L}_m(\boldsymbol{\theta}) := \sum_{n \in \mathcal{N}_m} \log \left(1 + \exp(-y_n \mathbf{x}_n^\top \boldsymbol{\theta}) \right) + \frac{\lambda}{2} \|\boldsymbol{\theta}\|^2. \tag{2.31}$$

Dataset	**# features** (d)	**# samples** (N)	**worker index**
Ionosphere	34	351	1,2,3
Adult fat	113	1605	4,5,6
Derm	34	358	7,8,9

Table 2.4: A summary of real datasets used in the logistic regression tests.

where $\lambda = 10^{-3}$ is the regularization constant.

Real datasets. Performance is tested on the following datasets; see a summary in Table 2.4.

• **Ionosphere** dataset [148] is to predict whether it is a “good” radar return or not – “good” if the features in $\mathbf{x}_n$ show evidence of some structures in the ionosphere.

• **Adult** dataset [79] contains samples that predict whether a person makes over $50K$ a year based on features in $\mathbf{x}_n$ such as work-class, education, and marital-status.

• **Derm** dataset [61] for differential diagnosis of erythemato-squaxous diseases, which is determined by clinical and histopathological attributes in $\mathbf{x}_n$ such as erythema, family history, focal hypergranulosis and melanin incontinence.

By default, we consider one server, and nine workers. Throughout the test, we use the optimality error in objective $\mathcal{L}(\boldsymbol{\theta}^k) - \mathcal{L}(\boldsymbol{\theta}^*)$ as figure of merit of our solution. To benchmark LAG, we consider the following approaches.

▷ **Cyc-IAG** is the cyclic version of the incremental aggregated gradient (IAG) method [16, 60] that resembles the recursion (2.4), but communicates with one worker per iteration cyclically.

▷ **Num-IAG** also resembles the recursion (2.4), but it randomly selects one worker to obtain a fresh gradient per iteration with the probability of choosing worker m equal to $L_m / \sum_{m\in\mathcal{M}} L_m$.

▷ **Batch-GD** is the GD iteration (2.2) that communicates with all the workers per iteration.

For LAG-WK, we choose $\xi_d = \xi = 1/D$ with $D = 10$, and for LAG-PS, we choose more aggressive $\xi_d = \xi = 10/D$ with $D = 10$. Stepsizes for LAG-WK, LAG-PS, and GD are chosen as $\alpha = 1/L$; to optimize performance and guarantee stability, stepsizes for Cyc-IAG and Num-IAG are chosen as $\alpha = 1/(ML)$. For the linear regression task, no regularization is added; for the logistic regression task, the ℓ_2-regularization parameter is set to $\lambda = 10^{-3}$.

We consider two **synthetic data** tests: a) linear regression with increasing smoothness constants, e.g., $L_m = (1.3^{m-1} + 1)^2$, $\forall m$; and, b) logistic regression with uniform smoothness constants, e.g., $L_1 = \ldots = L_9 = 4$. For each worker, we generate 50 samples $\mathbf{x}_n \in \mathbb{R}^{50}$ from the standard Gaussian distribution, and rescale the data to mimic the increasing and uniform smoothness constants. For the case of increasing L_m, it is not surprising that both LAG variants need fewer communication rounds; see Figure 2.3. Interesting enough, for uniform L_m, LAG-WK

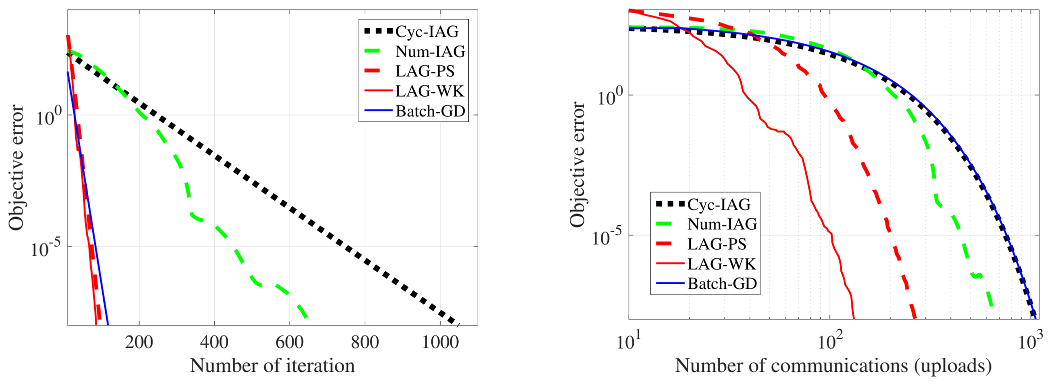

Figure 2.3: Iteration and communication complexity in synthetic datasets with increasing L_m.

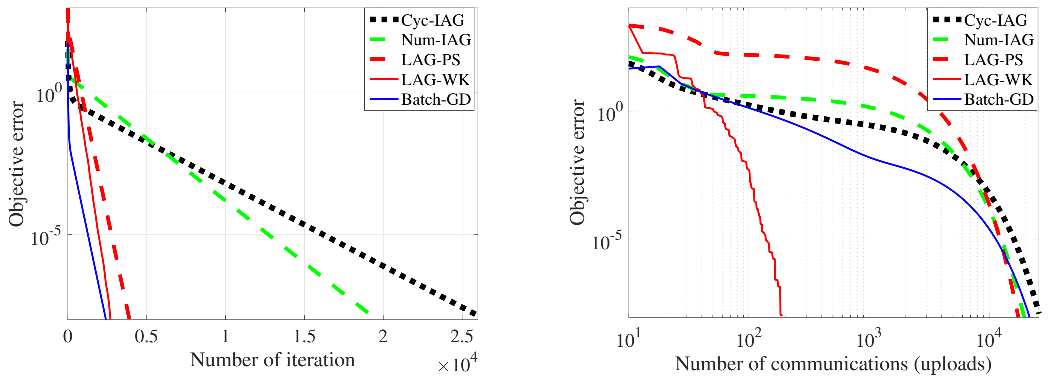

Figure 2.4: Iteration and communication complexity in synthetic datasets with uniform L_m.

still has marked improvements on communication, thanks to its ability of exploiting the *hidden* smoothness of the loss functions; that is, the local curvature of $\mathcal{L}_m$ may not be as steep as L_m; see Figure 2.4.

Performance is also tested on the **real datasets** [93]: a) linear regression using **Housing**, **Body fat**, **Abalone** datasets; and, b) logistic regression using **Ionosphere**, **Adult**, **Derm** datasets; see Figures 2.5-2.6. Each dataset is evenly split into three workers with the number of features used in the test equal to the minimal number of features among all datasets. In all tests, LAG-WK outperforms the alternatives in terms of both metrics, especially reducing the needed communication rounds by several orders of magnitude. Its needed communication rounds can be even *smaller* than the number of iterations, if none of workers violate the trigger condition (2.15) at certain iterations. Additional tests on real datasets under different number of workers are listed in Table 2.5. Under all the tested settings, LAG-WK consistently achieves the lowest communication complexity,

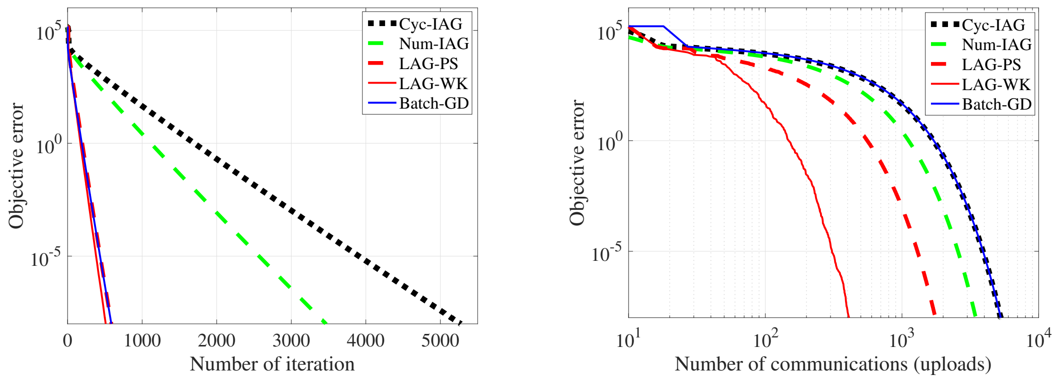

Figure 2.5: Iteration and communication complexity for linear regression in real datasets.

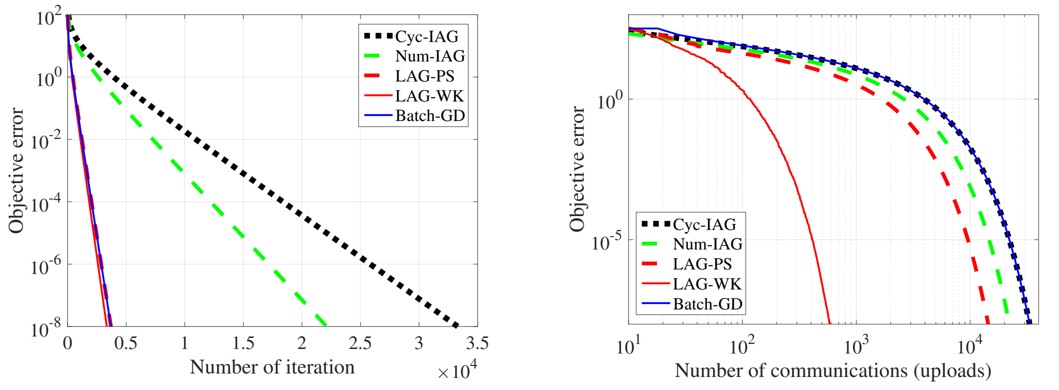

Figure 2.6: Iteration and communication complexity for logistic regression in real datasets.

which corroborates the effectiveness of LAG when it comes to communication reduction.

Similar performance gain has also been observed in the test on a larger dataset **Gisette**. The Gisette dataset was constructed from the MNIST data [85]. After random selecting subset of samples and eliminating all-zero features, it contains 2000 samples $\mathbf{x}_n \in \mathbb{R}^{4837}$. We randomly split this dataset into nine workers. The performance of all the algorithms is reported in Figure 2.7 in terms of the iteration and communication complexity. Clearly, LAG-WK and LAG-PS achieve the same iteration complexity as GD, and outperform Cyc- and Num-IAG. Regarding communication complexity, two LAG variants reduce the needed communication rounds by several orders of magnitude compared with the alternatives.

	Linear regression			Logistic regression		
Algorithm	$M = 9$	$M = 18$	$M = 27$	$M = 9$	$M = 18$	$M = 27$
Cyclic-IAG	5271	10522	15773	33300	65287	97773
Num-IAG	3466	5283	5815	22113	30540	37262
LAG-PS	**1756**	**3610**	**5944**	**14423**	**29968**	**44598**
LAG-WK	**412**	**657**	**1058**	**584**	**1098**	**1723**
Batch GD	5283	10548	15822	33309	65322	97821

Table 2.5: Communication complexity ($\epsilon = 10^{-8}$) under different number of workers.

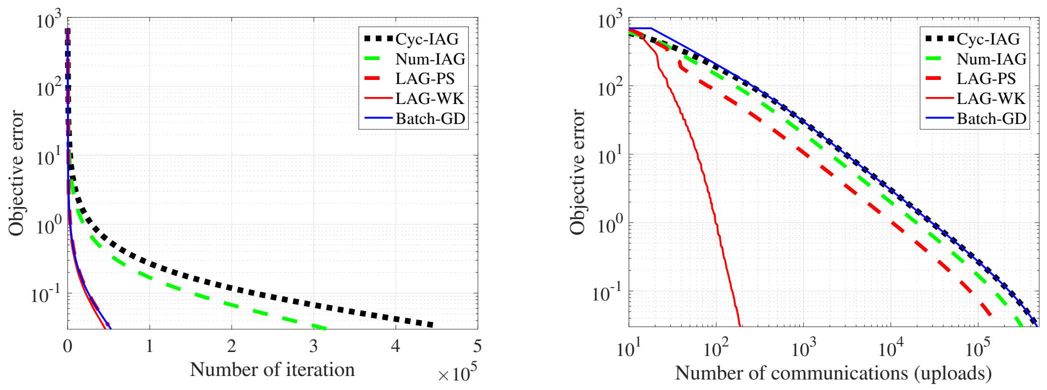

Figure 2.7: Iteration and communication complexity in Gisette dataset.

2.5 Proofs of lemmas and theorems

2.5.1 Proof of Lemma 2

Using the smoothness of $\mathcal{L}(\cdot)$ in Assumption 1, we have that

$$\mathcal{L}(\boldsymbol{\theta}^{k+1}) - \mathcal{L}(\boldsymbol{\theta}^k) \leq \left\langle \nabla \mathcal{L}(\boldsymbol{\theta}^k), \boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k \right\rangle + \frac{L}{2} \left\| \boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k \right\|^2. \tag{2.32}$$

Plugging (2.6) into $\langle \nabla \mathcal{L}(\boldsymbol{\theta}^k), \boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k \rangle$ leads to (cf. $\hat{\boldsymbol{\theta}}_m^k = \hat{\boldsymbol{\theta}}_m^{k-1}$, $\forall m \in \mathcal{M}_c^k$)

$$\begin{aligned}
&\left\langle \nabla \mathcal{L}(\boldsymbol{\theta}^k), \boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k \right\rangle \\
=& -\alpha \left\langle \nabla \mathcal{L}(\boldsymbol{\theta}^k), \nabla \mathcal{L}(\boldsymbol{\theta}^k) + \sum_{m \in \mathcal{M}_c^k} \left(\nabla \mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^k) - \nabla \mathcal{L}_m(\boldsymbol{\theta}^k) \right) \right\rangle \\
=& -\alpha \left\| \nabla \mathcal{L}(\boldsymbol{\theta}^k) \right\|^2 - \alpha \left\langle \nabla \mathcal{L}(\boldsymbol{\theta}^k), \sum_{m \in \mathcal{M}_c^k} \left(\nabla \mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^k) - \nabla \mathcal{L}_m(\boldsymbol{\theta}^k) \right) \right\rangle
\end{aligned}$$

$$= -\alpha\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + \left\langle -\sqrt{\alpha}\nabla\mathcal{L}(\boldsymbol{\theta}^k), \sqrt{\alpha}\sum_{m\in\mathcal{M}_c^k}\left(\nabla\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^k) - \nabla\mathcal{L}_m(\boldsymbol{\theta}^k)\right)\right\rangle. \tag{2.33}$$

Using $2\mathbf{a}^\top\mathbf{b} = \|\mathbf{a}\|^2 + \|\mathbf{b}\|^2 - \|\mathbf{a}-\mathbf{b}\|^2$, we can re-write the inner product in (2.33) as

$$\begin{aligned}
&\left\langle -\sqrt{\alpha}\nabla\mathcal{L}(\boldsymbol{\theta}^k), \sqrt{\alpha}\sum_{m\in\mathcal{M}_c^k}\left(\nabla\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^k) - \nabla\mathcal{L}_m(\boldsymbol{\theta}^k)\right)\right\rangle \\
=&\frac{\alpha}{2}\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + \frac{\alpha}{2}\left\|\sum_{m\in\mathcal{M}_c^k}\left(\nabla\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^k) - \nabla\mathcal{L}_m(\boldsymbol{\theta}^k)\right)\right\|^2 \\
&\qquad - \frac{1}{2}\left\|\sqrt{\alpha}\nabla\mathcal{L}(\boldsymbol{\theta}^k) + \sqrt{\alpha}\sum_{m\in\mathcal{M}_c^k}\left(\nabla\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^k) - \nabla\mathcal{L}_m(\boldsymbol{\theta}^k)\right)\right\|^2 \\
\overset{(a)}{=}&\frac{\alpha}{2}\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + \frac{\alpha}{2}\left\|\sum_{m\in\mathcal{M}_c^k}\left(\nabla\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^k) - \nabla\mathcal{L}_m(\boldsymbol{\theta}^k)\right)\right\|^2 - \frac{1}{2\alpha}\left\|\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k\right\|^2
\end{aligned} \tag{2.34}$$

where (a) follows from the LAG update (2.6).

Combining (2.33) and (2.34), and plugging into (2.32), the claim of Lemma 2 follows.

2.5.2 Proof of Lemma 3

Using the definition of $\mathbb{V}^k$ in (2.16), it follows that

$$\begin{aligned}
\mathbb{V}^{k+1} - \mathbb{V}^k =& \mathcal{L}(\boldsymbol{\theta}^{k+1}) - \mathcal{L}(\boldsymbol{\theta}^k) + \sum_{d=1}^{D}\beta_d\left\|\boldsymbol{\theta}^{k+2-d} - \boldsymbol{\theta}^{k+1-d}\right\|^2 - \sum_{d=1}^{D}\beta_d\left\|\boldsymbol{\theta}^{k+1-d} - \boldsymbol{\theta}^{k-d}\right\|^2 \\
\overset{(a)}{\leq}& -\frac{\alpha}{2}\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + \frac{\alpha}{2}\left\|\sum_{m\in\mathcal{M}_c^k}\left(\nabla\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^k) - \nabla\mathcal{L}_m(\boldsymbol{\theta}^k)\right)\right\|^2 + \sum_{d=2}^{D}\beta_d\left\|\boldsymbol{\theta}^{k+2-d} - \boldsymbol{\theta}^{k+1-d}\right\|^2 \\
&+ \left(\frac{L}{2} - \frac{1}{2\alpha} + \beta_1\right)\left\|\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k\right\|^2 - \sum_{d=1}^{D}\beta_d\left\|\boldsymbol{\theta}^{k+1-d} - \boldsymbol{\theta}^{k-d}\right\|^2
\end{aligned} \tag{2.35}$$

where (a) uses (2.8) in Lemma 2.

Decomposing the square distance as

$$\left\|\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k\right\|^2 = \left\|\alpha\nabla\mathcal{L}(\boldsymbol{\theta}^k) + \alpha\sum_{m\in\mathcal{M}_c^k}\left(\nabla\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^k) - \nabla\mathcal{L}_m(\boldsymbol{\theta}^k)\right)\right\|^2 \tag{2.36}$$

$$\overset{(b)}{\leq} (1+\rho)\,\alpha^2 \left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + \left(1+\rho^{-1}\right)\alpha^2 \left\| \sum_{m\in\mathcal{M}_c^k}\Big(\nabla\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^k) - \nabla\mathcal{L}_m(\boldsymbol{\theta}^k)\Big)\right\|^2$$

where (b) follows from Young's inequality. Plugging (2.36) into (2.35), we arrive at

$$\begin{aligned}\mathbb{V}^{k+1} - \mathbb{V}^k \leq & \left(\left(\frac{L}{2} - \frac{1}{2\alpha} + \beta_1\right)(1+\rho)\,\alpha^2 - \frac{\alpha}{2}\right)\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 \\ & + \sum_{d=1}^{D-1}(\beta_{d+1} - \beta_d)\left\|\boldsymbol{\theta}^{k+1-d} - \boldsymbol{\theta}^{k-d}\right\|^2 - \beta_D\left\|\boldsymbol{\theta}^{k+1-D} - \boldsymbol{\theta}^{k-D}\right\|^2 \\ & + \left(\left(\frac{L}{2} - \frac{1}{2\alpha} + \beta_1\right)\left(1+\rho^{-1}\right)\alpha^2 + \frac{\alpha}{2}\right)\left\|\sum_{m\in\mathcal{M}_c^k}\Big(\nabla\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^k) - \nabla\mathcal{L}_m(\boldsymbol{\theta}^k)\Big)\right\|^2.\end{aligned} \tag{2.37}$$

Using $(\sum_{n=1}^N a_n)^2 \leq N\sum_{n=1}^N a_n^2$, it follows that

$$\left\|\sum_{m\in\mathcal{M}_c^k}\Big(\nabla\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^k) - \nabla\mathcal{L}_m(\boldsymbol{\theta}^k)\Big)\right\|^2 \leq \left|\mathcal{M}_c^k\right| \sum_{m\in\mathcal{M}_c^k}\left\|\nabla\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^k) - \nabla\mathcal{L}_m(\boldsymbol{\theta}^k)\right\|^2 \tag{2.38a}$$

$$\overset{(c)}{\leq} \left|\mathcal{M}_c^k\right| \sum_{m\in\mathcal{M}_c^k} L_m^2\left\|\hat{\boldsymbol{\theta}}_m^k - \boldsymbol{\theta}^k\right\|^2 \tag{2.38b}$$

$$\overset{(d)}{\leq} \frac{|\mathcal{M}_c^k|^2}{\alpha^2|\mathcal{M}|^2}\sum_{d=1}^{D}\xi_d\left\|\boldsymbol{\theta}^{k+1-d} - \boldsymbol{\theta}^{k-d}\right\|^2 \tag{2.38c}$$

where (c) follows the smoothness condition in Assumption 1, and (d) uses the trigger condition (2.15a) if we derive from (2.38a) to (2.38c), uses (2.15b) if we derive from (2.38b) to (2.38c).

Plugging (2.38) into (2.37), we have

$$\begin{aligned}&\mathbb{V}^{k+1} - \mathbb{V}^k \\ \leq & \left(\left(\frac{L}{2} - \frac{1}{2\alpha} + \beta_1\right)(1+\rho)\,\alpha^2 - \frac{\alpha}{2}\right)\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 \\ & + \sum_{d=1}^{D-1}\left(\left(\left(\frac{L}{2} - \frac{1}{2\alpha} + \beta_1\right)\left(1+\rho^{-1}\right)\alpha^2 + \frac{\alpha}{2}\right)\frac{\xi_d\left|\mathcal{M}_c^k\right|^2}{\alpha^2|\mathcal{M}|^2} - \beta_d + \beta_{d+1}\right)\left\|\boldsymbol{\theta}^{k+1-d} - \boldsymbol{\theta}^{k-d}\right\|^2 \\ & + \left(\left(\left(\frac{L}{2} - \frac{1}{2\alpha} + \beta_1\right)\left(1+\rho^{-1}\right)\alpha^2 + \frac{\alpha}{2}\right)\frac{\xi_D\left|\mathcal{M}_c^k\right|^2}{\alpha^2|\mathcal{M}|^2} - \beta_D\right)\left\|\boldsymbol{\theta}^{k+1-D} - \boldsymbol{\theta}^{k-D}\right\|^2.\end{aligned} \tag{2.39}$$

After defining some constants to simplify the notation, the proof is then complete.

Furthermore, if the stepsize α, parameters $\{\beta_d\}$, and the trigger constants $\{\xi_d\}$ satisfy

$$\left(\frac{L}{2}-\frac{1}{2\alpha}+\beta_1\right)(1+\rho)\,\alpha^2-\frac{\alpha}{2}\leq 0 \tag{2.40a}$$

$$\left(\left(\frac{L}{2}-\frac{1}{2\alpha}+\beta_1\right)\left(1+\rho^{-1}\right)\alpha^2+\frac{\alpha}{2}\right)\frac{\xi_d\left|\mathcal{M}_c^k\right|^2}{\alpha^2|\mathcal{M}|^2}-\beta_d+\beta_{d+1}\leq 0,\ \forall d=1,\ldots,D-1 \tag{2.40b}$$

$$\left(\left(\frac{L}{2}-\frac{1}{2\alpha}+\beta_1\right)\left(1+\rho^{-1}\right)\alpha^2+\frac{\alpha}{2}\right)\frac{\xi_D\left|\mathcal{M}_c^k\right|^2}{\alpha^2|\mathcal{M}|^2}-\beta_D\leq 0 \tag{2.40c}$$

then Lyapunov function is non-increasing; that is, $\mathbb{V}^{k+1}\leq\mathbb{V}^k$.

Choice of parameters. We discuss several choices of parameters that satisfy (2.40).

• If $\beta_1=\frac{1-\alpha L}{2\alpha}$ so that $\frac{L}{2}-\frac{1}{2\alpha}+\beta_1=0$, after rearranging terms, (2.40) is equivalent to

$$\alpha\leq\frac{1}{L};\ \ \xi_d\leq\frac{2\alpha(\beta_d-\beta_{d+1})|\mathcal{M}|^2}{|\mathcal{M}_c^k|^2},\ \forall d\in[1,D-1];\ \ \xi_D\leq\frac{2\alpha\beta_D|\mathcal{M}|^2}{|\mathcal{M}_c^k|^2}. \tag{2.41}$$

• If $\beta_1\neq\frac{1-\alpha L}{2\alpha}$, after rearranging terms, (2.40) is equivalent to

$$\alpha\leq\frac{1+(1+\rho)^{-1}}{L+2\beta_1}; \tag{2.42a}$$

$$\xi_d\leq\frac{2\alpha(\beta_d-\beta_{d+1})|\mathcal{M}|^2}{\left((1+\rho^{-1})(2\alpha\beta_1+\alpha L-1)+1\right)|\mathcal{M}_c^k|^2},\quad d=1,\ldots,D-1 \tag{2.42b}$$

$$\xi_D\leq\frac{2\alpha\beta_D|\mathcal{M}|^2}{\left((1+\rho^{-1})(2\alpha\beta_1+\alpha L-1)+1\right)|\mathcal{M}_c^k|^2}. \tag{2.42c}$$

i) If $\rho\to 0$ and $\beta_1\to 0$, (2.42a) becomes $0\leq\alpha\leq 2/L$, matching the stepsize region of GD.

ii) If $\alpha=1/L$ and $\beta_1>0$, (2.42b) and (2.42c) reduce to

$$\xi_d\leq\frac{2\alpha(\beta_d-\beta_{d+1})|\mathcal{M}|^2}{(2\alpha\beta_1(1+\rho^{-1})+1)|\mathcal{M}_c^k|^2}\ \ \text{and}\ \ \xi_D\leq\frac{2\alpha\beta_D|\mathcal{M}|^2}{(2\alpha\beta_1(1+\rho^{-1})+1)\,|\mathcal{M}_c^k|^2}. \tag{2.43}$$

Since (2.41) is in a simpler form, we will use this choice in the subsequent iteration and communication analysis for brevity.

2.5.3 Proof of Theorem 1

Using Lemma 3, it follows that (with $\tilde{c}(\alpha,\beta_1) := \frac{L}{2} - \frac{1}{2\alpha} + \beta_1$)

$$\begin{aligned}
&\mathbb{V}^{k+1} - \mathbb{V}^k \\
\leq& -\left(\frac{\alpha}{2} - \tilde{c}(\alpha,\beta_1)\left(1+\rho\right)\alpha^2\right)\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 \\
&-\left(\beta_D - \left(\tilde{c}(\alpha,\beta_1)\left(1+\rho^{-1}\right)\alpha^2 + \frac{\alpha}{2}\right)\frac{\xi_D\left|\mathcal{M}_c^k\right|^2}{\alpha^2|\mathcal{M}|^2}\right)\left\|\boldsymbol{\theta}^{k+1-D} - \boldsymbol{\theta}^{k-D}\right\|^2 \\
&-\sum_{d=1}^{D-1}\left(\beta_d - \beta_{d+1} - \left(\tilde{c}(\alpha,\beta_1)\left(1+\rho^{-1}\right)\alpha^2 + \frac{\alpha}{2}\right)\frac{\xi_d\left|\mathcal{M}_c^k\right|^2}{\alpha^2|\mathcal{M}|^2}\right)\left\|\boldsymbol{\theta}^{k+1-d} - \boldsymbol{\theta}^{k-d}\right\|^2 \\
\overset{(a)}{\leq}& -\left(\alpha\mu - 2\tilde{c}(\alpha,\beta_1)\left(1+\rho\right)\mu\alpha^2\right)\left(\mathcal{L}(\boldsymbol{\theta}^k) - \mathcal{L}(\boldsymbol{\theta}^*)\right) \\
&-\left(\beta_D - \left(\tilde{c}(\alpha,\beta_1)\left(1+\rho^{-1}\right)\alpha^2 + \frac{\alpha}{2}\right)\frac{\xi_D\left|\mathcal{M}_c^k\right|^2}{\alpha^2|\mathcal{M}|^2}\right)\left\|\boldsymbol{\theta}^{k+1-D} - \boldsymbol{\theta}^{k-D}\right\|^2 \\
&-\sum_{d=1}^{D-1}\left(\beta_d - \beta_{d+1} - \left(\tilde{c}(\alpha,\beta_1)\left(1+\rho^{-1}\right)\alpha^2 + \frac{\alpha}{2}\right)\frac{\xi_d\left|\mathcal{M}_c^k\right|^2}{\alpha^2|\mathcal{M}|^2}\right)\left\|\boldsymbol{\theta}^{k+1-d} - \boldsymbol{\theta}^{k-d}\right\|^2 \qquad (2.44)
\end{aligned}$$

where (a) uses the strong convexity or the PL condition in Assumption 3, e.g.,

$$2\mu\left(\mathcal{L}(\boldsymbol{\theta}^k) - \mathcal{L}(\boldsymbol{\theta}^*)\right) \leq \left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2. \qquad (2.45)$$

With the constant $c(\alpha;\{\xi_d\})$ defined as

$$\begin{aligned}
c(\alpha;\{\xi_d\}) := \min_k \min_{d=1,\dots,D-1}&\left\{\alpha\mu - 2\tilde{c}(\alpha,\beta_1)\left(1+\rho\right)\mu\alpha^2, 1 - \left(\tilde{c}(\alpha,\beta_1)\left(1+\rho^{-1}\right)\alpha^2 + \frac{\alpha}{2}\right)\frac{\xi_D\left|\mathcal{M}_c^k\right|^2}{\alpha^2\beta_D|\mathcal{M}|^2},\right. \\
&\left. 1 - \frac{\beta_{d+1}}{\beta_d} - \left(\tilde{c}(\alpha,\beta_1)\left(1+\rho^{-1}\right)\alpha^2 + \frac{\alpha}{2}\right)\frac{\xi_d\left|\mathcal{M}_c^k\right|^2}{\alpha^2\beta_d|\mathcal{M}|^2}\right\} \qquad (2.46)
\end{aligned}$$

we have from (2.44) that

$$\begin{aligned}
\mathbb{V}^{k+1} - \mathbb{V}^k \overset{(b)}{\leq}& -c(\alpha;\{\xi_d\})\left(\mathcal{L}(\boldsymbol{\theta}^k) - \mathcal{L}(\boldsymbol{\theta}^*) + \sum_{d=1}^{D}\beta_d\left\|\boldsymbol{\theta}^{k+1-d} - \boldsymbol{\theta}^{k-d}\right\|^2\right) \\
=& -c(\alpha;\{\xi_d\})\mathbb{V}^k. \qquad (2.47)
\end{aligned}$$

Rearranging terms in (2.47), we can conclude that

$$\mathbb{V}^{k+1} \leq (1 - c(\alpha; \{\xi_d\}))\, \mathbb{V}^k. \tag{2.48}$$

The Q-linear convergence of $\mathbb{V}^k$ implies the R-linear convergence of $\mathcal{L}(\boldsymbol{\theta}^k) - \mathcal{L}(\boldsymbol{\theta}^*)$. The proof is then complete.

Iteration complexity. Since the linear rate constant in (2.48) is in a complex form, we discuss the iteration complexity under a set of specific parameters (not necessarily optimal). Specifically, we choose

$$\xi_1 = \ldots = \xi_D := \xi < \frac{1}{D} \quad \text{and} \quad \alpha := \frac{1 - D\xi/\eta}{L} \quad \text{and} \quad \beta_d := \frac{(D-d+1)\xi}{2\alpha\eta}, \forall d = 1, \cdots, D \tag{2.49}$$

where η is a constant. Clearly, (2.49) satisfies the condition in (2.41).

Plugging (2.49) into (2.46), we have (cf. $\tilde{c}(\alpha, \beta_1) = 0$)

$$\Gamma := 1 - c(\alpha; \{\xi_d\}) = \max_k \max_{d=1,\ldots,D} \left\{ 1 - \frac{1 - D\xi/\eta}{\kappa}, \frac{\eta \left|\mathcal{M}_c^k\right|^2}{|\mathcal{M}|^2}, \frac{D - d + \eta \left|\mathcal{M}_c^k\right|^2 / |\mathcal{M}|^2}{D - d + 1} \right\}. \tag{2.50}$$

If we choose $\eta := \sqrt{D\xi}$ such that $\frac{\eta|\mathcal{M}_c^k|^2}{|\mathcal{M}|^2} < 1$, we can simplify (2.50) as

$$\Gamma = \max_k \left\{ 1 - \frac{1 - \sqrt{D\xi}}{\kappa}, \frac{D - 1 + \sqrt{D\xi} \left|\mathcal{M}_c^k\right|^2 / |\mathcal{M}|^2}{D} \right\} \overset{(a)}{=} 1 - \frac{1 - \sqrt{D\xi}}{\kappa}. \tag{2.51}$$

where (a) holds since we choose $D \leq \kappa$. With the linear convergence rate in (2.51), we can derive the iteration complexity as

$$\begin{aligned}
&\frac{\mathbb{V}^K}{\mathbb{V}^0} \leq \left(1 - \frac{1 - \sqrt{D\xi}}{\kappa}\right)^K \leq \epsilon \\
\Longrightarrow & K \log\left(1 - \frac{1 - \sqrt{D\xi}}{\kappa}\right) \leq \log(\epsilon) \\
\Longrightarrow & \log\left(\frac{1}{\epsilon}\right) \leq K \log\left(1 - \frac{1 - \sqrt{D\xi}}{\kappa}\right)^{-1} \overset{(b)}{\leq} \frac{K}{\frac{\kappa}{1 - \sqrt{D\xi}} - 1} \\
\Longrightarrow & K \geq \frac{\kappa}{1 - \sqrt{D\xi}} \log\left(\epsilon^{-1}\right)
\end{aligned} \tag{2.52}$$

where (b) uses $\log(1 + x) \leq x, \forall x > -1$. Therefore, we can conclude that $\mathbb{I}_{\text{LAG}}(\epsilon) =$

$\frac{\kappa}{1-\sqrt{D\xi}}\log\left(\epsilon^{-1}\right)$.

2.5.4 Proof of Lemma 4

The idea is essentially to show that if (2.21) holds, then for any iteration k, the worker m will not violate the trigger conditions in (2.15) so that does not communicate with the server at the current iteration, if it has communicated with the server at least once during the previous consecutive d iterations.

Suppose at iteration k, the most recent iteration that the worker m did communicate with the server is iteration $k-d'$ with $1 \leq d' \leq d$. Thus, we have $\hat{\boldsymbol{\theta}}_m^{k-1} = \boldsymbol{\theta}^{k-d'}$, which implies that

$$\begin{aligned}
L_m^2\left\|\hat{\boldsymbol{\theta}}_m^{k-1}-\boldsymbol{\theta}^k\right\|^2 &= L_m^2\left\|\boldsymbol{\theta}^{k-d'}-\boldsymbol{\theta}^k\right\|^2 \\
&= d'L^2\mathbb{H}^2(m)\sum_{b=1}^{d'}\left\|\boldsymbol{\theta}^{k+1-b}-\boldsymbol{\theta}^{k-b}\right\|^2 \\
&\overset{(a)}{\leq} \frac{\xi_d}{\alpha^2|\mathcal{M}|^2}\sum_{b=1}^{d'}\left\|\boldsymbol{\theta}^{k+1-b}-\boldsymbol{\theta}^{k-b}\right\|^2 \\
&\overset{(b)}{\leq} \frac{\sum_{b=1}^{d'}\xi_b\left\|\boldsymbol{\theta}^{k+1-b}-\boldsymbol{\theta}^{k-b}\right\|^2}{\alpha^2|\mathcal{M}|^2} + \frac{\sum_{b=d'+1}^{D}\xi_b\left\|\boldsymbol{\theta}^{k+1-b}-\boldsymbol{\theta}^{k-b}\right\|^2}{\alpha^2|\mathcal{M}|^2} \\
&= \text{RHS of (2.15b)}
\end{aligned} \tag{2.53}$$

where (a) follows since the condition (2.21) is satisfied, so that

$$\mathbb{H}^2(m) \leq \frac{\xi_d}{d\alpha^2L^2M^2} \leq \frac{\xi_d}{d'\alpha^2L^2M^2} \tag{2.54}$$

and (b) follows from our choice of $\{\xi_d\}$ such that for $1 \leq d' \leq d$, we have $\xi_d \leq \xi_{d'} \leq \ldots \leq \xi_1$ and $\left\|\boldsymbol{\theta}^{k+1-b}-\boldsymbol{\theta}^{k-b}\right\|^2 \geq 0$. Therefore, the trigger condition (2.15b) does not activate, and the worker m does not communicate with the server at iteration k. With an additional step that $\|\nabla\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^{k-1})-\nabla\mathcal{L}_m(\boldsymbol{\theta}^k)\|^2 \leq L_m^2\|\hat{\boldsymbol{\theta}}_m^{k-1}-\boldsymbol{\theta}^k\|^2$, we can also prove that if $\hat{\boldsymbol{\theta}}_m^{k-1}=\boldsymbol{\theta}^{k-d'}$, the trigger condition (2.15a) does not activate either.

Note that the above argument holds for any $1 \leq d' \leq d$, and thus if (2.21) holds, the worker m communicates with the server at most every other d iterations.

2.5.5 Proof of Proposition 1

The condition of communication reduction given in (2.21) is equivalent to

$$\mathbb{H}^2(m) \leq \frac{\xi_d}{\alpha^2 L^2 |\mathcal{M}|^2 d} := \gamma_d. \tag{2.55}$$

Together with the definition of heterogeneity score function in (2.22), given γ_d, the quantity $h\left(\gamma_d\right)$ essentially lower bounds the percentage of workers that communicate with the server at most every other d iterations; that is at most $K/(d+1)$ times until iteration K.

To calculate the communication complexity of LAG, we split all the workers into $D+1$ subgroups:

$\mathcal{M}_0$ - every worker m that does not satisfy $\mathbb{H}^2(m) < \gamma_1$;

$\ldots$

$\mathcal{M}_d$ - every worker m that does satisfy $\mathbb{H}^2(m) < \gamma_d$ but does not satisfy $\mathbb{H}^2(m) < \gamma_{d+1}$;

$\ldots$

$\mathcal{M}_D$ - every worker m that does satisfy $\mathbb{H}^2(m) < \gamma_D$.

The above splitting is according to our claims in Lemma 4, which splits all the workers without overlapping. The neat thing is that for workers in each subgroup $\mathcal{M}_d$, we can upper bound its communication rounds until the current iteration. Hence, the total communication complexity of LAG is upper bounded by

$$\begin{aligned}
\mathbb{C}_{\mathrm{LAG}}(\epsilon) &= \sum_{m\in\mathcal{M}} \text{Communication rounds of worker } m \\
&= \sum_{d=0}^{D} \text{Total communication rounds of workers in } \mathcal{M}_d \\
&= \sum_{d=0}^{D} |\mathcal{M}_d| \times \frac{\mathbb{I}_{\mathrm{LAG}}(\epsilon)}{d+1} \\
&\overset{(a)}{\leq} \left(1 - h\left(\gamma_1\right) + \frac{1}{2}\Big(h\left(\gamma_1\right) - h\left(\gamma_2\right)\Big) + \ldots + \frac{1}{D+1} h\left(\gamma_D\right)\right) M\, \mathbb{I}_{\mathrm{LAG}}(\epsilon) \\
&= \left(1 - \underbrace{\sum_{d=1}^{D}\left(\frac{1}{d} - \frac{1}{d+1}\right) h\left(\gamma_d\right)}_{\Delta\bar{\mathbb{C}}(h;\{\gamma_d\})}\right) M\, \mathbb{I}_{\mathrm{LAG}}(\epsilon) := \left(1 - \Delta\bar{\mathbb{C}}(h;\{\gamma_d\})\right) M\, \mathbb{I}_{\mathrm{LAG}}(\epsilon)
\end{aligned} \tag{2.56}$$

where (a) uses the definition of subgroups $\{\mathcal{M}_d\}$ and the function $h(\gamma)$ in (2.22).

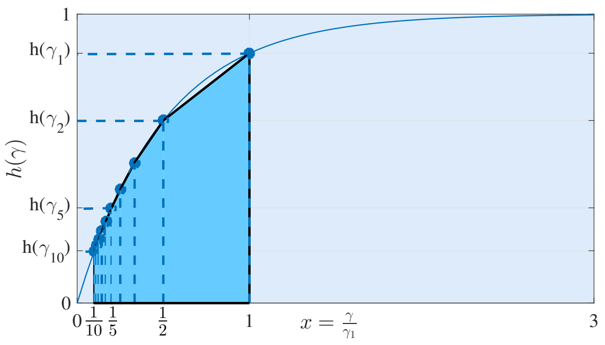

Figure 2.8: The area of the light blue polygon lower bounds the quantity $\Delta\bar{\mathbb{C}}(h;\xi)$ in (2.59). It is generated according to $\gamma_d := 1/(d\gamma_1)$ and $D = 10$.

If we choose the parameters as those in (2.49), we can simplify the expression of (2.56) and arrive at

$$\mathbb{C}_{\mathrm{LAG}}(\epsilon) \leq \left(1 - \Delta\bar{\mathbb{C}}(h;\xi)\right) \frac{M\kappa}{1-\sqrt{D\xi}} \log(\epsilon^{-1}) \tag{2.57}$$

where $\Delta\bar{\mathbb{C}}(h;\{\gamma_d\})$ is written as $\Delta\bar{\mathbb{C}}(h;\xi)$ in this case, because $\gamma_d := \frac{\xi}{(1-\sqrt{D\xi})^2 M^2 d}$, $\forall d$.

On the other hand, even with a larger stepsize $\alpha = 1/L$, the communication complexity of GD is $\mathbb{C}_{\mathrm{GD}}(\epsilon) := M\kappa \log(\epsilon^{-1})$. Therefore, if we can show that

$$\frac{1 - \Delta\bar{\mathbb{C}}(h;\xi)}{1-\sqrt{D\xi}} \leq 1 \quad \Longleftrightarrow \quad \sqrt{D\xi} \leq \Delta\bar{\mathbb{C}}(h;\xi) \tag{2.58}$$

then it is safe to conclude that the communication complexity of LAG is lower than that of GD. Using the nondecreasing property of h, we have that (cf. the area of the light blue polygon in Figure 2.8)

$$\Delta\bar{\mathbb{C}}(h;\xi) \in \left[\frac{Dh(\gamma_D)}{D+1}, \frac{Dh(\gamma_1)}{D+1}\right] \subseteq \left[0, \frac{D}{D+1}\right] \tag{2.59}$$

where we use the fact that $0 \leq h(\gamma) \leq 1$. Since for any $\xi \in (0, 1/D)$, there exists a function h such that $\Delta\bar{\mathbb{C}}(h;\xi)$ achieves any value within $[0, D/(D+1)]$. Therefore, we can conclude that if $\xi \leq \frac{D}{(D+1)^2}$ so that $\sqrt{D\xi} \leq D/(D+1)$, there always exists $h(\gamma)$ or a distributed learning setting such that $\mathbb{C}_{\mathrm{LAG}}(\epsilon) < \mathbb{C}_{\mathrm{GD}}(\epsilon)$.

2.5.6 Proof of Theorem 2

Before establishing the convergence in the convex case, we present a critical lemma.

Lemma 5. *Under Assumptions 1-2, the sequences of Lyapunov functions $\{\mathbb{V}^k\}$ satisfy*

$$\begin{aligned}\left(\mathbb{V}^k\right)^2 \leq & \left(\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + \sum_{d=1}^{D}\beta_d\left\|\boldsymbol{\theta}^{k+1-d}-\boldsymbol{\theta}^{k-d}\right\|^2\right)\left(\left\|\boldsymbol{\theta}^k-\boldsymbol{\theta}^*\right\|^2+\sum_{d=1}^{D}\beta_d\left\|\boldsymbol{\theta}^{k+1-d}-\boldsymbol{\theta}^{k-d}\right\|^2\right)\\ := & \qquad\qquad \overline{\mathbb{V}}^k(1) \qquad\qquad \times \qquad\qquad \overline{\mathbb{V}}^k(2)\end{aligned} \tag{2.60}$$

where $\overline{\mathbb{V}}^k(1)$ and $\overline{\mathbb{V}}^k(2)$ denote the two terms upper bounding $\left(\mathbb{V}^k\right)^2$, respectively.

Proof: Define two vectors as

$$\mathbf{a}^k := \left[\nabla^\top\mathcal{L}(\boldsymbol{\theta}^k), \sqrt{\beta_1}\left\|\boldsymbol{\theta}^k-\boldsymbol{\theta}^{k-1}\right\|,\ldots,\sqrt{\beta_D}\left\|\boldsymbol{\theta}^{k+1-D}-\boldsymbol{\theta}^{k-D}\right\|\right]^\top \tag{2.61a}$$

$$\mathbf{b}^k := \left[(\boldsymbol{\theta}^k-\boldsymbol{\theta}^*)^\top, \sqrt{\beta_1}\left\|\boldsymbol{\theta}^k-\boldsymbol{\theta}^{k-1}\right\|,\ldots,\sqrt{\beta_D}\left\|\boldsymbol{\theta}^{k+1-D}-\boldsymbol{\theta}^{k-D}\right\|\right]^\top. \tag{2.61b}$$

The convexity of $\mathcal{L}(\boldsymbol{\theta})$ implies that

$$\mathcal{L}(\boldsymbol{\theta}^k)-\mathcal{L}(\boldsymbol{\theta}^*) \leq \langle\nabla\mathcal{L}(\boldsymbol{\theta}^k), \boldsymbol{\theta}^k-\boldsymbol{\theta}^*\rangle. \tag{2.62}$$

Recalling the definition of $\mathbb{V}^k$ in (2.16), it follows that

$$\begin{aligned}\mathbb{V}^k &= \mathcal{L}(\boldsymbol{\theta}^k)-\mathcal{L}(\boldsymbol{\theta}^*)+\sum_{d=1}^{D}\beta_d\left\|\boldsymbol{\theta}^{k+1-d}-\boldsymbol{\theta}^{k-d}\right\|^2\\ &\leq \langle\mathbf{a}^k,\mathbf{b}^k\rangle \leq \|\mathbf{a}^k\|\|\mathbf{b}^k\|\end{aligned} \tag{2.63}$$

and squaring both sides of (2.63) leads to

$$\left(\mathbb{V}^k\right)^2 \leq \left(\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + \sum_{d=1}^{D}\beta_d\left\|\boldsymbol{\theta}^{k+1-d}-\boldsymbol{\theta}^{k-d}\right\|^2\right)\left(\left\|\boldsymbol{\theta}^k-\boldsymbol{\theta}^*\right\|^2+\sum_{d=1}^{D}\beta_d\left\|\boldsymbol{\theta}^{k+1-d}-\boldsymbol{\theta}^{k-d}\right\|^2\right) \tag{2.64}$$

from which we can conclude the proof.

Now we are ready to prove Theorem 2. Lemma 3 implies that

$$\begin{aligned}\mathbb{V}^{k+1}-\mathbb{V}^k \leq & -\left(\frac{\alpha}{2}-\tilde{c}(\alpha,\beta_1)\left(1+\rho\right)\alpha^2\right)\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2\\ & -\left(\beta_D-\left(\tilde{c}(\alpha,\beta_1)\left(1+\rho^{-1}\right)\alpha^2+\frac{\alpha}{2}\right)\frac{\xi_D\left|\mathcal{M}_c^k\right|^2}{\alpha^2|\mathcal{M}|^2}\right)\left\|\boldsymbol{\theta}^{k+1-D}-\boldsymbol{\theta}^{k-D}\right\|^2\end{aligned}$$

$$
\begin{aligned}
&-\sum_{d=1}^{D-1}\left(\beta_d-\beta_{d+1}-\left(\tilde{c}(\alpha,\beta_1)\left(1+\rho^{-1}\right)\alpha^2+\frac{\alpha}{2}\right)\frac{\xi_d\left|\mathcal{M}_c^k\right|^2}{\alpha^2|\mathcal{M}|^2}\right)\left\|\boldsymbol{\theta}^{k+1-d}-\boldsymbol{\theta}^{k-d}\right\|^2\\
\le&-c(\alpha;\{\xi_d\})\left(\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2+\sum_{d=1}^{D}\beta_d\left\|\boldsymbol{\theta}^{k+1-d}-\boldsymbol{\theta}^{k-d}\right\|^2\right)\\
=&-c(\alpha;\{\xi_d\})\overline{\mathbb{V}}^k(1)
\end{aligned}
\tag{2.65}
$$

where the definition of $c(\alpha;\{\xi_d\})$ is given by

$$
\begin{aligned}
c(\alpha;\{\xi_d\}):=\min_k\Bigg\{&\frac{\alpha}{2}-\tilde{c}(\alpha,\beta_1)\left(1+\rho\right)\alpha^2, 1-\left(\tilde{c}(\alpha,\beta_1)\left(1+\rho^{-1}\right)\alpha^2+\frac{\alpha}{2}\right)\frac{\xi_D\left|\mathcal{M}_c^k\right|^2}{\alpha^2\beta_D|\mathcal{M}|^2},\\
&1-\frac{\beta_{d+1}}{\beta_d}-\left(\tilde{c}(\alpha,\beta_1)\left(1+\rho^{-1}\right)\alpha^2+\frac{\alpha}{2}\right)\frac{\xi\left|\mathcal{M}_c^k\right|^2}{\alpha^2\beta_d|\mathcal{M}|^2}\Bigg\}.
\end{aligned}
\tag{2.66}
$$

On the other hand, without strong convexity, we can bound $\overline{\mathbb{V}}^k(2)$ as

$$
\overline{\mathbb{V}}^k(2):=\left\|\boldsymbol{\theta}^k-\boldsymbol{\theta}^*\right\|^2+\sum_{d=1}^{D}\beta_d\left\|\boldsymbol{\theta}^{k+1-d}-\boldsymbol{\theta}^{k-d}\right\|^2\le R \tag{2.67}
$$

where the constant R in the last inequality exists since $\mathcal{L}(\boldsymbol{\theta})$ is coercive in Assumption 2 so that $\mathcal{L}(\boldsymbol{\theta}^*)\le\mathcal{L}(\boldsymbol{\theta}^k)<\infty$ implies $\|\boldsymbol{\theta}^k\|<\infty$ thus $\left\|\boldsymbol{\theta}^k-\boldsymbol{\theta}^*\right\|<\infty$ and $\left\|\boldsymbol{\theta}^k-\boldsymbol{\theta}^{k-1}\right\|<\infty$.

Plugging (2.65) and (2.33) into (2.60) in Lemma 5, we have

$$
\left(\mathbb{V}^k\right)^2\le\overline{\mathbb{V}}^k(1)\overline{\mathbb{V}}^k(2)\le\frac{R}{c(\alpha;\{\xi_d\})}(\mathbb{V}^k-\mathbb{V}^{k+1}). \tag{2.68}
$$

Using the fact that the non-increasing property of $\mathbb{V}^k$ in Lemma 3, we have that

$$
\mathbb{V}^{k+1}\mathbb{V}^k\le\left(\mathbb{V}^k\right)^2\le\frac{R}{c(\alpha;\{\xi_d\})}(\mathbb{V}^k-\mathbb{V}^{k+1}). \tag{2.69}
$$

Dividing $\mathbb{V}^{k+1}\mathbb{V}^k$ on both sides of (2.69) and rearranging terms, we have

$$
\frac{c(\alpha;\{\xi_d\})}{R}\le\frac{1}{\mathbb{V}^{k+1}}-\frac{1}{\mathbb{V}^k}. \tag{2.70}
$$

Summing up (2.70), it follows that

$$\frac{Kc(\alpha;\{\xi_d\})}{R} \leq \frac{1}{\mathbb{V}^K} - \frac{1}{\mathbb{V}^0} \leq \frac{1}{\mathbb{V}^K} \tag{2.71}$$

from which we can conclude the proof.

2.5.7 Proof of Theorem 3

Lemma 3 implies that

$$\begin{aligned}\mathbb{V}^{k+1} - \mathbb{V}^k \leq & -\left(\frac{\alpha}{2} - \tilde{c}(\alpha,\beta_1)\left(1+\rho\right)\alpha^2\right)\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 \\ & -\left(\beta_D - \left(\tilde{c}(\alpha,\beta_1)\left(1+\rho^{-1}\right)\alpha^2 + \frac{\alpha}{2}\right)\frac{\xi_D\left|\mathcal{M}_c^k\right|^2}{\alpha^2|\mathcal{M}|^2}\right)\left\|\boldsymbol{\theta}^{k+1-D} - \boldsymbol{\theta}^{k-D}\right\|^2 \\ & -\sum_{d=1}^{D-1}\left(\beta_d - \beta_{d+1} - \left(\tilde{c}(\alpha,\beta_1)\left(1+\rho^{-1}\right)\alpha^2 + \frac{\alpha}{2}\right)\frac{\xi_d\left|\mathcal{M}_c^k\right|^2}{\alpha^2|\mathcal{M}|^2}\right)\left\|\boldsymbol{\theta}^{k+1-d} - \boldsymbol{\theta}^{k-d}\right\|^2 \\ \leq & -c(\alpha;\{\xi_d\})\left(\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + \sum_{d=1}^{D}\beta_d\left\|\boldsymbol{\theta}^{k+1-d} - \boldsymbol{\theta}^{k-d}\right\|^2\right)\end{aligned} \tag{2.72}$$

Summing up both sides of (2.72), we have

$$c(\alpha;\{\xi_d\})\sum_{k=1}^{K}\left(\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + \sum_{d=1}^{D}\beta_d\left\|\boldsymbol{\theta}^{k+1-d} - \boldsymbol{\theta}^{k-d}\right\|^2\right) \leq \mathbb{V}^1 - \mathbb{V}^{K+1}. \tag{2.73}$$

Taking $K \to \infty$, we have that

$$c(\alpha;\{\xi_d\})\lim_{K\to\infty}\sum_{k=1}^{K}\left(\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + \sum_{d=1}^{D}\beta_d\left\|\boldsymbol{\theta}^{k+1-d} - \boldsymbol{\theta}^{k-d}\right\|^2\right) \leq \mathbb{V}^1 \tag{2.74}$$

where the last inequality holds since the Lyapunov function (2.16) is lower bounded by $\mathbb{V}^k \geq 0$, $\forall k$, and $\mathbb{V}^1 < \infty$. Given the choice of α and $\{\xi_d\}$ in (2.40), the constant in (2.74) is $c(\alpha;\{\xi_d\}) > 0$, and thus two terms in the LHS of (2.74) are summable, which implies that

$$\sum_{k=1}^{\infty}\left\|\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k\right\|^2 < \infty \tag{2.75}$$

and likewise that

$$\sum_{k=1}^{\infty} \left\|\nabla \mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 < \infty. \tag{2.76}$$

Using the implications of summable sequences in [43, Lemma 3], the theorem follows.

2.5.8 Proof of Proposition 2

Choosing $\beta_d := \frac{1}{2\alpha}\sum_{\tau=d}^{D}\xi_\tau$ in the Lyapunov function (2.16), we have

$$\mathbb{V}^k := \mathcal{L}(\boldsymbol{\theta}^k) - \mathcal{L}(\boldsymbol{\theta}^*) + \sum_{d=1}^{D} \frac{(\sum_{j=d}^{D}\xi_j)}{2\alpha}\|\boldsymbol{\theta}^{k+1-d} - \boldsymbol{\theta}^{k-d}\|^2 \tag{2.77}$$

Using Lemma 2, we arrive at

$$\mathbb{V}^{k+1} - \mathbb{V}^k \le -\frac{\alpha}{2}\left\|\nabla \mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + \left(\frac{L}{2} - \frac{1}{2\alpha} + \frac{\sum_{d=1}^{D}\xi_d}{2\alpha}\right)\left\|\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k\right\|^2. \tag{2.78}$$

If the stepsize is chosen as $\alpha = \frac{1}{L}(1 - \sum_{d=1}^{D}\xi_d)$, we have

$$\mathbb{V}^{k+1} - \mathbb{V}^k \le -\frac{\alpha}{2}\left\|\nabla \mathcal{L}(\boldsymbol{\theta}^k)\right\|^2. \tag{2.79}$$

Summing up both sides from $k = 1, \ldots, K$, and initializing $\boldsymbol{\theta}^{1-D} = \cdots = \boldsymbol{\theta}^0 = \boldsymbol{\theta}^1$, we have

$$\sum_{k=1}^{K}\left\|\nabla \mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 \le \frac{2}{\alpha}\mathbb{V}^1 = \frac{2}{\alpha}(\mathcal{L}(\boldsymbol{\theta}^1) - \mathcal{L}(\boldsymbol{\theta}^*)) = \frac{2L}{1 - \sum_{d=1}^{D}\xi_d}(\mathcal{L}(\boldsymbol{\theta}^1) - \mathcal{L}(\boldsymbol{\theta}^*)) \tag{2.80}$$

which implies that

$$\min_{k=1,\cdots,K}\left\|\nabla \mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 \le \frac{2L}{(1 - \sum_{d=1}^{D}\xi_d)K}(\mathcal{L}(\boldsymbol{\theta}^1) - \mathcal{L}(\boldsymbol{\theta}^*)) \tag{2.81}$$

With regard to GD, it has the following guarantees [119]

$$\min_{k=1,\cdots,K}\left\|\nabla \mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 \le \frac{2L}{K}(\mathcal{L}(\boldsymbol{\theta}^1) - \mathcal{L}(\boldsymbol{\theta}^*)). \tag{2.82}$$

Thus, to achieve the same ϵ-gradient error, the iteration of LAG is $(1 - \sum_{d=1}^{D}\xi_d)^{-1}$ times than GD. Similar to the derivations in (2.56), since the LAG's average communication rounds per iteration is $(1 - \Delta\bar{\mathbb{C}}(h; \{\gamma_d\}))$ times that of GD, we arrive at (2.28).

If we choose $\xi_1 = \xi_2 = \ldots = \xi_D = \xi$, then $\alpha = \frac{1-D\xi}{L}$, and $\gamma_d = \frac{\xi/d}{\alpha^2 L^2 M^2}$, $d = 1, \ldots, D$. As $h(\cdot)$ is non-decreasing, if $\gamma_D \geq \gamma'$, we have $h(\gamma_D) \geq h(\gamma')$. With the definition of $\Delta\bar{\mathbb{C}}(h; \{\gamma_d\})$ in (2.23), we get

$$\Delta\bar{\mathbb{C}}(h; \{\gamma_d\}) = \sum_{d=1}^{D} \left(\frac{1}{d} - \frac{1}{d+1}\right) h\left(\gamma_d\right) \geq \sum_{d=1}^{D} \left(\frac{1}{d} - \frac{1}{d+1}\right) h\left(\gamma_D\right) \geq \frac{D}{D+1} h(\gamma'). \quad (2.83)$$

Therefore, the total communications are reduced if

$$\left(1 - \frac{D}{D+1} h(\gamma')\right) \cdot \frac{1}{1 - D\xi} < 1 \quad (2.84)$$

which is equivalent to $h(\gamma') > (D+1)\xi$. The condition $\gamma_D \geq \gamma'$ requires

$$\xi/D \geq \gamma'(1 - D\xi)^2 |\mathcal{M}|^2. \quad (2.85)$$

Obviously, if $\xi > \gamma' D |\mathcal{M}|^2$, then (2.85) holds. In summary, we need

$$\gamma' < \frac{\xi}{DM^2} < \frac{h(\gamma')}{(D+1)DM^2}. \quad (2.86)$$

Therefore, we need the function h to satisfy that there exists γ' such that (2.86) holds.

Chapter 3

Federated reinforcement learning over networked agents

3.1 Introduction

Reinforcement learning (RL) involves a sequential decision-making procedure, where a learner takes (possibly randomized) actions in a stochastic environment over a sequence of time steps, and aims to maximize the long-term cumulative rewards received from the interacting environment. Usually modeled as a Markov decision process (MDP) [158], the sequential decision-making process has been tackled by various RL algorithms including Q-learning [169], policy gradient (PG) [159], and actor-critic methods [80]. While these popular RL algorithms are originally developed for the single-learner task, a number of practical RL tasks such as autonomous driving [142], robotics [154] and video games [161], involve *multiple learners* operating in a distributed fashion. In this paper, we consider the distributed reinforcement learning (DRL) problem that covers two general RL settings: *multi-agent collaborative* RL and *parallel* RL. The DRL settings we consider include a central controller that coordinates the learning processes of all learners. The learners can be agents in the multi-agent collaborative RL, or, workers in the parallel RL. In the former setting, multiple agents aim to maximize the team-averaged long-term reward via collaboration in a common environment [76, 165, 180]; while in the latter, multiple parallel machines are used for solving large-scale MDPs with improved exploration and high data efficiency [114, 111]. Similar learning paradigms have been investigated in distributed *supervised learning* [129, 90], e.g., Federated Learning [110], and also in parallel training of large-scale RL tasks [111].

To coordinate the distributed learners, the central controller must exchange information with all learners, by collecting their rewards and local observations, or, broadcasting the policy to them. This type of information exchange requires frequent communication between the controller and the learners. However, in many DRL applications, including cloud-edge AI systems [153], autonomous driving [142], and other applications in IoT [30], the communication is costly and the latency caused by frequent communication becomes the bottleneck of the overall performance. These considerations motivate well the development of communication-efficient approaches for latency-sensitive DRL tasks. Although there has been a surging interest in studying communication-efficient approaches for supervised learning [5, 73, 31], no prior work has focused on the DRL setting. In this context, our goal is to develop a simple yet general algorithm for solving various DRL problems, with provable convergence guarantees and reduced communication overhead.

3.1.1 Our contributions

Targeting a communication-efficient solver for DRL, we propose a new PG method that we term Lazily Aggregated Policy Gradient (LAPG). With judiciously designed communication trigger rules, LAPG is shown capable of: i) achieving the same order of convergence rate (thus iteration complexity) as vanilla PG under standard conditions; and, ii) reducing the communication rounds required to achieve a desirable learning accuracy, when the distributed agents are heterogeneous (meaning reward functions and initial states are not homogeneous). In certain learning settings, we show that LAPG requires only $\mathcal{O}(1/M)$ communication of PG with M denoting the number of learners. Empirically, we evaluate the performance of LAPG using neural network-parameterized policies on the popular multi-agent RL benchmark, and corroborate that LAPG can considerably reduce the communication required by PG.

3.1.2 Related work

PG methods. PG methods have been recognized as one of the most pervasive RL algorithms [158], especially for RL tasks with large and possibly continuous state-action spaces. By parameterizing the infinite-dimensional policy with finite-dimensional vectors [158], PG methods reduce the search for the optimal policy over functional spaces to that over parameter spaces. Early PG methods include the well-known REINFORCE algorithm [171], as well as the variance-reduced G(PO)MDP algorithm [11]. Both REINFORCE and G(PO)MDP are Monte-Carlo sampling-type algorithms that estimate the policy gradient using the rollout trajectory data. To further reduce the variance, a policy gradient estimate that utilizes Q-function approximation was developed in

[159], based on a policy gradient theorem derived therein. Recently, several PG variants have made significant progress in accelerating convergence [74], reducing variance [123], handling continuous action spaces [149], and ensuring policy improvement [124], by employing deep neural networks as function approximators [138, 94, 139]. However, all these algorithms are developed for the single-learner setting.

DRL. DRL has been investigated in the regimes of both multi-agent RL and parallel RL. The studies of multi-agent RL can be traced back to [41] and [172], with applications to network routing [20] and power network control [137]. All these works however, rather heuristically build on the direct modification of Q-learning from a single- to multi-agent settings, without performance guarantees. The first DRL algorithm with convergence guarantees is reported in [84], although tailored for the tabular multi-agent MDP setting. More recently, [76] developed a distributed Q-learning algorithm, termed *QD-learning*, over networked agents that can only communicate with their neighbors. In the same setup, fully decentralized actor-critic algorithms with function approximation were developed in [179, 180] to handle huge or even continuous state-action spaces. From an empirical viewpoint, a number of deep multi-agent collaborative RL algorithms has also been developed [59, 100, 122]. On the other hand, parallel RL, which can efficiently tackle the single-learner yet large-scale RL problem by exploiting parallel computation, has also drawn increasing attention in recent years. In particular, [92] applied the Map Reduce framework to parallelize batch RL methods, while [114] introduced the first massively distributed framework for RL. In [111], asynchronous RL algorithms have also been introduced to solve large-scale MDPs. This parallelism was shown to stabilize the training process, and also benefit data efficiency [111]. Nonetheless, none of these algorithms has dealt with communication-efficiency in DRL.

Communication-efficient learning. Improving communication efficiency in generic distributed learning settings has attracted much attention recently, especially for supervised learning [73, 110]. With their undisputed performance granted, available communication-efficient methods do not directly apply to DRL, because they are either non-stochastic [31, 183], or, they are tailored for convex problems [152]. Algorithms for nonconvex problems are available e.g., [5], but they are designed to minimize the required bandwidth per communication, not the rounds. Compared to communication-efficient supervised learning in [31], the novelty of LAPG here lies in the fact that the gradient used in DRL is stochastic and biased, which requires new algorithmic design and more involved analysis. Another unique feature of RL is that the distribution used to sample data is a function of the time-varying policy parameters, which introduces non-stationarity. Therefore, communication-efficient DRL is a challenging task, and so far it has been an uncharted territory.

3.2 Distributed reinforcement learning

In this section, we present the essential background on DRL and the plain-vanilla PG methods that can be applied to solve the DRL tasks.

3.2.1 Problem statement

Consider a central controller and a group of M distributed learners belonging to a set $\mathcal{M} := \{1, \cdots, M\}$. Depending on the specific DRL setting to be introduced shortly, a learner can be either an agent in the multi-agent collaborative RL, or, a worker in the parallel RL. As in conventional RL, the DRL problem can be characterized under the umbrella of MDP, described by the following tuple

$$(\mathcal{S}, \mathcal{A}, \mathcal{P}, \gamma, \rho, \{\ell_m\}_{m\in\mathcal{M}}) \tag{3.1}$$

where $\mathcal{S}$ and $\mathcal{A}$ are the state space and the action space for all learners, respectively; $\mathcal{P}$ is the space of the state transition kernels defined as $\mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$; $\gamma \in (0, 1)$ is the discounting factor; ρ is the initial state distribution; and $\ell_m : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the local loss (or the negative reward) for learner m.

In addition to the tuple (3.1) that describes an MDP, another important component of MDP is a policy. We consider the stochastic policy $\boldsymbol{\pi} : \mathcal{S} \to \Delta(\mathcal{A})$ that specifies a conditional distribution of all possible joint actions given the current state $\mathbf{s}$, where the probability density of taking the joint action $\mathbf{a}$ is denoted as $\boldsymbol{\pi}(\mathbf{a}|\mathbf{s})$. For the commonly used Gaussian policy [46], it is a function of the state-dependent mean $\boldsymbol{\mu}(\mathbf{s})$ and a covariance matrix $\boldsymbol{\Sigma}$, given by $\boldsymbol{\pi}(\,\cdot\,|\mathbf{s}) = \mathcal{N}(\boldsymbol{\mu}(\mathbf{s}), \boldsymbol{\Sigma})$. In addition to the state-dependent mean, the covariance of a Gaussian policy can be also state-dependent in general; that is, $\boldsymbol{\pi}(\,\cdot\,|\mathbf{s}) = \mathcal{N}(\boldsymbol{\mu}(\mathbf{s}), \boldsymbol{\Sigma}(\mathbf{s}))$. Considering discrete time $t \in \mathbb{N}$ in an infinite horizon, a policy $\boldsymbol{\pi}$ can generate a trajectory of state-action pairs $\mathcal{T} := \{\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \mathbf{s}_2, \mathbf{a}_2, \cdots\}$ with $\mathbf{s}_t \in \mathcal{S}$ and $\mathbf{a}_t \in \mathcal{A}$. In distributed RL, the objective is to find the optimal policy $\boldsymbol{\pi}$ that minimizes the infinite-horizon discounted long-term loss aggregated over all learners, that is

$$\min_{\boldsymbol{\pi}} \sum_{m\in\mathcal{M}} \mathcal{L}_m(\boldsymbol{\pi}) \quad \text{with} \quad \mathcal{L}_m(\boldsymbol{\pi}) := \mathbb{E}_{\mathcal{T}\sim\mathbb{P}(\cdot|\boldsymbol{\pi})}\left[\sum_{t=0}^{\infty} \gamma^t \ell_m(\mathbf{s}_t, \mathbf{a}_t)\right] \tag{3.2}$$

where $\ell_m(\mathbf{s}_t, \mathbf{a}_t)$ and $\mathcal{L}_m(\boldsymbol{\pi})$ are the loss given the state-action pair $(\mathbf{s}_t, \mathbf{a}_t)$ and the cumulative loss for learner m, respectively. The expectation in (3.2) is taken over the random trajectory $\mathcal{T}$. Given a policy $\boldsymbol{\pi}$, the probability of generating trajectory $\mathcal{T}$ is given by

$$\mathbb{P}(\mathcal{T}|\boldsymbol{\pi}) = \mathbb{P}(\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \mathbf{s}_2, \mathbf{a}_2, \cdots|\boldsymbol{\pi}) = \rho(\mathbf{s}_0)\prod_{t=0}^{\infty}\boldsymbol{\pi}(\mathbf{a}_t|\mathbf{s}_t)\mathbb{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) \tag{3.3}$$

where $\rho(\mathbf{s}_0)$ is the probability of initial state being $\mathbf{s}_0$, $\mathbb{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ is the transition probability from the current state $\mathbf{s}_t$ to the next state $\mathbf{s}_{t+1}$ by taking action $\mathbf{a}_t$. Clearly, the trajectory $\mathcal{T}$ is determined by both the underlying MDP and the policy $\boldsymbol{\pi}$.

Depending on how different learners are coupled with each other, the generic DRL formulation (3.2) includes the two popular RL settings, as highlighted below.

Multi-agent collaborative reinforcement learning. A number of important RL applications involve interaction between multiple heterogeneous but collaborative decision-makers (a.k.a. agents), such as those in controlling unmanned aerial vehicle, autonomous driving [142] and many more in future IoT paradigm [30]. In multi-agent collaborative RL, each agent m observes a global state $\mathbf{s}_t \in \mathcal{S}$ shared by all the agents, and takes an action $\mathbf{a}_{m,t} \in \mathcal{A}_m$ with the local action space denoted as $\mathcal{A}_m$. The local action of agent m is generated by a local policy $\boldsymbol{\pi}_m : \mathcal{S} \to \Delta(\mathcal{A}_m)$. While the local action spaces of different agents can be different, agents interact with a common environment that is influenced by the joint actions of all the agents, where the joint action space can be written as $\mathcal{A} := \prod_{m\in\mathcal{M}}\mathcal{A}_m$. In other words, the joint action $(\mathbf{a}_{1,t}, \cdots, \mathbf{a}_{M,t}) \in \mathcal{A}$, not any of the local action $\mathbf{a}_{m,t}$, determines the transition probability to the next state $\mathbf{s}_{t+1}$ as well as the loss of each agent $\ell_m(\mathbf{s}_t, (\mathbf{a}_{1,t}, \cdots, \mathbf{a}_{M,t}))$. As a consequence, the multi-agent collaborative RL problem can be characterized as an MDP using the following tuple $\big(\mathcal{S}, \prod_{m\in\mathcal{M}}\mathcal{A}_m, \mathcal{P}, \gamma, \rho, \{\ell_m\}_{m\in\mathcal{M}}\big)$, which can be formulated in the following form

$$\min_{\boldsymbol{\pi}} \sum_{m\in\mathcal{M}}\mathcal{L}_m(\boldsymbol{\pi}) \quad \text{with} \quad \mathcal{L}_m(\boldsymbol{\pi}) := \mathbb{E}_{\mathcal{T}\sim\mathbb{P}(\cdot|\boldsymbol{\pi})}\left[\sum_{t=0}^{\infty}\gamma^t\ell_m\big(\mathbf{s}_t, (\mathbf{a}_{1,t}, \cdots, \mathbf{a}_{M,t})\big)\right] \tag{3.4}$$

where $\boldsymbol{\pi} := (\boldsymbol{\pi}_1 \cdots, \boldsymbol{\pi}_M)$ is a joint policy that concatenates all the local policies $\{\boldsymbol{\pi}_m\}_{m\in\mathcal{M}}$, and the expectation in $\mathcal{L}_m(\boldsymbol{\pi})$ is taken over all possible joint state-action trajectories, given by $\mathcal{T} := \{\mathbf{s}_0, (\mathbf{a}_{1,0}, \cdots, \mathbf{a}_{M,0}), \mathbf{s}_1, (\mathbf{a}_{1,1}, \cdots, \mathbf{a}_{M,1}), \mathbf{s}_2, (\mathbf{a}_{1,2}, \cdots, \mathbf{a}_{M,2}), \cdots\}$. Replacing the action $\mathbf{a}_t$ in (3.2) by the joint action $(\mathbf{a}_{1,t}, \cdots, \mathbf{a}_{M,t})$, the multi-agent collaborative RL problem can be viewed as an instance of DRL. Different from a single-agent MDP, the agents here are coupled by the state transition that depends on the joint action, and the local loss function that depends on the joint state.

Parallel reinforcement learning. Different from multi-agent RL, parallel RL is motivated by solving a large-scale single-agent RL task that needs to be run in parallel on multiple computing units (a.k.a. workers) [114]. The advantage of parallel RL is that it can reduce the training time and stabilize the training processes [111]. Under such a setting, multiple workers typically aim to learn a common policy $\boldsymbol{\pi} : \mathcal{S} \to \Delta(\mathcal{A})$ for different instances of an *identical* MDP. By different instances of an identical MDP, we mean that each worker m aims to solve an independent MDP characterized by $(\mathcal{S}_m, \mathcal{A}_m, \mathcal{P}_m, \gamma, \rho_m, \ell_m)$. In particular, the local action and state spaces as well as the transition probability of each worker are the same; that is, $\mathcal{A} = \mathcal{A}_m, \mathcal{P} = \mathcal{P}_m,$, and $\mathcal{S} = \mathcal{S}_m$, $\forall m \in \mathcal{M}$. However, the losses and the initial state distributions are different among workers, where the initial state distribution of worker m is ρ_m, and the loss of worker m is $\ell_m : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. Nevertheless, they are quantities drawn from the same distribution, which satisfy $\mathbb{E}[\rho_m(\mathbf{s})] = \rho(\mathbf{s})$ and $\mathbb{E}[\ell_m(\mathbf{s}, \mathbf{a})] = \ell(\mathbf{s}, \mathbf{a})$ for any $(\mathbf{s}, \mathbf{a}) \in \mathcal{S} \times \mathcal{A}$. Therefore, the parallel RL can be written as follows

$$\min_{\boldsymbol{\pi}} \sum_{m \in \mathcal{M}} \mathcal{L}_m(\boldsymbol{\pi}) \quad \text{with} \quad \mathcal{L}_m(\boldsymbol{\pi}) := \mathbb{E}_{\mathcal{T}_m \sim \mathbb{P}(\cdot|\boldsymbol{\pi})} \left[\sum_{t=0}^{\infty} \gamma^t \ell_m(\mathbf{s}_{m,t}, \mathbf{a}_{m,t}) \right] \tag{3.5}$$

where $\mathbf{s}_{m,t} \in \mathcal{S}_m$, $\mathbf{a}_{m,t} \in \mathcal{A}_m$ are the state and action of worker m, and $\boldsymbol{\pi}$ is the common policy to be learned. The expectation in $\mathcal{L}_m(\boldsymbol{\pi})$ is taken over all possible state-action trajectories of worker m, given by $\mathcal{T}_m := \{\mathbf{s}_{m,0}, \mathbf{a}_{m,0}, \mathbf{s}_{m,1}, \mathbf{a}_{m,1}, \mathbf{s}_{m,2}, \mathbf{a}_{m,2}, \cdots\}$. In contrast to the formulation of multi-agent RL in (3.4), the workers in parallel RL are not coupled by the joint state transition distributions or the loss functions, but rather, they are intertwined by employing a common local policy.

3.2.2 Policy gradient methods

Policy gradient methods have been widely used in various RL problems with massive and possibly continuous state and action spaces. In those cases, tabular RL approaches are no longer tractable, and the intended solver typically involves function approximation. To overcome the inherent difficulty of learning a function, policy gradient methods restrict the search for the best performing policy over a class of parametrized policies. In particular, the policy $\boldsymbol{\pi}$ is usually parameterized by $\boldsymbol{\theta} \in \mathbb{R}^d$, which is denoted as $\boldsymbol{\pi}(\cdot|\mathrm{s}; \boldsymbol{\theta})$ or $\boldsymbol{\pi}(\boldsymbol{\theta})$ for simplicity. For example, the commonly used

Gaussian policy can be parameterized as

$$\boldsymbol{\pi}(\,\cdot\,|\mathrm{s};\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu}(\mathrm{s};\boldsymbol{\theta}), \boldsymbol{\Sigma}) \tag{3.6}$$

where $\boldsymbol{\mu}(\mathrm{s};\boldsymbol{\theta})$ is a general nonlinear mapping from $\mathcal{S}$ to $\mathcal{A}$ parameterized by $\boldsymbol{\theta}$. The mapping $\boldsymbol{\mu}(\mathrm{s};\boldsymbol{\theta})$ can either be a deep neural network with the weight parameters $\boldsymbol{\theta}$, or a linear function of $\boldsymbol{\theta}$ of the form $\boldsymbol{\mu}(\mathrm{s};\boldsymbol{\theta}) = \phi(\mathrm{s})^\top\boldsymbol{\theta}$, where $\phi(\mathrm{s})$ is the feature matrix corresponding to the state s. Accordingly, the long-term discounted reward of a parametric policy per agent m is denoted by $\mathcal{L}_m(\boldsymbol{\theta}) := \mathcal{L}_m(\boldsymbol{\pi}(\boldsymbol{\theta}))$. Hence, the DRL problem (3.2) can be rewritten as the following parametric optimization problem

$$\min_{\boldsymbol{\theta}} \sum_{m\in\mathcal{M}} \mathcal{L}_m(\boldsymbol{\theta}) \quad \text{with} \quad \mathcal{L}_m(\boldsymbol{\theta}) := \mathbb{E}_{\mathcal{T}\sim\mathbb{P}(\,\cdot\,|\boldsymbol{\theta})}\left[\sum_{t=1}^{\infty}\gamma^t \ell_m(\mathrm{s}_t, \mathbf{a}_t)\right] \tag{3.7}$$

where the probability distribution of a trajectory $\mathcal{T}$ under the policy $\boldsymbol{\pi}(\boldsymbol{\theta})$ is denoted as $\mathbb{P}(\cdot|\boldsymbol{\theta})$. The search for an optimal policy can thus be performed by applying the gradient descent-type iterative methods to the parameterized optimization problem (3.7). By virtue of the *log-trick*, the gradient of each learner's cumulative loss $\mathcal{L}_m(\boldsymbol{\theta})$ in (3.7) can be written as [11]

Policy gradient
$$\nabla\mathcal{L}_m(\boldsymbol{\theta}) = \mathbb{E}_{\mathcal{T}\sim\mathbb{P}(\,\cdot\,|\boldsymbol{\theta})}\left[\sum_{t=0}^{\infty}\left(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\tau|\mathrm{s}_\tau;\boldsymbol{\theta})\right)\gamma^t\ell_m(\mathrm{s}_t,\mathbf{a}_t)\right]. \tag{3.8}$$

When the MDP model (3.1) is unknown, or, the expectation in (3.8) is computationally difficult to calculate, the stochastic estimate of the policy gradient (3.8) is often used, that is

G(PO)MDP gradient
$$\hat{\nabla}\mathcal{L}_m(\boldsymbol{\theta}) = \sum_{t=0}^{\infty}\left(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\tau|\mathrm{s}_\tau;\boldsymbol{\theta})\right)\gamma^t\ell_m(\mathrm{s}_t,\mathbf{a}_t) \tag{3.9}$$

which is first proposed in [11] abbreviated as G(PO)MDP policy gradient. The G(PO)MDP policy gradient is an unbiased estimator of the policy gradient, while the latter incurs lower variance than other estimators, e.g., REINFORCE [171]. As a result, we will leverage the G(PO)MDP gradient in the ensuing algorithm design and performance analysis.

Even though, the variance of G(PO)MDP gradient is still high in general, which requires using small stepsizes and running sufficiently many iterations to guarantee convergence. For vanilla PG method in DRL, a number of needed iterations result in high communication overhead, since

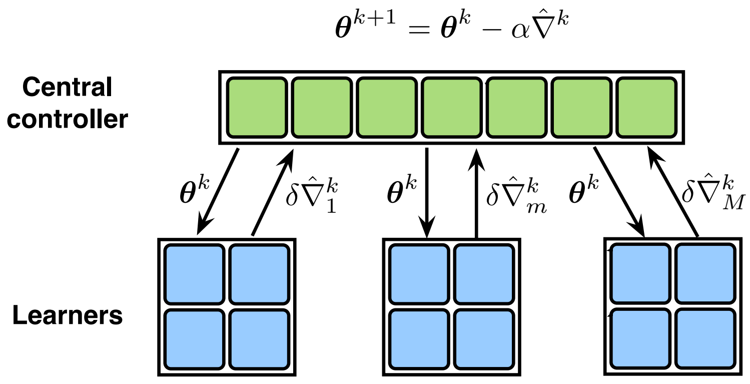

Figure 3.1: LAPG for communication-efficient distributed reinforcement learning.

all learners' gradients need to be uploaded at each iteration in order to form the gradient for the collective objective in (3.2). This motivates the development of communication-efficient DRL algorithms to be introduced next.

3.3 Communication-efficient policy gradient approach

Before introducing our approach, we first revisit the popular G(PO)MDP gradient method for solving (3.7) in the DRL setting: At iteration k, the central controller broadcasts the current policy parameter $\boldsymbol{\theta}^k$ to *all* the learners; every learner $m \in \mathcal{M}$ computes an approximate policy gradient via

$$\hat{\nabla}_{N,T}\mathcal{L}_m\big(\boldsymbol{\theta}^k\big) := \frac{1}{N}\sum_{n=1}^{N}\sum_{t=0}^{T}\left(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\tau^{n,m}|\mathbf{s}_\tau^{n,m};\boldsymbol{\theta}^k)\right)\gamma^t\ell_m(\mathbf{s}_t^{n,m},\mathbf{a}_t^{n,m}) \tag{3.10}$$

where $\mathcal{T}_T^{n,m} := (\mathbf{s}_0^{n,m}, \mathbf{a}_0^{n,m}, \mathbf{s}_1^{n,m}, \mathbf{a}_1^{n,m}, \cdots, \mathbf{s}_T^{n,m}, \mathbf{a}_T^{n,m})$ is the nth T-slot trajectory (a.k.a. episode) generated at learner m; every learner m then uploads $\hat{\nabla}_{N,T}\mathcal{L}_m\big(\boldsymbol{\theta}^k\big)$ to the central controller; and once receiving gradients from all learners, the controller updates the policy parameters via

PG iteration
$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \alpha\hat{\nabla}_{\text{PG}}^k \quad \text{with} \quad \hat{\nabla}_{\text{PG}}^k := \sum_{m\in\mathcal{M}}\hat{\nabla}_{N,T}\mathcal{L}_m(\boldsymbol{\theta}^k) \tag{3.11}$$

where α is a stepsize, and $\hat{\nabla}_{\text{PG}}^k$ is an aggregated policy gradient with each component received from each learner. The policy gradient in (3.10) is a mini-batch G(PO)MDP gradient computed by learner m using N batch trajectories $\{\mathcal{T}_T^{n,m}\}_{n=1}^N$ over T time slots. To implement the mini-batch PG update (3.11) however, the controller has to communicate with *all* learners to obtain fresh

$\{\hat{\nabla}_{N,T}\mathcal{L}_m(\boldsymbol{\theta}^k)\}$.

In this context, the present paper puts forward a new policy gradient-based method for DRL (as simple as PG) that can *skip* communication at certain rounds, which justifies the term **L**azily **A**ggregated **Policy G**radient (**LAPG**). With derivations deferred later, we introduce the LAPG iteration for the DRL problem (3.7) that resembles the PG update (3.11), given by

$$\textbf{LAPG iteration} \qquad \boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \alpha\hat{\nabla}^k \quad \text{with} \quad \hat{\nabla}^k := \sum_{m\in\mathcal{M}} \hat{\nabla}_{N,T}\mathcal{L}_m\big(\hat{\boldsymbol{\theta}}_m^k\big) \tag{3.12}$$

where each policy gradient $\hat{\nabla}_{N,T}\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^k)$ is either $\hat{\nabla}_{N,T}\mathcal{L}_m(\boldsymbol{\theta}^k)$, when $\hat{\boldsymbol{\theta}}_m^k = \boldsymbol{\theta}^k$, or an outdated policy gradient that has been computed using an old copy $\hat{\boldsymbol{\theta}}_m^k \neq \boldsymbol{\theta}^k$. Instead of requesting fresh batch policy gradients from every learner in (3.11), our fresh idea is to obtain $\hat{\nabla}^k$ by refining the previous aggregated gradient $\hat{\nabla}^{k-1}$; e.g., using only the new gradients from the learners in $\mathcal{M}^k$, while reusing the outdated gradients from the rest of the learners. Therefore, with $\hat{\boldsymbol{\theta}}_m^k := \boldsymbol{\theta}^k$, $\forall m \in \mathcal{M}^k$, $\hat{\boldsymbol{\theta}}_m^k := \hat{\boldsymbol{\theta}}_m^{k-1}$, $\forall m \notin \mathcal{M}^k$, LAPG in (3.12) is equivalent to

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \alpha \sum_{m\in\mathcal{M}} \hat{\nabla}_{N,T}\mathcal{L}_m\big(\hat{\boldsymbol{\theta}}_m^{k-1}\big) - \alpha \sum_{m\in\mathcal{M}^k} \Big(\hat{\nabla}_{N,T}\mathcal{L}_m\big(\boldsymbol{\theta}^k\big) - \hat{\nabla}_{N,T}\mathcal{L}_m\big(\hat{\boldsymbol{\theta}}_m^{k-1}\big)\Big) \tag{3.13a}$$

$$:= \boldsymbol{\theta}^k - \alpha\hat{\nabla}^{k-1} - \alpha \sum_{m\in\mathcal{M}^k} \delta\hat{\nabla}_m^k \tag{3.13b}$$

where $\delta\hat{\nabla}_m^k := \hat{\nabla}_{N,T}\mathcal{L}_m(\boldsymbol{\theta}^k) - \hat{\nabla}_{N,T}\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^{k-1})$ denotes the *innovation* between two evaluations of $\hat{\nabla}_{N,T}\mathcal{L}_m$ at the current policy parameter $\boldsymbol{\theta}^k$ and the old copy $\hat{\boldsymbol{\theta}}_m^{k-1}$. Note that the old copies for evaluating policy gradient at each learner can be different here, depending on the most recent iteration that each learner uploads its fresh batch policy gradient.

To this point, a myopic approach to minimizing per-iteration communication is to include as few learners in $\mathcal{M}^k$ as possible. However, it will turn out that such simple selection will lead to much more number of iterations such that increasing the total number of needed communication rounds. A more principled way is to guide the communication selection according to learners' optimization progress. The first step of implementing such principle is to characterize the optimization progress as follows.

Lemma 6 (LAPG descent lemma)**.** *Suppose $\mathcal{L}(\boldsymbol{\theta}) := \sum_{m\in\mathcal{M}} \mathcal{L}_m(\boldsymbol{\theta})$ is L-smooth, and $\boldsymbol{\theta}^{k+1}$ is generated by running one-step LAPG iteration* (3.12) *given $\boldsymbol{\theta}^k$. If the stepsize is selected such that*

Algorithm 3 PG for DRL

```
1: Input: Stepsize $\alpha > 0$, $N$, and $T$.
2: Initialize: $\boldsymbol{\theta}^1$.
3: for $k = 1, 2, \ldots, K$ do
4:     Controller broadcasts $\boldsymbol{\theta}^k$ to all learners.
5:     for learner $m = 1, \ldots, M$ do
6:         Learner $m$ computes $\hat{\nabla}_{N,T}\mathcal{L}_m(\boldsymbol{\theta}^k)$.
7:         Learner $m$ uploads $\hat{\nabla}_{N,T}\mathcal{L}_m(\boldsymbol{\theta}^k)$.
8:     end for
9:     Controller updates the policy via (3.11).
10: end for
```

Algorithm 4 LAPG for DRL

```
1: Input: Stepsize $\alpha > 0$, $\{\xi_d\}$, $N$ and $T$.
2: Initialize: $\boldsymbol{\theta}^1, \hat{\nabla}^0, \{\hat{\boldsymbol{\theta}}_m^0, \forall m\}$.
3: for $k = 1, 2, \ldots, K$ do
4:     Controller broadcasts the current policy.
5:     for learner $m = 1, \ldots, M$ do
6:         Learner $m$ computes $\hat{\nabla}_{N,T}\mathcal{L}_m(\boldsymbol{\theta}^k)$.
7:         if learner $m$ violates the condition then
8:             Learner $m$ uploads $\delta\hat{\nabla}_m^k$.
9:                 ▷ Save $\hat{\boldsymbol{\theta}}_m^k = \boldsymbol{\theta}^k$ at learner $m$
10:        else
11:            No actions at learner $m$.
12:        end if
13:    end for
14:    Controller updates the policy via (3.12).
15: end for
```

Table 3.1: A comparison of PG and LAPG for DRL.

$\alpha \le 1/L$, then the objective values satisfy

$$\mathcal{L}(\boldsymbol{\theta}^{k+1}) - \mathcal{L}(\boldsymbol{\theta}^k) \le -\frac{\alpha}{2}\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + \frac{3\alpha}{2}\left\|\sum_{m\in\mathcal{M}_c^k}\delta\hat{\nabla}_m^k\right\|^2 + \frac{3\alpha}{2}\left\|\hat{\nabla}_{N,T}\mathcal{L}(\boldsymbol{\theta}^k) - \nabla_T\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + \frac{3\alpha}{2}\left\|\nabla_T\mathcal{L}(\boldsymbol{\theta}^k) - \nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 \tag{3.14}$$

where $\delta\hat{\nabla}_m^k$ is defined in (3.13) *and $\mathcal{M}_c^k$ is the set of learners that* do not *upload at iteration k.*

In Lemma 6, the first term on the right hand side of (3.14) drives the descent in the objective of DRL, while the error induced by skipping communication (the second term), the variance of stochastic policy gradient (the third term), as well as the finite-horizon gradient approximation error (the fourth term) increase the DRL objective thus impede the optimization progress. Intuitively, the error induced by skipping communication should be properly controlled so that it is small or even negligible relative to the magnitude of policy gradients that drives the optimization progress, and also the variance of policy gradients that originally appears in the PG-type algorithms [123].

To account for these error terms in our algorithmic design, we first quantify the variance of using mini-batch policy gradient estimation.

Lemma 7 (PG concentration). *Under Assumptions 1 and 2, there exists a constant V_m depending on $G, \gamma, \bar{\ell}_m$ such that given K and $\delta \in (0,1)$, with probability at least $1 - \delta/K$, for any $\boldsymbol{\theta}$ we* *have that*

$$\left\|\hat{\nabla}_{N,T}\mathcal{L}_m\big(\boldsymbol{\theta}\big)-\nabla_T\mathcal{L}_m(\boldsymbol{\theta})\right\|^2\leq\frac{2\log(2K/\delta)V_m^2}{N}:=\sigma^2_{m,N,\delta/K} \tag{3.15}$$

where $\hat{\nabla}_{N,T}\mathcal{L}_m\big(\boldsymbol{\theta}\big)$ *and* $\nabla_T\mathcal{L}_m(\boldsymbol{\theta})$ *are the batch stochastic policy gradient* (3.10)*, and the full policy gradient for the* T*-slot truncated objective* (3.7)*, namely,* $\mathbb{E}_{\mathcal{T}\sim\mathbb{P}(\cdot\,|\boldsymbol{\theta})}\big[\sum_{t=1}^{T}\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\big]$.

Building upon Lemmas 6 and 7, we will include the learner m in $\mathcal{M}^k$ of (3.13) only if its current policy gradient has enough innovation relative to the most recently uploaded one; that is, it satisfies

LAPG condition
$$\left\|\delta\hat{\nabla}_m^k\right\|^2\geq\frac{1}{\alpha^2M^2}\sum_{d=1}^{D}\xi_d\left\|\boldsymbol{\theta}^{k+1-d}-\boldsymbol{\theta}^{k-d}\right\|^2+6\sigma^2_{m,N,\delta/K} \tag{3.16}$$

where $\{\xi_d\}_{d=1}^D$ are constant weights, and $\sigma^2_{m,N,\delta/K}$ is the variance of the policy gradient in (3.15). The values of $\{\xi_d\}$ and D are hyper-parameters and can be optimized case-by-case, and the variance $\sigma^2_{m,N,\delta/K}$ can be estimated on-the-fly in simulations. In a nutshell, a comparison of PG and LAPG for solving the DRL problem (3.7) is summarized in Table 3.1.

Regarding our proposed LAPG method, two remarks are in order.

LAPG implementation. With recursive update of the lagged gradients in (3.12) and the lagged condition in (3.16), implementing LAPG is as simple as PG. The only additional complexity comes from storing the most recently uploaded policy gradient $\hat{\nabla}_{N,T}\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^k)$ and checking the LAPG communication condition (3.16). Despite its simplicity, we will further demonstrate that using lagged policy gradients in DRL can cut down a portion of unnecessary yet costly communication among learners.

Beyond LAPG. Compared with existing efforts for improving the performance of PG in single-agent RL settings such as the trust region PG [138], the deterministic PG [149], and the variance-reduced PG [123], LAPG is not orthogonal to any of them. Instead, LAPG points out an alternating direction for improving communication efficiency of solving DRL, and can be combined with these methods to develop even more powerful DRL schemes. Extension to the actor-critic version of LAPG is also possible to accelerate and stablize the learning processes.

3.4 Finite-sample analysis

In this section, we present the main theorems of LAPG. Before that, we introduce several assumptions that serve as the stepping stone for the subsequent analysis.

Assumption 1: *For each state-action pair* $(\mathbf{s},\mathbf{a})$*, the loss* $\ell_m(\mathbf{s},\mathbf{a})$ *is bounded as* $\ell_m(\mathbf{s},\mathbf{a}) \in [0, \bar{\ell}_m]$*, and thus for each parameter* $\boldsymbol{\theta}$*, the per-learner cumulative loss is bounded as* $\mathcal{L}_m(\boldsymbol{\theta}) \in [0, \bar{\ell}_m/(1-\gamma)]$.

Assumption 2: *For each state-action pair* $(\mathbf{s},\mathbf{a})$*, and any policy parameter* $\boldsymbol{\theta} \in \mathbb{R}^d$*, there exist constants* G *and* F *such that*

$$\|\nabla \log \boldsymbol{\pi}(\mathbf{a}|\mathbf{s};\boldsymbol{\theta})\| \leq G \quad \text{and} \quad \left|\frac{\partial^2}{\partial\theta_i\partial\theta_j}\log \boldsymbol{\pi}(\mathbf{a}|\mathbf{s};\boldsymbol{\theta})\right| \leq F \tag{3.17}$$

where θ_i and θ_j denote the ith and jth entries of $\boldsymbol{\theta}$, respectively.

Assumption 1 requires the boundedness of the instantaneous loss and thus the discounted cumulative loss, which is natural and commonly assumed in analyzing RL algorithms, e.g., [123, 180, 11]. Assumption 2 requires the score function and its partial derivatives to be bounded, which can be also satisfied by a wide range of stochastic policies, e.g., parametrized Gaussian policies [123]. As we will see next, Assumptions 1 and 2 are sufficient to guarantee the smoothness of the objective function in (3.7).

Lemma 8 (smoothness in cumulative losses)**.** *Under Assumptions 1 and 2, for any policy parameter* $\boldsymbol{\theta} \in \mathbb{R}^d$*, the accumulated loss* $\mathcal{L}_m(\boldsymbol{\theta})$ *for worker* m *is* L_m*-smooth, that is*

$$\|\nabla\mathcal{L}_m(\boldsymbol{\theta}_1)-\nabla\mathcal{L}_m(\boldsymbol{\theta}_2)\| \leq L_m\|\boldsymbol{\theta}_1-\boldsymbol{\theta}_2\| \;\; \text{with} \;\; L_m := \left(F + G^2 + \frac{2\gamma G^2}{1-\gamma}\right)\frac{\gamma\bar{\ell}_m}{(1-\gamma)^2} \tag{3.18}$$

where $\bar{\ell}_m$ *is the upper bound of the instantaneous loss in Assumption 1, and* F*,* G *are constants bounding the score function in* (3.17)*. Likewise, the overall accumulated loss* $\mathcal{L}(\boldsymbol{\theta})$ *is* L*-smooth, that is*

$$\|\nabla\mathcal{L}(\boldsymbol{\theta}_1)-\nabla\mathcal{L}(\boldsymbol{\theta}_2)\| \leq L\|\boldsymbol{\theta}_1-\boldsymbol{\theta}_2\| \;\; \text{with} \;\; L := \left(F + G^2 + \frac{2\gamma G^2}{1-\gamma}\right)\frac{\gamma\sum_{m\in\mathcal{M}}\bar{\ell}_m}{(1-\gamma)^2}. \tag{3.19}$$

The smoothness of the objective function is critical in the convergence analyses of many nonconvex optimization algorithms. Building upon Lemma 8, the subsequent analysis critically builds on the following Lyapunov function:

$$\mathbb{V}^k := \mathcal{L}(\boldsymbol{\theta}^k) - \mathcal{L}(\boldsymbol{\theta}^*) + \frac{3}{2\alpha}\sum_{d=1}^{D}\sum_{\tau=d}^{D}\xi_\tau\left\|\boldsymbol{\theta}^{k+1-d}-\boldsymbol{\theta}^{k-d}\right\|^2 \tag{3.20}$$

where $\boldsymbol{\theta}^*$ is the minimizer of (3.2), and α, $\{\xi_\tau\}$ are constants that will be determined later.

For the DRL problem in (3.7), LAPG can guarantee the following convergence result.

Theorem 4 (iteration complexity)**.** *Under Assumptions 1 and 2, if the stepsize α and the parameters $\{\xi_d\}$ in the LAPG condition* (3.16) *are chosen such that*

$$\alpha \leq \frac{\left(1 - 3\sum_{d=1}^{D} \xi_d\right)}{L} \tag{3.21}$$

and the constants T, K, and N are chosen satisfying

$$T = \mathcal{O}(\log(1/\epsilon)), \quad K = \mathcal{O}(1/\epsilon), \quad \text{and} \quad N = \mathcal{O}(\log(K/\delta)/\epsilon) \tag{3.22}$$

then with probability at least $1-\delta$, the iterates $\{\boldsymbol{\theta}^k\}$ generated by LAPG satisfy

$$\frac{1}{K}\sum_{k=1}^{K} \left\|\nabla \mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 \leq \frac{2}{\alpha K}\mathbb{V}^1 + 3\sigma_T^2 + 21\sigma_{N,\delta/K}^2 \leq \epsilon \tag{3.23}$$

where σ_T and $\sigma_{N,\delta/K}$ are some constants depending on $T, N, F, G, \gamma, \{\bar{\ell}_m\}$.

Theorem 4 demonstrates that even with the adaptive communication rules, LAPG can still achieve sublinear convergence to the stationary point of (3.7).

Regarding the communication complexity, it would be helpful to first estimate each learner's frequency of activating the communication condition (3.16). Ideally, we want those learners with a small reward thus a small smoothness constant to communicate with the controller less frequently. This intuition will be formally treated in the next lemma.

Lemma 9 (lazy gradient communication)**.** *Under Assumptions 1 and 2, define the task hardness of every learner m as $\mathbb{H}(m) := L_m^2/L^2$. If the constants $\{\xi_d\}$ in the communication condition* (3.16) *are chosen to be $\xi_D \leq \cdots \leq \xi_1$ and the hardness of the learner m satisfies*

$$\mathbb{H}(m) \leq \frac{\xi_d}{3d\alpha^2 L^2 M^2} := \gamma_d \tag{3.24}$$

then it uploads to the controller at most $1/(d+1)$ fraction of time with probability at least $1-2\delta$.

Lemma 9 implies that the communication frequency of each learner is proportional to its task hardness. In addition, choosing larger trigger constants $\{\xi_d\}$ and a smaller stepsize α will reduce the communication frequencies of all learners. However, such choice of parameters will generally require much more number of iterations to a targeted accuracy. To formally characterize the overall communication overhead of solving DRL, we define the communication complexity of solving the DRL problem (3.2) as the number of needed uploads to achieve ϵ-policy gradient error; e.g.,

$\min_{k=1,\cdots,K} \|\nabla \mathcal{L}(\boldsymbol{\theta}^k)\|^2 \leq \epsilon$.

Building upon Theorem 4 and Lemma 9, the communication complexity is established next.

Theorem 5 (communication complexity)**.** *Under Assumptions 1 and 2, define* $\Delta\mathbb{C}(h; \{\gamma_d\})$ *as*

$$\Delta\mathbb{C}(h; \{\gamma_d\}) := \sum_{d=1}^{D} \Big(\frac{1}{d} - \frac{1}{d+1}\Big) h\left(\gamma_d\right) \tag{3.25}$$

where h *is the cumulative density function of the learners' task hardness, given by*

$$h(\gamma) := \frac{1}{M} \sum_{m \in \mathcal{M}} \mathbb{1}(\mathbb{H}(m) \leq \gamma). \tag{3.26}$$

With the communication complexity of LAPG and PG denoted as $\mathbb{C}_{\mathrm{LAPG}}(\epsilon)$ *and* $\mathbb{C}_{\mathrm{PG}}(\epsilon)$*, if the parameters are chosen as* (3.22)*, with probability at least* $1 - 4\delta$*, we have that*

$$\mathbb{C}_{\mathrm{LAPG}}(\epsilon) \leq (1 - \Delta\mathbb{C}(h; \{\gamma_d\})) \frac{\mathbb{C}_{\mathrm{PG}}(\epsilon)}{(1 - 3\sum_{d=1}^{D} \xi_d)}. \tag{3.27}$$

Choosing the parameters as Theorem 4, if the heterogeneity function $h(\gamma)$ *satisfies that there exists* γ' *such that* $\gamma' < \frac{h(\gamma')}{(D+1)DM^2}$*, then we have that* $\mathbb{C}_{\mathrm{LAPG}}(\epsilon) < \mathbb{C}_{\mathrm{PG}}(\epsilon)$.

By carefully designing our communication selection rule, Theorem 5 demonstrates that the overall communication of LAPG is less than that of PG, provided that the reward functions thus the smoothness constants of each learner are very heterogeneous. Consider the extreme case where $L_m = \mathcal{O}(1)$, $\forall 1, \ldots, M-1$, and $L_M = L = \mathcal{O}(M^2)$. One can easily verify that using proper parameters, LAPG only requires $\mathcal{O}(1/M)$ number of communication rounds of vanilla PG.

While the improved communication complexity in Theorem 5 builds on slightly restrictive dependence on the problem parameters, the LAPG's empirical performance gain over PG goes far beyond the above worst-case theoretical analysis, and lies in a much broader DRL setting (e.g., $h(\cdot)$ does not satisfy the condition in Theorem 5), which is confirmed by the subsequent numerical tests.

3.5 Numerical tests

To validate the theoretical results, this section reports the empirical performance of LAPG in the multi-agent RL task, as an example of DRL. All experiments were performed using Python 3.6 on an Intel i7 CPU @ 3.4 GHz (32 GB RAM) desktop. Throughout this section, we consider

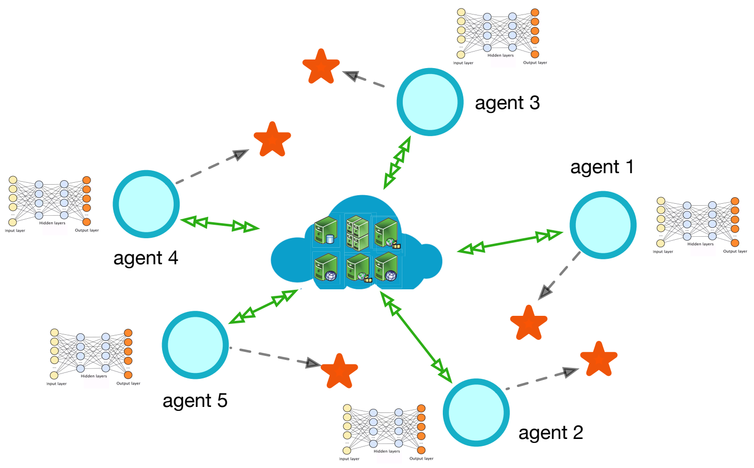

Figure 3.2: Multi-agent cooperative navigation task used in the simulation. Specifically, the blue circles represent the agents, the stars represent the landmarks, the green arrows represent the agent-cloud communication links, and the gray arrows direct the target landmark each agent aims to cover.

the simulation environment of the *Cooperative Navigation* task in [100], which builds on the popular OpenAI Gym paradigm [21]. In this RL environment, M agents aim to reach a set of M landmarks through physical movement, which is controlled by a set of five actions $\{$*stay, left, right, up, down*$\}$. Agents are connected to a remote central coordinator, and are rewarded based on the proximity of their position to the one-to-one associated landmark; see a diagram in Figure 3.2.

In the simulation, we modify the environment in [100] from following aspects: i) we assume the state is globally observable, i.e., the position and velocity of other agents in a two-dimension grid are observable to each agent; and, ii) each agent has a certain target landmark to cover, and the individual reward is determined by the proximity to that certain landmark, as well as the penalty of collision with other agents. In this way, the reward function varies among agents, and the individual reward of an agent also depends on the other agents' movement, which is consistent with the multi-agent RL formulation (3.4). The reward is further scaled by different positive coefficients, representing the heterogeneity (e.g., different priority) of different agents. The collaborative goal of the agents is to maximize the network averaged long-term reward so as to reduce distances to the landmark and avoid collisions. We implement LAPG using G(PO)MDP gradient estimators, and compare it with the G(PO)MDP-based PG method. The discounting factor in the cumulative loss is $\gamma = 0.99$ in all the tests. For each episode, both algorithms terminate after $T = 20$ iterations.

In the first test with $M = 2$ agents, the targeted local policy of each agent $\boldsymbol{\pi}_m(\boldsymbol{\theta}_m)$ is

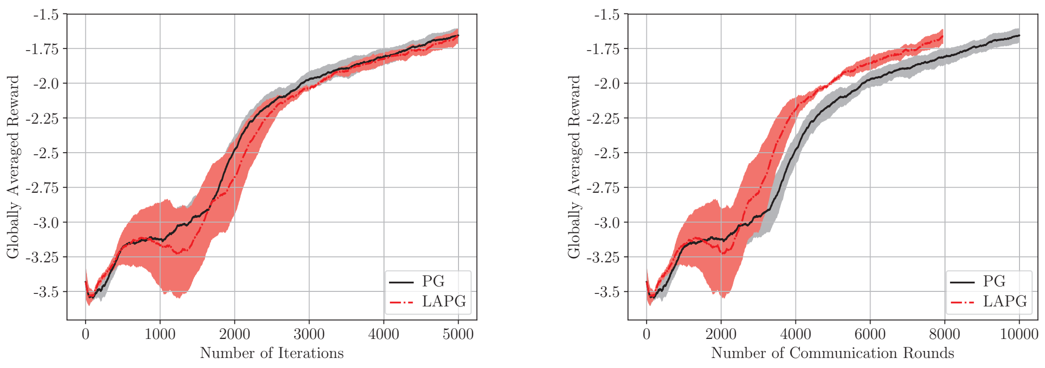

Figure 3.3: Iteration and communication complexity in a heterogeneous environment (Non momentum). The shaded region in all the figures represents the globally averaged reward distribution of each scheme within the half standard deviation of the mean.

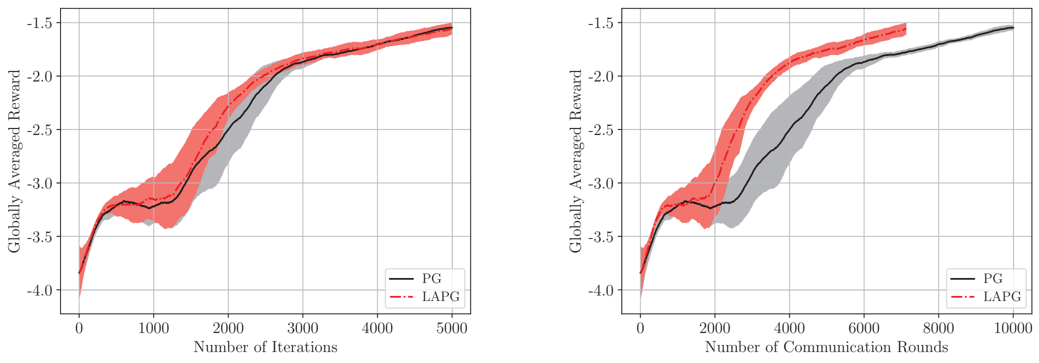

Figure 3.4: Iteration and communication complexity in a heterogeneous setting (Momentum).

parameterized by a three-layer neural network, where the first and the second hidden layers contain 30 and 10 neural units with ReLU as the activation function, and the output layer is the softmax operator. We run in total $N = 10$ batch episodes in each Monte Carlo run, and report the globally averaged reward from 10 Monte Carlo runs. LAPG and PG are first implemented using gradient descent update for the heterogeneous (scaled reward) case. As shown in Figure 3.3, LAPG converges within the same number of iterations as PG, and the communication reduction is observable. To accelerate the training of neural networks used in policy parameterization, both LAPG and PG are implemented using heavy-ball based *momentum update* thereafter, where the stepsize and the momentum factor are set as 0.01 and 0.6, respectively. The corresponding performance is reported in Figure 3.4 for the heterogeneous case and in Figure 3.5 for the homogeneous (non-scaled reward) case. Clearly, in both Figures 3.4 and 3.5, our LAPG converges

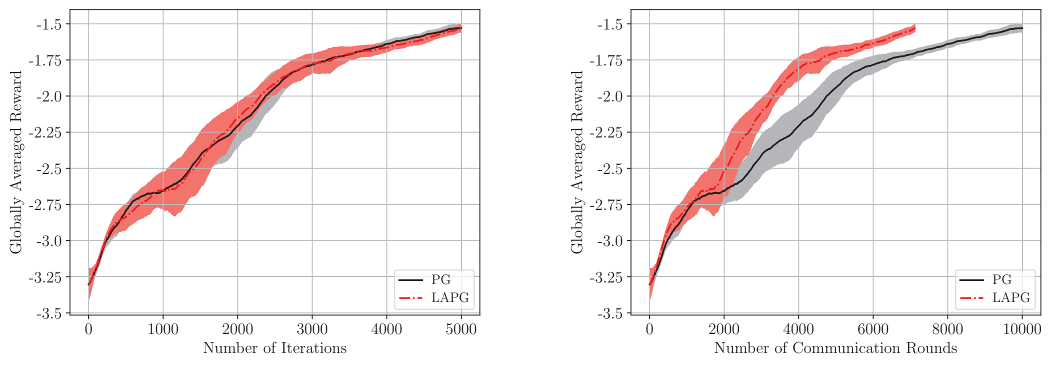

Figure 3.5: Iteration and communication complexity in a homogeneous setting (Momentum).

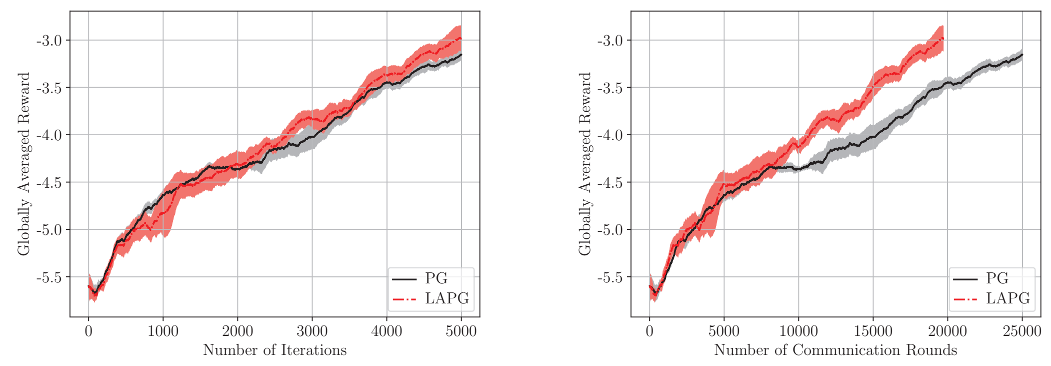

Figure 3.6: Iteration and communication complexity in a five-agent setting (RELU activation).

within the same number of iterations as the PG algorithm. When it comes to the number of communication rounds, LAPG requires significantly smaller amount than PG in both homogeneous and heterogeneous cases. With momentum update, the performance gain of LAPG is larger than that without momentum update, which is partially due to that both algorithms converge faster in this case.

In the second test with $M = 5$, the targeted policy is again parameterized by a three-layer neural network. For this larger multi-agent RL task, we use a larger network to characterize the optimal policy, where the first and the second hidden layers contain 50 and 20 neural units. To reduce the runtime, we only run in total $N = 8$ batch episodes in each Monte Carlo run, and report the globally averaged reward from 5 Monte Carlo runs in Figure 3.6 for the heterogeneous case. It is shown in Figure 3.6 that LAPG successfully converges using the same number of iterations as PG, but it requires fewer number of communication rounds than PG. The performance gain is

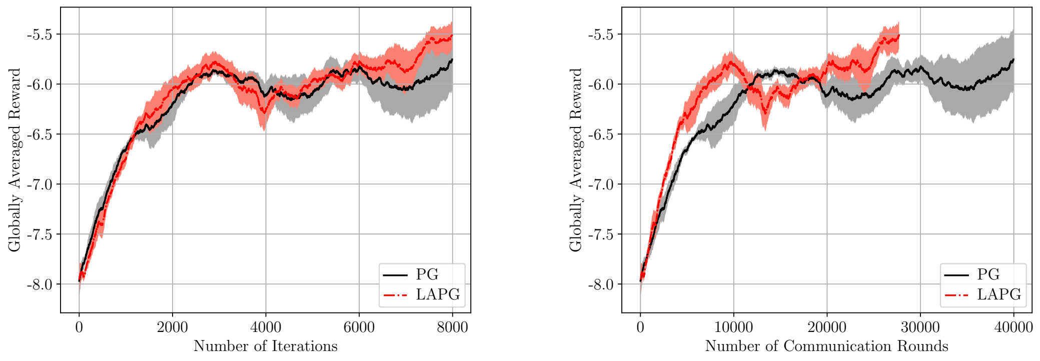

Figure 3.7: Iteration and communication complexity in a five-agent setting (Softplus).

sizable in terms of communication.

Since RELU-based activation functions may introduce certain nonsmoothness in the resultant state-to-action distribution mapping, the performance is also evaluated using the softplus activation, which is a smooth approximation of the RELU activation function. Clearly, the comparison in Figure 3.7 confirms that LAPG still converges within much fewer number of communication rounds than PG with the smooth activation functions. These observations shed the light on the potential applicability of our LAPG algorithm to large-scale DRL problems, when the communication cost of exchanging policy gradients is high especially using the over-parameterized neural networks as policy approximators.

3.6 Proofs of lemmas and theorems

3.6.1 Preliminary lemmas

In this section, we introduce several supporting lemmas that will lead to the subsequent convergence and communication complexity analysis of LAPG.

Define the finite-horizon approximation of the policy gradient (3.8) as

$$\nabla_T \mathcal{L}_m(\boldsymbol{\theta}) = \mathbb{E}_{\mathcal{T}\sim\mathbb{P}(\mathcal{T}|\boldsymbol{\theta})}\left[\sum_{t=0}^{T}\left(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau;\boldsymbol{\theta})\right)\gamma^t \ell_m(\mathbf{s}_t,\mathbf{a}_t)\right] \tag{3.28}$$

and its stochastic estimate based on a single-trajectory evaluation as

$$\hat{\nabla}_T \mathcal{L}_m(\boldsymbol{\theta}) = \sum_{t=0}^{T} \left(\sum_{\tau=0}^{t} \nabla \log \boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau; \boldsymbol{\theta}) \right) \gamma^t \ell_m(\mathbf{s}_t, \mathbf{a}_t). \tag{3.29}$$

We have the following lemma that bounds the discrepancy between them.

Lemma 10 (bounded PG deviation)**.** *For the finite-horizon approximation of the policy gradient* (3.28) *and its corresponding version* (3.29)*, at any $\boldsymbol{\theta}$ and any learner m, their discrepancy is bounded by*

$$\left\| \hat{\nabla}_T \mathcal{L}_m(\boldsymbol{\theta}) - \nabla_T \mathcal{L}_m(\boldsymbol{\theta}) \right\| \leq V_m \tag{3.30}$$

where V_m is a constant depending on $G, \gamma, \bar{\ell}_m$.

Proof: Using the definition of the G(PO)MDP gradient, we have that

$$\begin{aligned}
&\left\| \hat{\nabla}_T \mathcal{L}_m(\boldsymbol{\theta}) - \nabla_T \mathcal{L}_m(\boldsymbol{\theta}) \right\| \\
=&\left\| \sum_{t=0}^{T} \left(\sum_{\tau=0}^{t} \nabla \log \boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau; \boldsymbol{\theta}) \right) \gamma^t \ell_m(\mathbf{s}_t, \mathbf{a}_t) - \mathbb{E}_{\mathcal{T}\sim\mathbb{P}(\cdot|\boldsymbol{\theta})} \left[\sum_{t=0}^{T} \left(\sum_{\tau=0}^{t} \nabla \log \boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau; \boldsymbol{\theta}) \right) \gamma^t \ell_m(\mathbf{s}_t, \mathbf{a}_t) \right] \right\| \\
\leq& 2 \sup_{\mathcal{T}\sim\mathbb{P}(\cdot|\boldsymbol{\theta})} \sum_{t=0}^{T} \left\| \left(\sum_{\tau=0}^{t} \nabla \log \boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau; \boldsymbol{\theta}) \right) \gamma^t \ell_m(\mathbf{s}_t, \mathbf{a}_t) \right\| \\
\overset{(a)}{\leq}& 2 \sum_{t=0}^{T} t G \gamma^t \bar{\ell}_m \leq 2 G \bar{\ell}_m \sum_{t=0}^{\infty} t \gamma^t \\
=& \frac{2 G \bar{\ell}_m \gamma}{(1-\gamma)^2} := V_m
\end{aligned} \tag{3.31}$$

where (a) follows from the upper bounds in Assumptions 1 and 2, and V_m is the uniform upper bound of the G(PO)MDP stochastic policy gradient.

Lemma 11 (finite horizon approximation)**.** *For the infinite-horizon problem* (3.2) *and its finite-horizon approximation, for any $\boldsymbol{\theta}$, the corresponding policy gradients are bounded by*

$$\| \nabla \mathcal{L}(\boldsymbol{\theta}) - \nabla_T \mathcal{L}(\boldsymbol{\theta}) \| \leq \sum_{m \in \mathcal{M}} G \bar{\ell}_m \left(T + \frac{\gamma}{1-\gamma} \right) \gamma^T := \sigma_T. \tag{3.32}$$

Proof: For any $\boldsymbol{\theta} \in \mathbb{R}^d$, it follows that

$$\begin{aligned}
\|\nabla\mathcal{L}(\boldsymbol{\theta}) - \nabla_T\mathcal{L}(\boldsymbol{\theta})\| &= \left\|\mathbb{E}_{\mathcal{T}\sim\mathbb{P}(\cdot|\boldsymbol{\theta})}\left[\sum_{m\in\mathcal{M}}\sum_{t=T}^{\infty}\left(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau;\boldsymbol{\theta})\right)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\right]\right\| \\
&\overset{(a)}{\leq} \mathbb{E}_{\mathcal{T}\sim\mathbb{P}(\cdot|\boldsymbol{\theta})}\left[\left\|\sum_{m\in\mathcal{M}}\sum_{t=T}^{\infty}\left(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau;\boldsymbol{\theta})\right)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\right\|\right] \\
&\overset{(b)}{\leq} \mathbb{E}_{\mathcal{T}\sim\mathbb{P}(\cdot|\boldsymbol{\theta})}\left[\sum_{m\in\mathcal{M}}\sum_{t=T}^{\infty}\left\|\left(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau;\boldsymbol{\theta})\right)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\right\|\right] \\
&\overset{(c)}{\leq} \mathbb{E}_{\mathcal{T}\sim\mathbb{P}(\cdot|\boldsymbol{\theta})}\left[\sum_{m\in\mathcal{M}}\sum_{t=T}^{\infty}tG\gamma^t\bar{\ell}_m\right] = \mathbb{E}_{\mathcal{T}\sim\mathbb{P}(\cdot|\boldsymbol{\theta})}\left[\sum_{m\in\mathcal{M}}G\bar{\ell}_m\sum_{t=T}^{\infty}t\gamma^t\right]
\end{aligned} \tag{3.33}$$

where (a) uses the Jensen's inequality, (b) follows from the triangular inequality, and (c) uses the bounds on the loss and the score functions in Assumptions 1 and 2. We can calculate the summation as

$$\sum_{t=T}^{\infty}t\gamma^t = \left(\frac{T}{1-\gamma} + \frac{\gamma}{(1-\gamma)^2}\right)\gamma^T. \tag{3.34}$$

Plugging (3.34) into (3.33) leads to

$$\|\nabla\mathcal{L}(\boldsymbol{\theta}) - \nabla_T\mathcal{L}(\boldsymbol{\theta})\| \leq \mathbb{E}_{\mathcal{T}\sim\mathbb{P}(\cdot|\boldsymbol{\theta})}\left[\sum_{m\in\mathcal{M}}G\bar{\ell}_m\sum_{t=T}^{\infty}t\gamma^t\right] = \sum_{m\in\mathcal{M}}G\bar{\ell}_m\left(T + \frac{\gamma}{1-\gamma}\right)\frac{\gamma^T}{1-\gamma} \tag{3.35}$$

from which the proof is complete.

3.6.2 Proof of Lemma 7

The policy gradient concentration result in Lemma 7 builds on the following concentration inequality.

Lemma 12 (concentration inequality [127]). *If* $\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_N \in \mathbb{R}^d$ *denote a vector-valued martingale difference sequence satisfying* $\mathbb{E}[\mathbf{X}_n|\mathbf{X}_1, \cdots, \mathbf{X}_{n-1}] = \mathbf{0}$*, and* $\|\mathbf{X}_n\| \leq V$, $\forall n$*, then for any scalar* $\delta \in (0, 1]$*, we have*

$$\mathbb{P}\left(\left\|\sum_{n=1}^{N}\mathbf{X}_n\right\|^2 > 2\log(2/\delta)V^2N\right) \leq \delta. \tag{3.36}$$

Therefore, viewing $\mathbf{X}_n := \hat{\nabla}_T\mathcal{L}_m(\boldsymbol{\theta}) - \nabla_T\mathcal{L}_m(\boldsymbol{\theta})$, and using the bounded PG deviation in

Lemma 10, we can readily arrive at Lemma 7.

3.6.3 Proof of Lemma 8

For any $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \mathbb{R}^d$, it follows that

$$\begin{aligned}\|\nabla\mathcal{L}_m(\boldsymbol{\theta}_1) - \nabla\mathcal{L}_m(\boldsymbol{\theta}_2)\| = \Bigg\| & \mathbb{E}_{\mathcal{T}\sim\mathbb{P}(\cdot|\boldsymbol{\theta}_1)}\left[\sum_{t=0}^{\infty}\left(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau;\boldsymbol{\theta}_1)\right)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\right] \\ & -\mathbb{E}_{\mathcal{T}\sim\mathbb{P}(\cdot|\boldsymbol{\theta}_2)}\left[\sum_{t=0}^{\infty}\left(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau;\boldsymbol{\theta}_1)\right)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\right] \\ & +\mathbb{E}_{\mathcal{T}\sim\mathbb{P}(\cdot|\boldsymbol{\theta}_2)}\left[\sum_{t=0}^{\infty}\left(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau;\boldsymbol{\theta}_1)\right)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\right] \\ & -\mathbb{E}_{\mathcal{T}\sim\mathbb{P}(\cdot|\boldsymbol{\theta}_2)}\left[\sum_{t=0}^{\infty}\left(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau;\boldsymbol{\theta}_2)\right)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\right]\Bigg\|.\end{aligned} \tag{3.37}$$

We can bound the first difference term in (3.37) as (cf. use $\mathcal{T}\sim\boldsymbol{\theta}_1$ for $\mathcal{T}\sim\mathbb{P}(\cdot|\boldsymbol{\theta}_1)$)

$$\begin{aligned}&\left\|\mathbb{E}_{\mathcal{T}\sim\boldsymbol{\theta}_1}\left[\sum_{t=0}^{\infty}\left(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau;\boldsymbol{\theta}_1)\right)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\right]-\mathbb{E}_{\mathcal{T}\sim\boldsymbol{\theta}_2}\left[\sum_{t=0}^{\infty}\left(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau;\boldsymbol{\theta}_1)\right)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\right]\right\| \\ =&\left\|\int\mathbb{P}(\mathcal{T}|\boldsymbol{\theta}_1)\sum_{t=0}^{\infty}\left(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau;\boldsymbol{\theta}_1)\right)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)-\mathbb{P}(\mathcal{T}|\boldsymbol{\theta}_2)\sum_{t=0}^{\infty}\left(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau;\boldsymbol{\theta}_1)\right)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\mathbf{d}\mathcal{T}\right\| \\ =&\left\|\sum_{t=0}^{\infty}\int\mathbb{P}(\mathcal{T}_t|\boldsymbol{\theta}_1)\left(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau;\boldsymbol{\theta}_1)\right)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)-\mathbb{P}(\mathcal{T}_t|\boldsymbol{\theta}_2)\left(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau;\boldsymbol{\theta}_1)\right)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\mathbf{d}\mathcal{T}_t\right\| \\ \le&\sum_{t=0}^{\infty}\int\left\|\left(\mathbb{P}(\mathcal{T}_t|\boldsymbol{\theta}_1)-\mathbb{P}(\mathcal{T}_t|\boldsymbol{\theta}_2)\right)\left(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau;\boldsymbol{\theta}_1)\right)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\right\|\mathbf{d}\mathcal{T}_t \\ \le&\sum_{t=0}^{\infty}\int\left|\mathbb{P}(\mathcal{T}_t|\boldsymbol{\theta}_1)-\mathbb{P}(\mathcal{T}_t|\boldsymbol{\theta}_2)\right|\left\|\left(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau;\boldsymbol{\theta}_1)\right)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\right\|\mathbf{d}\mathcal{T}_t\end{aligned} \tag{3.38}$$

where we use $\mathcal{T}_t$ for a t-slot trajectory $\{\mathbf{s}_0,\mathbf{a}_0,\mathbf{s}_1,\mathbf{a}_1,\cdots,\mathbf{s}_{t-1},\mathbf{a}_{t-1}\}$.

For the remaining difference term in (3.38), we can bound it as

$$\begin{aligned}\left|\mathbb{P}(\mathcal{T}_t|\boldsymbol{\theta}_1)-\mathbb{P}(\mathcal{T}_t|\boldsymbol{\theta}_2)\right| =&\left|\rho(\mathbf{s}_0)\prod_{\nu=0}^{t-1}\boldsymbol{\pi}(\mathbf{a}_\nu|\mathbf{s}_\nu;\boldsymbol{\theta}_1)\mathbb{P}(\mathbf{s}_{\nu+1}|\mathbf{s}_\nu,\mathbf{a}_\nu)-\rho(\mathbf{s}_0)\prod_{\nu=0}^{t-1}\boldsymbol{\pi}(\mathbf{a}_\nu|\mathbf{s}_\nu;\boldsymbol{\theta}_2)\mathbb{P}(\mathbf{s}_{\nu+1}|\mathbf{s}_\nu,\mathbf{a}_\nu)\right| \\ =&\rho(\mathbf{s}_0)\prod_{\nu=0}^{t-1}\mathbb{P}(\mathbf{s}_{\nu+1}|\mathbf{s}_\nu,\mathbf{a}_\nu)\left|\prod_{\nu=0}^{t-1}\boldsymbol{\pi}(\mathbf{a}_\nu|\mathbf{s}_\nu;\boldsymbol{\theta}_1)-\prod_{\nu=0}^{t-1}\boldsymbol{\pi}(\mathbf{a}_\nu|\mathbf{s}_\nu;\boldsymbol{\theta}_2)\right|\end{aligned}$$

$$
\begin{aligned}
&=\rho(\mathbf{s}_0)\prod_{\nu=0}^{t-1}\mathbb{P}(\mathbf{s}_{\nu+1}|\mathbf{s}_\nu,\mathbf{a}_\nu)\left|\prod_{t=0}^{t-1}\boldsymbol{\pi}(\mathbf{a}_\nu|\mathbf{s}_\nu;\boldsymbol{\theta}_1)-\prod_{\nu=0}^{t-1}\boldsymbol{\pi}(\mathbf{a}_\nu|\mathbf{s}_\nu;\boldsymbol{\theta}_2)\right| \\
&=\rho(\mathbf{s}_0)\prod_{\nu=0}^{t-1}\mathbb{P}(\mathbf{s}_{\nu+1}|\mathbf{s}_\nu,\mathbf{a}_\nu)\left|(\boldsymbol{\theta}_1-\boldsymbol{\theta}_2)^\top\nabla\prod_{\nu=0}^{t-1}\boldsymbol{\pi}(\mathbf{a}_\nu|\mathbf{s}_\nu;\tilde{\boldsymbol{\theta}})\right|. \qquad (3.39)
\end{aligned}
$$

Note that we have

$$
\begin{aligned}
\left|(\boldsymbol{\theta}_1-\boldsymbol{\theta}_2)^\top\nabla\prod_{\nu=0}^{t-1}\boldsymbol{\pi}(\mathbf{a}_\nu|\mathbf{s}_\nu;\tilde{\boldsymbol{\theta}})\right| &= \left|(\boldsymbol{\theta}_1-\boldsymbol{\theta}_2)^\top\nabla\log\prod_{\nu=0}^{t-1}\boldsymbol{\pi}(\mathbf{a}_\nu|\mathbf{s}_\nu;\tilde{\boldsymbol{\theta}})\prod_{\nu=0}^{t-1}\boldsymbol{\pi}(\mathbf{a}_\nu|\mathbf{s}_\nu;\tilde{\boldsymbol{\theta}})\right| \\
&= \prod_{\nu=0}^{t-1}\boldsymbol{\pi}(\mathbf{a}_\nu|\mathbf{s}_\nu;\tilde{\boldsymbol{\theta}})\left|(\boldsymbol{\theta}_1-\boldsymbol{\theta}_2)^\top\nabla\log\prod_{\nu=0}^{t-1}\boldsymbol{\pi}(\mathbf{a}_\nu|\mathbf{s}_\nu;\tilde{\boldsymbol{\theta}})\right| \\
&\le \prod_{\nu=0}^{t-1}\boldsymbol{\pi}(\mathbf{a}_\nu|\mathbf{s}_\nu;\tilde{\boldsymbol{\theta}})\left\|\sum_{\nu=0}^{t-1}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\nu|\mathbf{s}_\nu;\tilde{\boldsymbol{\theta}})\right\|\left\|\boldsymbol{\theta}_1-\boldsymbol{\theta}_2\right\| \\
&\le tG\left\|\boldsymbol{\theta}_1-\boldsymbol{\theta}_2\right\|\prod_{\nu=0}^{t-1}\boldsymbol{\pi}(\mathbf{a}_\nu|\mathbf{s}_\nu;\tilde{\boldsymbol{\theta}}) \qquad (3.40)
\end{aligned}
$$

and if plugging (3.39) and (3.40) into (3.38), it follows that

$$
\begin{aligned}
&\sum_{t=0}^{\infty}\int\left|\mathbb{P}(\mathcal{T}_t|\boldsymbol{\theta}_1)-\mathbb{P}(\mathcal{T}_t|\boldsymbol{\theta}_2)\right|\left\|\left(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau;\boldsymbol{\theta}_1)\right)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\right\|\mathbf{d}\mathcal{T}_t \\
\le&\sum_{t=0}^{\infty}\int\rho(\mathbf{s}_0)\prod_{\nu=0}^{t-1}\mathbb{P}(\mathbf{s}_{\nu+1}|\mathbf{s}_\nu,\mathbf{a}_\nu)\boldsymbol{\pi}(\mathbf{a}_\nu|\mathbf{s}_\nu;\tilde{\boldsymbol{\theta}})tG\left\|\boldsymbol{\theta}_1-\boldsymbol{\theta}_2\right\|\left\|\left(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_\tau|\mathbf{s}_\tau;\boldsymbol{\theta}_1)\right)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\right\|\mathbf{d}\mathcal{T}_t \\
\le&\sum_{t=0}^{\infty}\int\mathbb{P}(\mathcal{T}_t|\tilde{\boldsymbol{\theta}})tG\left\|\boldsymbol{\theta}_1-\boldsymbol{\theta}_2\right\|tG\gamma^t\bar{\ell}_m\mathbf{d}\mathcal{T}_t=\sum_{t=0}^{\infty}t^2G^2\gamma^t\bar{\ell}_m\left\|\boldsymbol{\theta}_1-\boldsymbol{\theta}_2\right\| \\
=&\left(\frac{\gamma}{(1-\gamma)^2}+\frac{2\gamma^2}{(1-\gamma)^3}\right)G^2\bar{\ell}_m\left\|\boldsymbol{\theta}_1-\boldsymbol{\theta}_2\right\| \qquad (3.41)
\end{aligned}
$$

where we use the equation that $\sum_{t=0}^{\infty}t^2\gamma^t=\frac{\gamma}{(1-\gamma)^2}+\frac{2\gamma^2}{(1-\gamma)^3}$.

We can separably bound the second difference term in (3.37) as

$$
\begin{aligned}
&\left\|\mathbb{E}_{\mathcal{T}\sim\boldsymbol{\theta}_2}\left[\sum_{t=0}^{\infty}\left(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_t|\mathbf{s}_t;\boldsymbol{\theta}_1)\right)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\right]-\mathbb{E}_{\mathcal{T}\sim\boldsymbol{\theta}_2}\left[\sum_{t=0}^{\infty}\left(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_t|\mathbf{s}_t;\boldsymbol{\theta}_2)\right)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\right]\right\| \\
\le&\int\mathbb{P}(\mathcal{T}|\boldsymbol{\theta}_2)\left\|\sum_{t=0}^{\infty}\left(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_t|\mathbf{s}_t;\boldsymbol{\theta}_1)\right)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)-\sum_{t=0}^{\infty}\left(\sum_{\tau=0}^{t}\nabla\log\boldsymbol{\pi}(\mathbf{a}_t|\mathbf{s}_t;\boldsymbol{\theta}_2)\right)\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\right\|\mathbf{d}\mathcal{T} \\
\le&\int\mathbb{P}(\mathcal{T}|\boldsymbol{\theta}_2)\sum_{t=0}^{\infty}\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\sum_{\tau=0}^{t}\left\|\nabla\log\boldsymbol{\pi}(\mathbf{a}_t|\mathbf{s}_t;\boldsymbol{\theta}_1)-\nabla\log\boldsymbol{\pi}(\mathbf{a}_t|\mathbf{s}_t;\boldsymbol{\theta}_2)\right\|\mathbf{d}\mathcal{T}
\end{aligned}
$$

$$\le \int \mathbb{P}(\mathcal{T}|\boldsymbol{\theta}_2)\sum_{t=0}^{\infty}\gamma^t\bar{\ell}_m\sum_{\tau=0}^{t}F\big\|\boldsymbol{\theta}_1-\boldsymbol{\theta}_2\big\|\mathrm{d}\mathcal{T} \le \sum_{t=0}^{\infty}\gamma^t\bar{\ell}_m tF\big\|\boldsymbol{\theta}_1-\boldsymbol{\theta}_2\big\| = \frac{F\bar{\ell}_m\gamma}{(1-\gamma)^2}\big\|\boldsymbol{\theta}_1-\boldsymbol{\theta}_2\big\|. \tag{3.42}$$

Combining (3.41) and (3.42), we have that

$$\begin{aligned}\|\nabla\mathcal{L}_m(\boldsymbol{\theta}_1)-\nabla\mathcal{L}_m(\boldsymbol{\theta}_2)\| &\le \left(\frac{F}{(1-\gamma)^2}+\left(\frac{1}{(1-\gamma)^2}+\frac{2\gamma}{(1-\gamma)^3}\right)G^2\right)\gamma\bar{\ell}_m\big\|\boldsymbol{\theta}_1-\boldsymbol{\theta}_2\big\| \\ &:= L_m\big\|\boldsymbol{\theta}_1-\boldsymbol{\theta}_2\big\|. \end{aligned}\tag{3.43}$$

Similarly, we can bound the Lipschitz constant of $\nabla\mathcal{L}(\boldsymbol{\theta})$, $\nabla_T\mathcal{L}(\boldsymbol{\theta})$, $\nabla_T\mathcal{L}_m(\boldsymbol{\theta})$, and the proof is complete.

3.6.4 Proof of Lemma 6

Using the smoothness of $\mathcal{L}_m$ thus $\mathcal{L}$ in Lemma 8, we have that

$$\mathcal{L}(\boldsymbol{\theta}^{k+1})-\mathcal{L}(\boldsymbol{\theta}^k) \le \left\langle\nabla\mathcal{L}(\boldsymbol{\theta}^k),\boldsymbol{\theta}^{k+1}-\boldsymbol{\theta}^k\right\rangle+\frac{L}{2}\left\|\boldsymbol{\theta}^{k+1}-\boldsymbol{\theta}^k\right\|^2. \tag{3.44}$$

Note that (3.13) can be also written as (cf. $\hat{\nabla}_{N,T}\mathcal{L}\big(\boldsymbol{\theta}^k\big) := \sum_{m\in\mathcal{M}}\hat{\nabla}_{N,T}\mathcal{L}_m\big(\boldsymbol{\theta}^k\big)$)

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k-\alpha\sum_{m\in\mathcal{M}}\hat{\nabla}_{N,T}\mathcal{L}_m\big(\boldsymbol{\theta}^k\big)-\alpha\sum_{m\in\mathcal{M}_c^k}\Big(\hat{\nabla}_{N,T}\mathcal{L}_m\big(\hat{\boldsymbol{\theta}}_m^{k-1}\big)-\hat{\nabla}_{N,T}\mathcal{L}_m\big(\boldsymbol{\theta}^k\big)\Big) \tag{3.45}$$

$$= \boldsymbol{\theta}^k-\alpha\hat{\nabla}_{N,T}\mathcal{L}\big(\boldsymbol{\theta}^k\big)+\alpha\sum_{m\in\mathcal{M}_c^k}\delta\hat{\nabla}_m^k \tag{3.46}$$

where $\mathcal{M}_c^k$ is the set of agents that *do not* communicate with the controller at iteration k.

Plugging (3.45) into $\big\langle\nabla\mathcal{L}(\boldsymbol{\theta}^k),\boldsymbol{\theta}^{k+1}-\boldsymbol{\theta}^k\big\rangle$ leads to (cf. $\hat{\boldsymbol{\theta}}_m^k=\hat{\boldsymbol{\theta}}_m^{k-1}$, $\forall m\in\mathcal{M}_c^k$)

$$\begin{aligned}\left\langle\nabla\mathcal{L}(\boldsymbol{\theta}^k),\boldsymbol{\theta}^{k+1}-\boldsymbol{\theta}^k\right\rangle =& -\alpha\left\langle\nabla\mathcal{L}(\boldsymbol{\theta}^k),\hat{\nabla}_{N,T}\mathcal{L}\big(\boldsymbol{\theta}^k\big)-\sum_{m\in\mathcal{M}_c^k}\delta\hat{\nabla}_m^k\right\rangle \\ =& -\alpha\left\langle\nabla\mathcal{L}(\boldsymbol{\theta}^k),\nabla\mathcal{L}(\boldsymbol{\theta}^k)-\nabla\mathcal{L}(\boldsymbol{\theta}^k)+\hat{\nabla}_{N,T}\mathcal{L}\big(\boldsymbol{\theta}^k\big)-\sum_{m\in\mathcal{M}_c^k}\delta\hat{\nabla}_m^k\right\rangle \\ =& -\alpha\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2-\alpha\left\langle\nabla\mathcal{L}(\boldsymbol{\theta}^k),\hat{\nabla}_{N,T}\mathcal{L}\big(\boldsymbol{\theta}^k\big)-\nabla\mathcal{L}(\boldsymbol{\theta}^k)-\sum_{m\in\mathcal{M}_c^k}\delta\hat{\nabla}_m^k\right\rangle. \end{aligned}\tag{3.47}$$

Using $2\mathbf{a}^\top\mathbf{b} = \|\mathbf{a}\|^2 + \|\mathbf{b}\|^2 - \|\mathbf{a}-\mathbf{b}\|^2$, we can re-write the inner product in (3.47) as

$$\begin{aligned}
&\left\langle -\nabla\mathcal{L}(\boldsymbol{\theta}^k), \hat{\nabla}_{N,T}\mathcal{L}\big(\boldsymbol{\theta}^k\big) - \nabla\mathcal{L}(\boldsymbol{\theta}^k) - \sum_{m\in\mathcal{M}_c^k}\delta\hat{\nabla}_m^k \right\rangle\\
=&\frac{1}{2}\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + \frac{1}{2}\left\|\hat{\nabla}_{N,T}\mathcal{L}\big(\boldsymbol{\theta}^k\big) - \nabla\mathcal{L}(\boldsymbol{\theta}^k) - \sum_{m\in\mathcal{M}_c^k}\delta\hat{\nabla}_m^k\right\|^2 - \frac{1}{2}\left\|\hat{\nabla}_{N,T}\mathcal{L}\big(\boldsymbol{\theta}^k\big) - \sum_{m\in\mathcal{M}_c^k}\delta\hat{\nabla}_m^k\right\|^2\\
\overset{(a)}{=}&\frac{1}{2}\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + \frac{1}{2}\left\|\hat{\nabla}_{N,T}\mathcal{L}\big(\boldsymbol{\theta}^k\big) - \nabla\mathcal{L}(\boldsymbol{\theta}^k) - \sum_{m\in\mathcal{M}_c^k}\delta\hat{\nabla}_m^k\right\|^2 - \frac{1}{2\alpha^2}\left\|\boldsymbol{\theta}^{k+1}-\boldsymbol{\theta}^k\right\|^2 \quad (3.48)
\end{aligned}$$

where (a) follows from the LAPG update (3.45).

Define the policy gradient for the finite-horizon discounted reward as

$$\nabla_T\mathcal{L}(\boldsymbol{\theta}) := \sum_{m\in\mathcal{M}}\nabla_T\mathcal{L}_m(\boldsymbol{\theta}) \;\text{ with }\; \nabla_T\mathcal{L}_m(\boldsymbol{\theta}) = \mathbb{E}_{\mathcal{T}\sim\mathbb{P}(\cdot|\boldsymbol{\theta})}\left[\nabla\log\mathbb{P}(\mathcal{T}|\boldsymbol{\theta})\left(\sum_{t=0}^{T}\gamma^t\ell_m(\mathbf{s}_t,\mathbf{a}_t)\right)\right] \quad (3.49)$$

and decompose the second term in (3.48) as

$$\begin{aligned}
&\left\|\hat{\nabla}_{N,T}\mathcal{L}\big(\boldsymbol{\theta}^k\big) - \nabla\mathcal{L}(\boldsymbol{\theta}^k) - \sum_{m\in\mathcal{M}_c^k}\delta\hat{\nabla}_m^k\right\|^2\\
=&\left\|\hat{\nabla}_{N,T}\mathcal{L}\big(\boldsymbol{\theta}^k\big) - \nabla_T\mathcal{L}(\boldsymbol{\theta}^k) + \nabla_T\mathcal{L}(\boldsymbol{\theta}^k) - \nabla\mathcal{L}(\boldsymbol{\theta}^k) - \sum_{m\in\mathcal{M}_c^k}\delta\hat{\nabla}_m^k\right\|^2\\
\overset{(b)}{=}&3\left\|\hat{\nabla}_{N,T}\mathcal{L}\big(\boldsymbol{\theta}^k\big) - \nabla_T\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + 3\left\|\nabla_T\mathcal{L}(\boldsymbol{\theta}^k) - \nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + 3\left\|\sum_{m\in\mathcal{M}_c^k}\delta\hat{\nabla}_m^k\right\|^2 \quad (3.50)
\end{aligned}$$

where (b) follows from the inequality $\|\mathbf{a}+\mathbf{b}+\mathbf{c}\|^2 \leq 3\|\mathbf{a}\|^2 + 3\|\mathbf{b}\|^2 + 3\|\mathbf{c}\|^2$. Combining (3.47), (3.48) and (3.50), and plugging into (3.44), the claim of Lemma 6 follows.

3.6.5 Proof of Theorem 4

Using the definition of $\mathbb{V}^k$ in (3.20), it follows that (with the short-hand notation $\beta_d := \frac{3}{2\alpha}\sum_{\tau=d}^{D}\xi_\tau$)

$$\begin{aligned}
\mathbb{V}^{k+1} - \mathbb{V}^k =&\mathcal{L}(\boldsymbol{\theta}^{k+1}) - \mathcal{L}(\boldsymbol{\theta}^k) + \sum_{d=1}^{D}\beta_d\left\|\boldsymbol{\theta}^{k+2-d} - \boldsymbol{\theta}^{k+1-d}\right\|^2 - \sum_{d=1}^{D}\beta_d\left\|\boldsymbol{\theta}^{k+1-d} - \boldsymbol{\theta}^{k-d}\right\|^2\\
&\overset{(a)}{\leq} -\frac{\alpha}{2}\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + \frac{3\alpha}{2}\left\|\sum_{m\in\mathcal{M}_c^k}\delta\hat{\nabla}_m^k\right\|^2 + \sum_{d=2}^{D}\beta_d\left\|\boldsymbol{\theta}^{k+2-d} - \boldsymbol{\theta}^{k+1-d}\right\|^2 + \frac{3\alpha}{2}\left\|\hat{\nabla}_{N,T}\mathcal{L}\big(\boldsymbol{\theta}^k\big) - \nabla_T\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2
\end{aligned}$$

$$+\left(\frac{L}{2}-\frac{1}{2\alpha}+\beta_1\right)\left\|\boldsymbol{\theta}^{k+1}-\boldsymbol{\theta}^{k}\right\|^2-\sum_{d=1}^{D}\beta_d\left\|\boldsymbol{\theta}^{k+1-d}-\boldsymbol{\theta}^{k-d}\right\|^2+\frac{3\alpha}{2}\left\|\nabla_T\mathcal{L}(\boldsymbol{\theta}^k)-\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 \tag{3.51}$$

where (a) uses (3.14) in Lemma 6.

Using $(\sum_{n=1}^{N} a_n)^2 \leq N\sum_{n=1}^{N} a_n^2$, it follows that

$$\left\|\sum_{m\in\mathcal{M}_c^k}\delta\hat{\nabla}_m^k\right\|^2=\left\|\sum_{m\in\mathcal{M}_c^k}\nabla\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^k)-\nabla\mathcal{L}_m(\boldsymbol{\theta}^k)\right\|^2 \tag{3.52a}$$

$$\leq\left|\mathcal{M}_c^k\right|\sum_{m\in\mathcal{M}_c^k}\left\|\nabla\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^k)-\nabla\mathcal{L}_m(\boldsymbol{\theta}^k)\right\|^2 \tag{3.52b}$$

$$\overset{(b)}{\leq}\frac{|\mathcal{M}_c^k|^2}{\alpha^2M^2}\sum_{d=1}^{D}\xi_d\left\|\boldsymbol{\theta}^{k+1-d}-\boldsymbol{\theta}^{k-d}\right\|^2+6\sigma_{N,\delta/K}^2 \tag{3.52c}$$

where (b) uses the communication trigger condition (3.16), and the fact that $\sigma_{N,\delta/K}^2=M\sum_{m\in\mathcal{M}}\sigma_{m,N,\delta/K}^2$.

Plugging (3.52) into (3.51), we have (for convenience, define $\beta_{D+1}=0$ in the analysis)

$$\begin{aligned}&\mathbb{V}^{k+1}-\mathbb{V}^{k}\\&\leq-\frac{\alpha}{2}\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2+\sum_{d=1}^{D}\left(\frac{3\xi_d\left|\mathcal{M}_c^k\right|^2}{2\alpha M^2}-\beta_d+\beta_{d+1}\right)\left\|\boldsymbol{\theta}^{k+1-d}-\boldsymbol{\theta}^{k-d}\right\|^2+9\alpha\sigma_{N,\delta/K}^2\\&+\left(\frac{L}{2}-\frac{1}{2\alpha}+\beta_1\right)\left\|\boldsymbol{\theta}^{k+1}-\boldsymbol{\theta}^{k}\right\|^2+\frac{3\alpha}{2}\left\|\nabla_T\mathcal{L}(\boldsymbol{\theta}^k)-\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2+\frac{3\alpha}{2}\left\|\hat{\nabla}_{N,T}\mathcal{L}(\boldsymbol{\theta}^k)-\nabla_T\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2.\end{aligned} \tag{3.53}$$

After defining some constants to simplify the notation, the proof is then complete.

Furthermore, using $\beta_d:=\frac{3}{2\alpha}\sum_{\tau=d}^{D}\xi_\tau$, if the stepsize α, and the trigger constants $\{\xi_d\}$ satisfy

$$\alpha\leq\frac{(1-3\sum_{d=1}^{D}\xi_d)}{L} \tag{3.54}$$

then it is easy to verify that all the parentheses of (3.53) are all nonpositive. Hence, we have that the descent in Lyapunov function is bounded as

$$\begin{aligned}&\mathbb{V}^{k+1}-\mathbb{V}^{k}\\&\leq-\frac{\alpha}{2}\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2+\frac{3\alpha}{2}\left\|\nabla_T\mathcal{L}(\boldsymbol{\theta}^k)-\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2+\frac{3\alpha}{2}\left\|\hat{\nabla}_{N,T}\mathcal{L}(\boldsymbol{\theta}^k)-\nabla_T\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2+9\alpha\sigma_{N,\delta/K}^2.\end{aligned} \tag{3.55}$$

Rearranging terms in (3.55), and summing up over $k = 1, \cdots, K$, we have

$$\frac{1}{K}\sum_{k=1}^{K}\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 \leq \frac{2}{\alpha K}\mathbb{V}^1 + \frac{3}{K}\sum_{k=1}^{K}\left\|\nabla_T\mathcal{L}(\boldsymbol{\theta}^k) - \nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + \frac{3}{K}\sum_{k=1}^{K}\left\|\hat{\nabla}_{N,T}\mathcal{L}(\boldsymbol{\theta}^k) - \nabla_T\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + 18\sigma^2_{N,\delta/K}$$
$$\overset{(c)}{\leq} \frac{2}{\alpha K}\mathbb{V}^1 + 3\sigma_T^2 + 21\sigma^2_{N,\delta/K}, \quad \text{w.p. } 1-\delta \tag{3.56}$$

where (c) follows from the finite-horizon truncation error in Lemma 11, and the gradient concentration result in Lemma 7 together with the union bound.

Therefore, using Lemmas 7 and 11, it readily follows that there exist $T = \mathcal{O}(\log(1/\epsilon))$, $K = \mathcal{O}(1/\epsilon)$, and $N = \mathcal{O}\left(\log(K/\delta)/\epsilon\right)$ such that

$$\frac{1}{K}\sum_{k=1}^{K}\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 \leq \frac{2}{\alpha K}\mathbb{V}^1 + 3\sigma_T^2 + 21\sigma^2_{N,\delta/K} \leq \epsilon, \quad \text{w.p. } 1-\delta \tag{3.57}$$

from which the proof is complete.

3.6.6 Proof of Lemma 9

The idea is essentially to show that if (3.24) holds, then the learner m will not violate the LAPG conditions in (3.16) so that does not upload, if it has uploaded at least once during the last d iterations.

To prove this argument, for the difference of two policy gradient evaluations, we have that

$$\left\|\hat{\nabla}_{N,T}\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^{k-1}) - \hat{\nabla}_{N,T}\mathcal{L}_m(\boldsymbol{\theta}^k)\right\|^2$$
$$= \left\|\hat{\nabla}_{N,T}\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^{k-1}) - \nabla_T\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^{k-1}) + \nabla_T\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^{k-1}) - \nabla_T\mathcal{L}_m(\boldsymbol{\theta}^k) + \nabla_T\mathcal{L}_m(\boldsymbol{\theta}^k) - \hat{\nabla}_{N,T}\mathcal{L}_m(\boldsymbol{\theta}^k)\right\|^2$$
$$\overset{(a)}{\leq} 3\left\|\hat{\nabla}_{N,T}\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^{k-1}) - \nabla_T\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^{k-1})\right\|^2 + 3\left\|\nabla_T\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^{k-1}) - \nabla_T\mathcal{L}_m(\boldsymbol{\theta}^k)\right\|^2 + 3\left\|\nabla_T\mathcal{L}_m(\boldsymbol{\theta}^k) - \hat{\nabla}_{N,T}\mathcal{L}_m(\boldsymbol{\theta}^k)\right\|^2$$
$$\overset{(b)}{\leq} 6\sigma^2_{m,N,\delta} + 3\left\|\nabla_T\mathcal{L}_m(\hat{\boldsymbol{\theta}}_m^{k-1}) - \nabla_T\mathcal{L}_m(\boldsymbol{\theta}^k)\right\|^2, \quad \text{w.p. } 1-2\delta/K$$
$$\overset{(c)}{\leq} 6\sigma^2_{m,N,\delta} + 3L_m^2\left\|\hat{\boldsymbol{\theta}}_m^{k-1} - \boldsymbol{\theta}^k\right\|^2, \quad \text{w.p. } 1-2\delta/K \tag{3.58}$$

where (a) uses $\|\mathbf{a}+\mathbf{b}+\mathbf{c}\|^2 \leq 3\|\mathbf{a}\|^2 + 3\|\mathbf{b}\|^2 + 3\|\mathbf{c}\|^2$; (b) uses Lemma 12 twice; and (c) follows from the smoothness property in Lemma 8.

Furthermore, suppose at iteration k, the most recent iteration that the learner m did communicate with the controller is iteration $k - d'$ with $1 \leq d' \leq d$. Thus, we have $\hat{\boldsymbol{\theta}}_m^{k-1} = \boldsymbol{\theta}^{k-d'}$, which implies that

$$\begin{aligned}
6\sigma_{m,N,\delta}^2 + 3L_m^2 \left\|\hat{\boldsymbol{\theta}}_m^{k-1} - \boldsymbol{\theta}^k\right\|^2 &= 6\sigma_{m,N,\delta}^2 + 3L_m^2 \left\|\boldsymbol{\theta}^{k-d'} - \boldsymbol{\theta}^k\right\|^2 \\
&= 6\sigma_{m,N,\delta}^2 + 3d'L^2\mathbb{H}(m)\sum_{b=1}^{d'}\left\|\boldsymbol{\theta}^{k+1-b} - \boldsymbol{\theta}^{k-b}\right\|^2 \\
&\overset{(d)}{\le} 6\sigma_{m,N,\delta}^2 + \frac{\xi_d}{\alpha^2 M^2}\sum_{b=1}^{d'}\left\|\boldsymbol{\theta}^{k+1-b} - \boldsymbol{\theta}^{k-b}\right\|^2 \\
&\overset{(e)}{\le} 6\sigma_{m,N,\delta}^2 + \frac{\sum_{b=1}^{D}\xi_b\left\|\boldsymbol{\theta}^{k+1-b} - \boldsymbol{\theta}^{k-b}\right\|^2}{\alpha^2 M^2}
\end{aligned} \tag{3.59}$$

where (d) follows since the condition (3.24) is satisfied, so that

$$\mathbb{H}(m) \le \frac{\xi_d}{3d\alpha^2L^2M^2} \le \frac{\xi_d}{3d'\alpha^2L^2M^2} \tag{3.60}$$

and (e) follows from our choice of $\{\xi_d\}$ such that for $1 \le d' \le d$, we have $\xi_d \le \xi_{d'} \le \ldots \le \xi_1$ and $\|\boldsymbol{\theta}^{k+1-b} - \boldsymbol{\theta}^{k-b}\|^2 \ge 0$. Since (3.59) is exactly the RHS of (3.16), the trigger condition (3.16) will not be activated, and the learner m does not communicate with the controller at iteration k.

Note that the above argument holds for any $1 \le d' \le d$, and thus if (3.24) holds, the learner m communicates with the controller at most every other d iterations. Since (3.58) holds with probability $1 - 2\delta/K$, by using union bound, this argument holds with probability $1 - 2\delta$ for all $k \in \{1, \cdots, K\}$.

3.6.7 Proof of Theorem 5

Recalling the Lyapunov function (3.20), we have

$$\mathbb{V}^k := \mathcal{L}(\boldsymbol{\theta}^k) - \mathcal{L}(\boldsymbol{\theta}^*) + \sum_{d=1}^{D}\frac{3\sum_{j=d}^{D}\xi_j}{2\alpha}\left\|\boldsymbol{\theta}^{k+1-d} - \boldsymbol{\theta}^{k-d}\right\|^2 \tag{3.61}$$

Using (3.55) in the proof of Theorem 4, and choosing the stepsize as $\alpha = \frac{1}{L}\left(1 - 3\sum_{d=1}^{D}\xi_d\right)$, we have

$$\mathbb{V}^{k+1} - \mathbb{V}^k \le -\frac{\alpha}{2}\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + \frac{3\alpha}{2}\left\|\nabla_T\mathcal{L}(\boldsymbol{\theta}^k) - \nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + \frac{3\alpha}{2}\left\|\hat{\nabla}_{N,T}\mathcal{L}(\boldsymbol{\theta}^k) - \nabla_T\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + 9\alpha\sigma_{N,\delta/K}^2. \tag{3.62}$$

Summing up both sides from $k = 1, \ldots, K$, and initializing $\boldsymbol{\theta}^{1-D} = \cdots = \boldsymbol{\theta}^0 = \boldsymbol{\theta}^1$, we have

$$\begin{aligned}
\frac{1}{K}\sum_{k=1}^{K}\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 &\leq \frac{2L\left[\mathcal{L}(\boldsymbol{\theta}^1)-\mathcal{L}(\boldsymbol{\theta}^*)\right]}{(1-3\sum_{d=1}^{D}\xi_d)K} + 3\left\|\nabla_T\mathcal{L}(\boldsymbol{\theta}^k)-\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + 3\left\|\hat{\nabla}_{N,T}\mathcal{L}(\boldsymbol{\theta}^k)-\nabla_T\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + 18\sigma_{N,\delta/K}^2 \\
&\leq \frac{2L\left[\mathcal{L}(\boldsymbol{\theta}^1)-\mathcal{L}(\boldsymbol{\theta}^*)\right]}{(1-3\sum_{d=1}^{D}\xi_d)K} + 3\sigma_T^2 + 21\sigma_{N,\delta/K}^2, \quad \text{w.p. } 1-\delta
\end{aligned} \tag{3.63}$$

With regard to PG, following the standard analysis, it can guarantee that

$$\begin{aligned}
\frac{1}{K}\sum_{k=1}^{K}\left\|\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 &\leq \frac{2L}{K}\left[\mathcal{L}(\boldsymbol{\theta}^1)-\mathcal{L}(\boldsymbol{\theta}^*)\right] + 3\left\|\nabla_T\mathcal{L}(\boldsymbol{\theta}^k)-\nabla\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 + 3\left\|\hat{\nabla}_{N,T}\mathcal{L}(\boldsymbol{\theta}^k)-\nabla_T\mathcal{L}(\boldsymbol{\theta}^k)\right\|^2 \\
&\leq \frac{2L}{K}\left[\mathcal{L}(\boldsymbol{\theta}^1)-\mathcal{L}(\boldsymbol{\theta}^*)\right] + 3\sigma_T^2 + 3\sigma_{N,\delta/K}^2, \quad \text{w.p. } 1-\delta
\end{aligned} \tag{3.64}$$

If the parameters T and N are chose large enough (cf. (3.22)), so the first term in the RHS of (3.63) and (3.64) dominates the rest two error terms. Therefore, to achieve the same ϵ-gradient error, with probability $1-2\delta$, the number of needed iterations under LAPG is $(1-3\sum_{d=1}^{D}\xi_d)^{-1}$ times that of PG. Similar to the derivations in [31, Proposition 1], using Lemma 9, we can show that the LAPG's average communication rounds per iteration is $(1-\Delta\bar{\mathbb{C}}(h;\{\gamma_d\}))$ times that of PG with probability $1-2\delta$, we arrive at (3.27) with probability $1-4\delta$.

If we choose the parameters as

$$\xi_1 = \xi_2 = \ldots = \xi_D = \xi \quad \text{and} \quad \alpha = \frac{1-3D\xi}{L} \quad \text{and} \quad \gamma_d = \frac{\xi/d}{3\alpha^2L^2M^2},\ d = 1, \ldots, D. \tag{3.65}$$

As $h(\cdot)$ is non-decreasing, if $\gamma_D \geq \gamma'$, it readily follows that $h(\gamma_D) \geq h(\gamma')$. Together with the definition of $\Delta\bar{\mathbb{C}}(h;\{\gamma_d\})$ in (3.25), we arrive at

$$\Delta\bar{\mathbb{C}}(h;\{\gamma_d\}) = \sum_{d=1}^{D}\left(\frac{1}{d}-\frac{1}{d+1}\right)h(\gamma_d) \geq \sum_{d=1}^{D}\left(\frac{1}{d}-\frac{1}{d+1}\right)h(\gamma_D) \geq \frac{D}{D+1}h(\gamma'). \tag{3.66}$$

Therefore, the total communication are reduced if the following relation is satisfied

$$\frac{\mathbb{C}_{\mathrm{LAPG}}(\epsilon)}{\mathbb{C}_{\mathrm{PG}}(\epsilon)} = \left(1-\frac{D}{D+1}h(\gamma')\right)\cdot\frac{1}{1-3D\xi} < 1 \tag{3.67}$$

which holds if we have $h(\gamma') > 3(D+1)\xi$. On the other hand, the condition $\gamma_D \geq \gamma'$ requires

$$\xi/D \geq \gamma'(1-D\xi)^2M^2. \tag{3.68}$$

Clearly, if $\xi > \gamma' DM^2$, then (3.68) holds. In all, if we have

$$\gamma' < \frac{\xi}{DM^2} < \frac{h(\gamma')}{3(D+1)DM^2} \tag{3.69}$$

then $\mathbb{C}_{\mathrm{LAPG}}(\epsilon) \leq \mathbb{C}_{\mathrm{PG}}(\epsilon)$ in Theorem 5 holds with probability $1 - 4\delta$.

Chapter 4

Statistical learning viewpoint of network resource management

4.1 Introduction

In the era of big data analytics, cloud computing and Internet of Things, the growing demand for massive data processing challenges existing resource allocation approaches. Huge volumes of data acquired by distributed sensors in the presence of operational uncertainties caused by, e.g., renewable energy, call for scalable and adaptive network control schemes. Scalability of a desired approach refers to low complexity and amenability to distributed implementation, while adaptivity implies capability of online adjustment to dynamic environments.

4.1.1 Related work

Allocation of network resources can be traced back to the seminal work of [162]. Since then, popular allocation algorithms operating in the dual domain are first-order methods based on dual gradient ascent, either deterministic [99] or stochastic [53, 116]. Thanks to their simple computation and implementation, these approaches have attracted a great deal of recent interest, and have been successfully applied to cloud, transportation and power grid networks; see, e.g., [28, 35, 58, 155]. However, their major limitation is *slow convergence*, which results in high *network delay*. Depending on the application domain, the delay can be viewed as workload queuing time in a cloud network, traffic congestion in a transportation network, or energy level of batteries in a power network. To address this delay issue, recent attempts aim at accelerating first- and second-order optimization algorithms [12, 97, 170, 178]. Specifically, momentum-based accelerations over first-order methods were investigated using Nesterov [12], or, heavy-ball iterations [97]. Though these approaches work well in static settings, their performance degrades with online scheduling, as evidenced by the increase in accumulated steady-state error [175]. On the other hand, second-order methods such as the decentralized quasi-Newton approach and its dynamic variant developed in [170] and [178], incur high overhead to compute and communicate the decentralized Hessian approximations.

Capturing prices of resources, Lagrange multipliers play a central role in stochastic resource allocation algorithms [70]. Given abundant historical data in an online optimization setting, a natural question arises: *Is it possible to learn the optimal prices from past data, so as to improve the performance of online resource allocation strategies?* The rationale here is that past data contain statistics of network states, and learning from them can aid coping with the stochasticity of future resource allocation. A recent work in this direction is [69], which considers resource allocation with a *finite* number of possible network states and allocation actions. The learning procedure, however, involves constructing a histogram to estimate the underlying distribution of the network states, and explicitly solves an empirical dual problem. While constructing a histogram is feasible for a probability distribution with finite support, quantization errors and prohibitively high complexity are inevitable for a continuous distribution with infinite support.

4.1.2 Our contributions

In this context, the present paper aims to design a novel online resource allocation algorithm that leverages online learning from historical data for stochastic optimization of the ensuing allocation stage. The resultant approach, which we term “learn-and-adapt” stochastic dual gradient (LA-SDG) method, only doubles computational complexity of the classic stochastic dual gradient (SDG) method. With this minimal cost, LA-SDG mitigates steady-state oscillation, which is common in stochastic first-order acceleration methods [175, 97], while avoiding computation of the Hessian approximations present in the second-order methods [170, 178]. Specifically, LA-SDG only requires one more past sample to compute an extra stochastic dual gradient, in contrast to constructing costly histograms and solving the resultant large-scale problem [69].

The main contributions of this paper are summarized next.

c1) Targeting a low-complexity online solution, LA-SDG only takes an additional dual gradient step relative to the classic SDG iteration. This step enables adapting the resource allocation strategy through learning from historical data. Meanwhile, LA-SDG is linked with the stochastic heavy-ball method, nicely inheriting its fast convergence in the initial stage, while reducing its steady-state oscillation.

c2) The novel LA-SDG approach, parameterized by a positive constant μ, provably yields an attractive cost-delay tradeoff $[\mu, \log^2(\mu)/\sqrt{\mu}]$, which improves upon the standard tradeoff $[\mu, 1/\mu]$ of the SDG method [116]. Numerical tests further corroborate the performance gain of LA-SDG over existing resource allocation schemes.

4.2 Network resource management

In this section, we start with a generic network model and its resource allocation task in Section 4.2.1, and then introduce a specific example of resource allocation in cloud networks in Section 4.2.2. The proposed approach is applicable to more general network resource allocation tasks such as geographical load balancing in cloud networks [28], traffic control in transportation networks [58], and energy management in power networks [155].

4.2.1 A unified resource allocation model

Consider discrete time $t \in \mathbb{N}$, and a network represented as a directed graph $\mathcal{G} = (\mathcal{I}, \mathcal{E})$ with nodes $\mathcal{I} := \{1, \ldots, I\}$ and edges $\mathcal{E} := \{1, \ldots, E\}$. Collect the workloads across edges $e = (i, j) \in \mathcal{E}$ in a resource allocation vector $\mathbf{x}_t \in \mathbb{R}^E$. The $I \times E$ node-incidence matrix is formed with the (i, e)-th entry

$$\mathbf{A}_{(i,e)} = \begin{cases} 1, & \text{if link } e \text{ enters node } i \\ -1, & \text{if link } e \text{ leaves node } i \\ 0, & \text{else.} \end{cases} \tag{4.1}$$

We assume that each row of $\mathbf{A}$ has at least one -1 entry, and each column of $\mathbf{A}$ has at most one -1 entry, meaning that each node has at least one outgoing link, and each link has at most one source node. With $\mathbf{c}_t \in \mathbb{R}_+^I$ collecting the randomly arriving workloads of all nodes per slot t, the aggregate (endogenous plus exogenous) workloads of all nodes are $\mathbf{A}\mathbf{x}_t + \mathbf{c}_t$. If the i-th entry of $\mathbf{A}\mathbf{x}_t + \mathbf{c}_t$ is positive, there is service residual queued at node i; otherwise, node i over-serves the current arrival. With a workload queue per node, the queue length vector $\mathbf{q}_t := [q_t^1, \ldots, q_t^I]^\top \in \mathbb{R}_+^I$ obeys the recursion

$$\mathbf{q}_{t+1} = [\mathbf{q}_t + \mathbf{A}\mathbf{x}_t + \mathbf{c}_t]^+, \ \forall t \tag{4.2}$$

where $\mathbf{q}_t$ can represent the amount of user requests buffered in data queues, or energy stored in batteries, and $\mathbf{c}_t$ is the corresponding exogenously arriving workloads or harvested renewable energy of all nodes per slot t. Defining $\Psi_t(\mathbf{x}_t) := \Psi(\mathbf{x}_t; \boldsymbol{\phi}_t)$ as the aggregate network cost parameterized by the random vector $\boldsymbol{\phi}_t$, the local cost per node i is $\Psi_t^i(\mathbf{x}_t) := \Psi^i(\mathbf{x}_t; \boldsymbol{\phi}_t^i)$, and $\Psi_t(\mathbf{x}_t) = \sum_{i\in\mathcal{I}} \Psi_t^i(\mathbf{x}_t)$. The model here is quite general. The duration of time slots can vary from (micro-)seconds in cloud networks, minutes in road networks, to even hours in power networks; the nodes can present the distributed front-end mapping nodes and back-end data centers in cloud networks, intersections in traffic networks, or, buses and substations in power networks; the links can model wireless/wireline channels, traffic lanes, and power transmission lines; while the resource vector $\mathbf{x}_t$ can include the size of data workloads, the number of vehicles, or the amount of energy.

Concatenating the random parameters into a random state vector $\mathbf{s}_t := [\boldsymbol{\phi}_t^\top, \mathbf{c}_t^\top]^\top$, the resource allocation task is to determine the allocation $\mathbf{x}_t$ in response to the observed (realization) $\mathbf{s}_t$ "on the fly," so as to minimize the *long-term average* network cost subject to queue stability at each node, and operation feasibility at each link. Concretely, we have

$$\Psi^* := \min_{\{\mathbf{x}_t, \forall t\}} \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\left[\Psi_t(\mathbf{x}_t)\right] \tag{4.3a}$$

$$\text{s.t.}\quad \mathbf{q}_{t+1} = [\mathbf{q}_t + \mathbf{A}\mathbf{x}_t + \mathbf{c}_t]^+,\ \forall t \tag{4.3b}$$

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\left[\mathbf{q}_t\right] < \infty \tag{4.3c}$$

$$\mathbf{x}_t \in \mathcal{X} := \{\mathbf{x} \,|\, \mathbf{0} \le \mathbf{x} \le \bar{\mathbf{x}}\},\ \forall t \tag{4.3d}$$

where Ψ^* is the optimal objective of problem (4.3), which includes also future information; $\mathbb{E}$ is taken over $\mathbf{s}_t := [\boldsymbol{\phi}_t^\top, \mathbf{c}_t^\top]^\top$ as well as possible randomness of optimization variable $\mathbf{x}_t$; constraints (4.3c) ensure queue stability[1]; and (4.3d) confines the instantaneous allocation variables to stay within a time-invariant box constraint set $\mathcal{X}$, which is specified by, e.g., link capacities, or, server/generator capacities.

The queue dynamics in (4.3b) couple the optimization variables over an infinite time horizon, which implies that the decision variable at the current slot will have effect on all the future decisions. Therefore, finding an optimal solution of (4.3) calls for dynamic programming [17], which is known to suffer from the "curse of dimensionality" and intractability in an online setting.

[1]Here we focus on the strong stability given by [116, Definition 2.7], which requires the time-average expected queue length to be finite.

In Section 4.3.1, we will circumvent this obstacle by relaxing (4.3b)-(4.3c) to limiting average constraints, and employing dual decomposition techniques.

4.2.2 Motivating setup

The geographic load balancing task in a cloud network [28, 163, 33] takes the form of (4.3) with J mapping nodes (e.g., DNS servers) indexed by $\mathcal{J} := \{1, \ldots, J\}$, K data centers indexed by $\mathcal{K} := \{J+1, \ldots, J+K\}$. To match the definition in Section 4.2.1, consider a virtual outgoing node (indexed by 0) from each data center, and let $(k, 0)$ represent this outgoing link. Define further the node set $\mathcal{I} := \mathcal{J} \bigcup \mathcal{K}$ that includes all nodes except the virtual one, and the edge set $\mathcal{E} := \{(j,k), \forall j \in \mathcal{J}, k \in \mathcal{K}\} \bigcup \{(k,0), \forall k \in \mathcal{K}\}$ that contains links connecting mapping nodes with data centers, and outgoing links from data centers.

Per slot t, each mapping node j collects the amount of user data requests c_t^j, and forwards the amount x_t^{jk} on its link to data center k constrained by the bandwidth availability. Each data center k schedules workload processing x_t^{k0} according to its resource availability. The amount x_t^{k0} can be also viewed as the resource on its virtual outgoing link $(k, 0)$. The bandwidth limit of link (j, k) is $\bar{x}^{jk}$, while the resource limit of data center k (or link $(k, 0)$) is $\bar{x}_t^{k0}$. Similar to those in Section 4.2.1, we have the optimization vector $\mathbf{x}_t := \{x_t^{ij}, \forall (i,j) \in \mathcal{E}\} \in \mathbb{R}^{|\mathcal{E}|}$, $\mathbf{c}_t := [c_t^1, \ldots, c_t^J, 0 \ldots, 0]^\top \in \mathbb{R}^{J+K}$, and $\bar{\mathbf{x}} := \{\bar{x}_t^{ij}, \forall (i,j) \in \mathcal{E}\} \in \mathbb{R}^{|\mathcal{E}|}$. With these notational conventions, we have an $|\mathcal{I}| \times |\mathcal{E}|$ node-incidence matrix $\mathbf{A}$ as in (4.1). At each mapping node and data center, undistributed or unprocessed workloads are buffered in queues obeying (4.3b) with queue length $\mathbf{q}_t \in \mathbb{R}_+^{J+K}$; see also the system diagram in Fig. 4.1.

Performance is characterized by the aggregate cost of power consumed at the data centers plus the bandwidth costs at the mapping nodes, namely

$$\Psi_t(\mathbf{x}_t) := \sum_{k \in \mathcal{K}} \underbrace{\Psi_t^k(x_t^{k0})}_{\text{power cost}} + \sum_{j \in \mathcal{J}} \sum_{k \in \mathcal{K}} \underbrace{\Psi_t^{jk}(x_t^{jk})}_{\text{bandwidth cost}} . \tag{4.4}$$

The power cost $\Psi_t^k(x_t^{k0}) := \Psi^k(x_t^{k0}; \boldsymbol{\phi}_t^k)$, parameterized by the random vector $\boldsymbol{\phi}_t^k$, captures the local marginal price, and the renewable generation at data center k during time period t. The bandwidth cost $\Psi_t^{jk}(x_t^{jk}) := \Psi^{jk}(x_t^{jk}; \boldsymbol{\phi}_t^{jk})$, parameterized by the random vector $\boldsymbol{\phi}_t^{jk}$, characterizes the heterogeneous cost of data transmission due to spatio-temporal differences. To match the unified model in Section II-A, the local cost at data center $k \in \mathcal{K}$ is its power cost $\Psi_t^k(x_t^{k0})$, and the local cost at mapping node $j \in \mathcal{J}$ becomes $\Psi_t^j(\{x_t^{jk}\}) := \sum_{k \in \mathcal{K}} \Psi_t^{jk}(x_t^{jk})$. Hence, the cost in (4.4) can be also written as $\Psi_t(\mathbf{x}_t) := \sum_{i \in \mathcal{I}} \Psi_t^i(\mathbf{x}_t)$. Aiming to minimize the time-average of

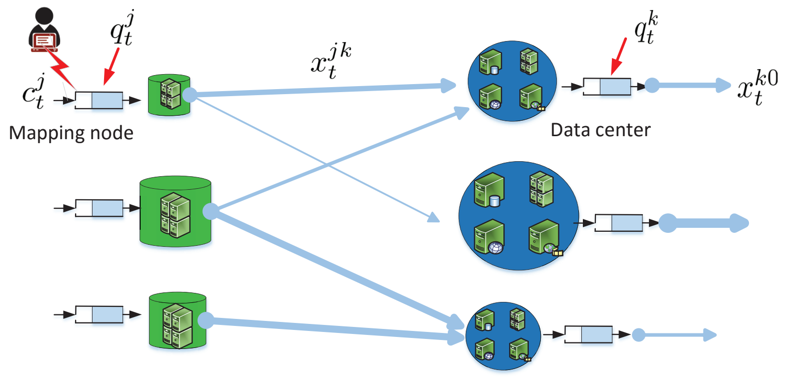

Figure 4.1: A diagram of online geographical load balancing. Per time t, mapping node j has an exogenous workload c_t^j plus that stored in the queue q_t^j, and schedules workload x_t^{jk} to data center k. Data center k serves an amount of workload x_t^{k0} out of all the assigned x_t^{jk} as well as that stored in the queue q_t^k. The thickness of each edge is proportional to its capacity.

(4.4), geographical load balancing fits the formulation in (4.3).

4.3 Online network management via SDG

In this section, the dynamic problem (4.3) is reformulated to a tractable form, and classical stochastic dual gradient (SDG) approach is revisited, along with a brief discussion of its online performance.

4.3.1 Problem reformulation

Recall in Section 4.2.1 that the main challenge of solving (4.3) resides in time-coupling constraints and unknown distribution of the underlying random processes. Regarding the first hurdle, combining (4.3b) with (4.3c), it can be shown that in the long term, workload arrival and departure rates must satisfy the following necessary condition [116, Theorem 2.8]

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\mathbf{A}\mathbf{x}_t+\mathbf{c}_t\right]\leq\mathbf{0} \tag{4.5}$$

given that the initial queue length is finite, i.e., $\|\mathbf{q}_1\|\leq\infty$. In other words, on average all buffered delay-tolerant workloads should be served. Using (4.5), a relaxed version of (4.3) is

$$\tilde{\Psi}^*:=\min_{\{\mathbf{x}_t,\forall t\}}\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\Psi_t(\mathbf{x}_t)\right]\quad\text{s.t. (4.3d) and (4.5)} \tag{4.6}$$

where $\tilde{\Psi}^*$ is the optimal objective for the relaxed problem (4.6).

Compared to (4.3), problem (4.6) eliminates the time coupling across variables $\{\mathbf{q}_t, \forall t\}$ by replacing (4.3b) and (4.3c) with (4.5). Since (4.6) is a relaxed version of (4.3) with the optimal objective $\tilde{\Psi}^* \leq \Psi^*$, if one solves (4.6) instead of (4.3), it will be prudent to derive an optimality bound on Ψ^*, provided that the sequence of solutions $\{\mathbf{x}_t, \forall t\}$ obtained by solving (4.6) is feasible for the relaxed constraints (4.3b) and (4.3c). Regarding the relaxed problem (4.6), using arguments similar to those in [116, Theorem 4.5], it can be shown that if the random state $\mathbf{s}_t$ is independent and identically distributed (i.i.d.) over time t, there exists a *stationary* control policy $\boldsymbol{\chi}^*(\cdot)$, which is a pure (possibly randomized) function of the realization of random state $\mathbf{s}_t$ (or the *observed* state $\mathbf{s}_t$); i.e., it satisfies (4.3d), as well as guarantees that $\mathbb{E}[\Psi_t(\boldsymbol{\chi}^*(\mathbf{s}_t))] = \tilde{\Psi}^*$ and $\mathbb{E}[\mathbf{A}\boldsymbol{\chi}^*(\mathbf{s}_t) + \mathbf{c}_t] \leq \mathbf{0}$. As the optimal policy $\boldsymbol{\chi}^*(\cdot)$ is time invariant, it implies that the *dynamic* problem (4.6) is equivalent to the following time-invariant *ensemble* program

$$\tilde{\Psi}^* := \min_{\boldsymbol{\chi}(\cdot)} \; \mathbb{E}\left[\Psi\big(\boldsymbol{\chi}(\mathbf{s}_t); \mathbf{s}_t\big)\right] \tag{4.7a}$$

$$\text{s.t.} \;\; \mathbb{E}[\mathbf{A}\boldsymbol{\chi}(\mathbf{s}_t) + \mathbf{c}(\mathbf{s}_t)] \leq \mathbf{0} \tag{4.7b}$$

$$\boldsymbol{\chi}(\mathbf{s}_t) \in \mathcal{X}, \; \forall \mathbf{s}_t \in \mathcal{S} \tag{4.7c}$$

where $\boldsymbol{\chi}(\mathbf{s}_t) := \mathbf{x}_t$, $\mathbf{c}(\mathbf{s}_t) = \mathbf{c}_t$, and $\Psi\big(\boldsymbol{\chi}(\mathbf{s}_t); \mathbf{s}_t\big) := \Psi_t(\mathbf{x}_t)$; set $\mathcal{S}$ is the sample space of $\mathbf{s}_t$, and the constraint (4.7c) holds almost surely. Observe that the index t in (4.7) can be dropped, since the expectation is taken over the distribution of random variable $\mathbf{s}_t$, which is time-invariant. Leveraging the equivalent form (4.7), the remaining task boils down to finding the optimal policy that achieves the minimal objective in (4.7a) and obeys the constraints (4.7b) and (4.7c).[2] Note that the optimization in (4.7) is with respect to a stationary policy $\boldsymbol{\chi}(\cdot)$, which is an infinite dimensional problem in the primal domain. However, there is a finite number of expected constraints [cf. (4.7b)]. Thus, the dual problem contains a finite number of variables, hinting to the effect that solving (4.7) is tractable in the dual domain [108, 130].

4.3.2 Lagrange dual and optimal policy

With $\boldsymbol{\lambda} \in \mathbb{R}_+^I$ denoting the Lagrange multipliers associated with (4.7b), the Lagrangian of (4.7) is

$$\mathcal{L}(\boldsymbol{\chi}, \boldsymbol{\lambda}) := \mathbb{E}\big[\mathcal{L}_t(\mathbf{x}_t, \boldsymbol{\lambda})\big] \tag{4.8}$$

[2]Though there may exist other time-dependent policies that generate the optimal solution to (4.6), our focus is restricted to the one that purely depends on the state $\mathbf{s} \in \mathcal{S}$, which can be time-independent [116, Theorem 4.5].

with $\boldsymbol{\lambda} \geq \mathbf{0}$, and the instantaneous Lagrangian is

$$\mathcal{L}_t(\mathbf{x}_t, \boldsymbol{\lambda}) := \Psi_t(\mathbf{x}_t) + \boldsymbol{\lambda}^\top (\mathbf{A}\mathbf{x}_t + \mathbf{c}_t) \tag{4.9}$$

where constraint (4.7c) remains implicit. Notice that the instantaneous objective $\Psi_t(\mathbf{x}_t)$ and the instantaneous constraint $\mathbf{A}\mathbf{x}_t + \mathbf{c}_t$ are both parameterized by the observed state $\mathbf{s}_t := [\boldsymbol{\phi}_t^\top, \mathbf{c}_t^\top]^\top$ at time t; i.e., $\mathcal{L}_t(\mathbf{x}_t, \boldsymbol{\lambda}) = \mathcal{L}(\boldsymbol{\chi}(\mathbf{s}_t), \boldsymbol{\lambda}; \mathbf{s}_t)$.

Correspondingly, the Lagrange dual function is defined as the minimum of the Lagrangian over the all feasible primal variables [14], given by

$$\begin{aligned} \mathcal{D}(\boldsymbol{\lambda}) : &= \min_{\{\boldsymbol{\chi}(\mathbf{s}_t) \in \mathcal{X},\ \forall \mathbf{s}_t \in \mathcal{S}\}} \mathcal{L}(\boldsymbol{\chi}, \boldsymbol{\lambda}) \\ &= \min_{\{\boldsymbol{\chi}(\mathbf{s}_t) \in \mathcal{X},\ \forall \mathbf{s}_t \in \mathcal{S}\}} \mathbb{E}\big[\mathcal{L}(\boldsymbol{\chi}(\mathbf{s}_t), \boldsymbol{\lambda}; \mathbf{s}_t)\big]. \end{aligned} \tag{4.10a}$$

Note that the optimization in (4.10a) is still w.r.t. a function. To facilitate the optimization, we re-write (4.10a) relying on the so-termed *interchangeability principle* [145, Theorem 7.80].

Lemma 13. *Let $\boldsymbol{\xi}$ denote a random variable on Ξ, and $\mathcal{H} := \{h(\,\cdot\,) : \Xi \to \mathbb{R}^n\}$ denote the function space of all the functions on Ξ. For any $\boldsymbol{\xi} \in \Xi$, if $f(\,\cdot\,, \boldsymbol{\xi}) : \mathbb{R}^n \to \mathbb{R}$ is a proper and lower semicontinuous convex function, then it follows that*

$$\min_{h(\cdot) \in \mathcal{H}} \mathbb{E}\big[f(h(\boldsymbol{\xi}), \boldsymbol{\xi})\big] = \mathbb{E}\left[\min_{\mathbf{h} \in \mathbb{R}^n} f(\mathbf{h}, \boldsymbol{\xi})\right]. \tag{4.10b}$$

Lemma 13 implies that under mild conditions, we can replace the optimization over a function space with (infinitely many) point-wise optimization problems. In the context here, we assume that $\Psi_t(\mathbf{x}_t)$ is proper, lower semicontinuous, and strongly convex (cf. Assumption 2 in Section V). Thus, for given finite $\boldsymbol{\lambda}$ and $\mathbf{s}_t$, $\mathcal{L}(\,\cdot\,, \boldsymbol{\lambda}; \mathbf{s}_t)$ is also strongly convex, proper and lower semicontinuous. Therefore, applying Lemma 13 yields

$$\min_{\{\boldsymbol{\chi}(\cdot) : \mathcal{S} \to \mathcal{X}\}} \mathbb{E}\big[\mathcal{L}(\boldsymbol{\chi}(\mathbf{s}_t), \boldsymbol{\lambda}; \mathbf{s}_t)\big] = \mathbb{E}\big[\min_{\boldsymbol{\chi}(\mathbf{s}_t) \in \mathcal{X}} \mathcal{L}(\boldsymbol{\chi}(\mathbf{s}_t), \boldsymbol{\lambda}; \mathbf{s}_t)\big] \tag{4.10c}$$

where the minimization and the expectation are interchanged. Accordingly, we re-write (4.10a) in the following form

$$\mathcal{D}(\boldsymbol{\lambda}) = \mathbb{E}\left[\min_{\boldsymbol{\chi}(\mathbf{s}_t) \in \mathcal{X}} \mathcal{L}(\boldsymbol{\chi}(\mathbf{s}_t), \boldsymbol{\lambda}; \mathbf{s}_t)\right] = \mathbb{E}\left[\min_{\mathbf{x}_t \in \mathcal{X}} \mathcal{L}_t(\mathbf{x}_t, \boldsymbol{\lambda})\right]. \tag{4.10d}$$

Likewise, for the instantaneous dual function $\mathcal{D}_t(\boldsymbol{\lambda}) = \mathcal{D}(\boldsymbol{\lambda}; \mathbf{s}_t) := \min_{\mathbf{x}_t \in \mathcal{X}} \mathcal{L}_t(\mathbf{x}_t, \boldsymbol{\lambda})$, the dual problem of (4.7) is

$$\max_{\boldsymbol{\lambda} \geq \mathbf{0}} \mathcal{D}(\boldsymbol{\lambda}) := \mathbb{E}\left[\mathcal{D}_t(\boldsymbol{\lambda})\right]. \tag{4.11}$$

In accordance with the ensemble primal problem (4.7), we will henceforth refer to (4.11) as the *ensemble* dual problem.

If the optimal Lagrange multiplier $\boldsymbol{\lambda}^*$ associated with (4.7b) were known, then optimizing (4.7) and consequently (4.6) would be equivalent to minimizing the Lagrangian $\mathcal{L}(\boldsymbol{\chi}, \boldsymbol{\lambda}^*)$ or infinitely many instantaneous $\{\mathcal{L}_t(\mathbf{x}_t, \boldsymbol{\lambda}^*)\}$, over the set $\mathcal{X}$ [17]. We restate this assertion as follows.

Proposition 3. *Consider the optimization problem in* (4.7)*. Given a realization* $\mathbf{s}_t$*, and the optimal Lagrange multiplier* $\boldsymbol{\lambda}^*$ *associated with the constraints* (4.7b)*, the optimal instantaneous resource allocation decision is*

$$\mathbf{x}_t^* = \boldsymbol{\chi}^*(\mathbf{s}_t) \in \arg \min_{\boldsymbol{\chi}(\mathbf{s}_t) \in \mathcal{X}} \mathcal{L}(\mathbf{x}_t, \boldsymbol{\lambda}^*; \mathbf{s}_t) \tag{4.12}$$

where $\in$ *accounts for possibly multiple minimizers of* $\mathcal{L}_t$*.*

When the realizations $\{\mathbf{s}_t\}$ are obtained sequentially, one can generate a sequence of optimal solutions $\{\mathbf{x}_t^*\}$ correspondingly for the dynamic problem (4.6). To obtain the optimal allocation in (4.12) however, $\boldsymbol{\lambda}^*$ must be known. This fact motivates our novel "learn-and-adapt" stochastic dual gradient (LA-SDG) method in Section 4.4. To this end, we will first outline the celebrated stochastic dual gradient iteration (a.k.a. Lyapunov optimization).

4.3.3 Revisiting stochastic dual (sub)gradient

To solve (4.11), a standard gradient iteration involves sequentially taking expectations over the distribution of $\mathbf{s}_t$ to compute the gradient. Note that when the Lagrangian minimization (cf. (4.12)) admits possibly multiple minimizers, a subgradient iteration is employed instead of the gradient one [14]. This is challenging because the distribution of $\mathbf{s}_t$ is typically unknown in practice. But even if the joint probability distribution functions were available, finding the expectations is not scalable as the dimensionality of $\mathbf{s}_t$ grows.

A common remedy to this challenge is stochastic approximation [131, 116], which corresponds to the following SDG iteration

$$\boldsymbol{\lambda}_{t+1} = \left[\boldsymbol{\lambda}_t + \mu \nabla \mathcal{D}_t(\boldsymbol{\lambda}_t)\right]^+, \ \forall t \tag{4.13a}$$

where μ is a positive (and typically pre-selected constant) stepsize. The stochastic (sub)gradient $\nabla\mathcal{D}_t(\boldsymbol{\lambda}_t) = \mathbf{A}\mathbf{x}_t + \mathbf{c}_t$ is an unbiased estimate of the true (sub)gradient; that is, $\mathbb{E}[\nabla\mathcal{D}_t(\boldsymbol{\lambda}_t)] = \nabla\mathcal{D}(\boldsymbol{\lambda}_t)$. Hence, the primal $\mathbf{x}_t$ can be found by solving the following instantaneous sub-problems, one per t

$$\mathbf{x}_t \in \arg\min_{\mathbf{x}_t \in \mathcal{X}} \mathcal{L}_t(\mathbf{x}_t, \boldsymbol{\lambda}_t). \tag{4.13b}$$

The iterate $\boldsymbol{\lambda}_{t+1}$ in (4.13a) depends only on the probability distribution of $\mathbf{s}_t$ through the stochastic (sub)gradient $\nabla\mathcal{D}_t(\boldsymbol{\lambda}_t)$. Consequently, the process $\{\boldsymbol{\lambda}_t\}$ is Markov with invariant transition probability when $\mathbf{s}_t$ is stationary. An interesting observation is that since $\nabla\mathcal{D}_t(\boldsymbol{\lambda}_t) := \mathbf{A}\mathbf{x}_t + \mathbf{c}_t$, the dual iteration can be written as [cf. (4.13a)]

$$\boldsymbol{\lambda}_{t+1}/\mu = [\boldsymbol{\lambda}_t/\mu + \mathbf{A}\mathbf{x}_t + \mathbf{c}_t]^+ \,, \ \forall t \tag{4.14}$$

which coincides with (4.3b) for $\boldsymbol{\lambda}_t/\mu = \mathbf{q}_t$; see also [116, 70, 163] for a virtual queue interpretation of this parallelism.

Thanks to its low complexity and robustness to non-stationary scenarios, SDG is widely used in various areas, including adaptive signal processing [81], stochastic network optimization [116, 70, 69], and energy management in power grids [163, 155]. For network management in particular, this iteration entails a cost-delay tradeoff as summarized next; see e.g., [116].

Proposition 4. *If Ψ^* is the optimal cost in* (4.3) *under any feasible control policy with the state distribution available, and if a constant stepsize μ is used in* (4.13a)*, the SDG recursion* (4.13) *achieves an $\mathcal{O}(\mu)$-optimal solution in the sense that*

$$\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[\Psi_t\left(\mathbf{x}_t(\boldsymbol{\lambda}_t)\right)\right] \leq \Psi^* + \mathcal{O}(\mu) \tag{4.15a}$$

where $\mathbf{x}_t(\boldsymbol{\lambda}_t)$ denotes the decisions obtained from (4.13b)*, and it incurs a steady-state queue length $\mathcal{O}(1/\mu)$, namely*

$$\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[\mathbf{q}_t\right] = \mathcal{O}\left(\frac{1}{\mu}\right). \tag{4.15b}$$

Proposition 4 asserts that SDG with stepsize μ will asymptotically yield an $\mathcal{O}(\mu)$-optimal solution [14, Prop. 8.2.11], and it will have steady-state queue length $\mathbf{q}_\infty$ inversely proportional to μ. This optimality gap is standard, because iteration (4.13a) with a constant stepsize[3] will

[3]A vanishing stepsize in the stochastic approximation iterations can ensure convergence, but necessarily implies an unbounded queue length as $\mu \to 0$ [116].

Algorithm 5 LA-SDG for Stochastic Network Optimization

```
1: Initialize: dual iterate $\boldsymbol{\lambda}_1$, empirical dual iterate $\hat{\boldsymbol{\lambda}}_1$, queue length $\mathbf{q}_1$, control variable $\boldsymbol{\theta} = \sqrt{\mu}\log^2(\mu)\cdot\mathbf{1}$, and proper stepsizes $\mu$ and $\{\eta_t, \forall t\}$.
2: for $t = 1, 2 \ldots$ do
3:     Resource allocation (1st gradient):
4:     Construct the effective dual variable via (4.17b), observe the current state $\mathbf{s}_t$, and obtain resource allocation $\mathbf{x}_t(\boldsymbol{\gamma}_t)$ by minimizing online Lagrangian (4.17a).
5:     Update the instantaneous queue length $\mathbf{q}_{t+1}$ via
           $$\mathbf{q}_{t+1} = \left[\mathbf{q}_t + \left(\mathbf{A}\mathbf{x}_t(\boldsymbol{\gamma}_t) + \mathbf{c}_t\right)\right]^+, \ \forall t. \quad (4.16)$$
6:     Sample recourse (2nd gradient):
7:     Obtain variable $\mathbf{x}_t(\hat{\boldsymbol{\lambda}}_t)$ by solving online Lagrangian minimization with sample $\mathbf{s}_t$ via (4.18b).
8:     Update the empirical dual variable $\hat{\boldsymbol{\lambda}}_{t+1}$ via (4.18a).
9: end for
```

converge to a neighborhood of the optimum $\boldsymbol{\lambda}^*$ [81]. Under mild conditions, the optimal multiplier is bounded, i.e., $\boldsymbol{\lambda}^* = \mathcal{O}(1)$, so that the steady-state queue length $\mathbf{q}_\infty$ naturally scales with $\mathcal{O}(1/\mu)$ since it hovers around $\boldsymbol{\lambda}^*/\mu$; see (4.14). As a consequence, to achieve near optimality (sufficiently small μ), SDG incurs large average queue lengths, and thus undesired average delay as per Little's law [116]. To overcome this limitation, we develop next an *online* approach, which can improve SDG's cost-delay tradeoff, while still preserving its affordable complexity and adaptability.

4.4 LA-SDG: Learn-and-adapt SDG

Our main approach is derived in this section, by nicely leveraging both learning and optimization tools. Its decentralized implementation is also developed.

4.4.1 LA-SDG as a foresighted learning scheme

The intuition behind our learn-and-adapt stochastic dual gradient (LA-SDG) approach is to incrementally learn network state statistics from observed data while adapting resource allocation driven by the learning process. A key element of LA-SDG could be termed as "foresighted" learning because instead of myopically learning the exact optimal argument from empirical data, LA-SDG maintains the capability to hedge against the risk of "future non-stationarities."

The proposed LA-SDG is summarized in Algorithm 5. It involves the queue length $\mathbf{q}_t$ and an empirical dual variable $\hat{\boldsymbol{\lambda}}_t$, along with a bias-control variable $\boldsymbol{\theta}$ to ensure that LA-SDG will attain near optimality in the steady state [cf. Theorems 7 and 8]. At each time slot t, LA-SDG obtains two stochastic gradients using the current $\mathbf{s}_t$: One for online resource allocation, and another one for sample learning/recourse. For the first gradient (lines 3-5), contrary to SDG that relies on the stochastic multiplier estimate $\boldsymbol{\lambda}_t$ [cf. (4.13b)], LA-SDG minimizes the instantaneous Lagrangian

$$\mathbf{x}_t(\boldsymbol{\gamma}_t) \in \arg\min_{\mathbf{x}_t \in \mathcal{X}} \mathcal{L}_t(\mathbf{x}_t, \boldsymbol{\gamma}_t) \tag{4.17a}$$

which depends on what we term *effective* multiplier, given by

$$\underbrace{\boldsymbol{\gamma}_t}_{\text{effective multiplier}} = \underbrace{\hat{\boldsymbol{\lambda}}_t}_{\text{statistical learning}} + \underbrace{\mu \mathbf{q}_t - \boldsymbol{\theta}}_{\text{online adaptation}}, \ \forall t. \tag{4.17b}$$

Variable $\boldsymbol{\gamma}_t$ also captures the effective price, which is a linear combination of the empirical $\hat{\boldsymbol{\lambda}}_t$ and the queue length $\mathbf{q}_t$, where the control variable μ tunes the weights of these two factors, and $\boldsymbol{\theta}$ controls the bias of $\boldsymbol{\gamma}_t$ in the steady state [69]. As a single pass of SDG "wastes" valuable online samples, LA-SDG resolves this limitation in a learning step by evaluating a second gradient (lines 6-8); that is, LA-SDG simply finds the stochastic gradient of (4.11) at the previous empirical dual variable $\hat{\boldsymbol{\lambda}}_t$, and implements a gradient ascent update as

$$\hat{\boldsymbol{\lambda}}_{t+1} = \left[\hat{\boldsymbol{\lambda}}_t + \eta_t \big(\mathbf{A}\mathbf{x}_t(\hat{\boldsymbol{\lambda}}_t) + \mathbf{c}_t\big)\right]^+, \ \forall t \tag{4.18a}$$

where η_t is a proper diminishing stepsize, and the "virtual" allocation $\mathbf{x}_t(\hat{\boldsymbol{\lambda}}_t)$ can be found by solving

$$\mathbf{x}_t(\hat{\boldsymbol{\lambda}}_t) \in \arg\min_{\mathbf{x}_t \in \mathcal{X}} \mathcal{L}_t(\mathbf{x}_t, \hat{\boldsymbol{\lambda}}_t). \tag{4.18b}$$

Note that different from $\mathbf{x}_t(\boldsymbol{\gamma}_t)$ in (4.17a), the "virtual" allocation $\mathbf{x}_t(\hat{\boldsymbol{\lambda}}_t)$ will not be physically implemented. The multiplicative constant μ in (4.17b) controls the degree of adaptability, and allows for adaptation even in the steady state ($t \to \infty$), but the vanishing η_t is for learning, as we shall discuss next.

The key idea of LA-SDG is to empower adaptive resource allocation (via $\boldsymbol{\gamma}_t$) with the learning process (effected through $\hat{\boldsymbol{\lambda}}_t$). As a result, the construction of $\boldsymbol{\gamma}_t$ relies on $\hat{\boldsymbol{\lambda}}_t$, but not vice versa. For a better illustration of the effective price (4.17b), we call $\hat{\boldsymbol{\lambda}}_t$ the statistically learnt price to obtain the exact optimal argument of the expected problem (4.11). We also call $\mu \mathbf{q}_t$ (which is exactly $\boldsymbol{\lambda}_t$ as shown in (4.13a)) the online adaptation term since it can track the instantaneous change of system statistics. Intuitively, a large μ will allow the effective policy to quickly respond to instantaneous variations so that the policy gains improved control of queue lengths, while a small μ puts more weight on learning from historical samples so that the allocation strategy will incur less variance in the steady state. In this sense, LA-SDG can attains both statistical efficiency and adaptability.

Distinctly different from SDG that combines statistical learning with resource allocation into a single adaptation step [cf. (4.13a)], LA-SDG performs these two tasks into two intertwined steps: resource allocation (4.17), and statistical learning (4.18). The additional learning step adopts diminishing stepsize to find the "best empirical" dual variable from all observed network states. This pair of complementary gradient steps endows LA-SDG with its attractive properties. In its transient stage, the extra gradient evaluations and empirical dual variables accelerate the convergence speed of SDG; while in the steady stage, the empirical multiplier approaches the optimal one, which significantly reduces the steady-state queue lengths.

Remark 1. Readers familiar with algorithms on statistical learning and stochastic network optimization can recognize their similarities and differences with LA-SDG.

(P1) SDG in [116] involves only the first part of LA-SDG (1st gradient), where the allocation policy purely relies on stochastic estimates of multipliers or instantaneous queue lengths, i.e., $\gamma_t = \mu \mathbf{q}_t$. In contrast, LA-SDG further leverages statistical learning from streaming data.

(P2) Several schemes have been developed recently for statistical learning at scale to find $\hat{\boldsymbol{\lambda}}_t$, namely, SAG in [132] and SAGA in [45]. However, directly applying $\gamma_t = \hat{\boldsymbol{\lambda}}_t$ to allocate resources causes infeasibility. For a finite time t, $\hat{\boldsymbol{\lambda}}_t$ is δ-optimal[4] for (4.11), and the primal variable $\mathbf{x}_t(\hat{\boldsymbol{\lambda}}_t)$ in turn is δ-feasible with respect to (4.7b) that is necessary for (4.3c). Since $\mathbf{q}_t$ essentially accumulates online constraint violations of (4.7b), it will grow linearly with t and eventually become unbounded.

4.4.2 LA-SDG as a modified heavy-ball iteration

The heavy-ball iteration belongs to the family of momentum-based first-order methods, and has well-documented acceleration merits in the deterministic setting [128]. Motivated by its convergence speed in solving deterministic problems, stochastic heavy-ball methods have been also pursued recently [175, 97].

The stochastic version of the heavy-ball iteration is [175]

$$\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t + \mu \nabla \mathcal{D}_t(\boldsymbol{\lambda}_t) + \beta(\boldsymbol{\lambda}_t - \boldsymbol{\lambda}_{t-1}), \ \forall t \tag{4.19}$$

[4]Iterate $\hat{\boldsymbol{\lambda}}_t$ is δ-optimal if $\|\hat{\boldsymbol{\lambda}}_t - \boldsymbol{\lambda}^*\| \leq \mathcal{O}(\delta)$, and likewise for δ-feasibility.

where $\mu > 0$ is an appropriate constant stepsize, $\beta \in [0, 1)$ denotes the momentum factor, and the stochastic gradient $\nabla\mathcal{D}_t(\boldsymbol{\lambda}_t)$ can be found by solving (4.13b) using heavy-ball iterate $\boldsymbol{\lambda}_t$. This iteration exhibits attractive convergence rate during the initial stage, but its performance degrades in the steady state. Recently, the performance of momentum iterations (heavy-ball or Nesterov) with constant stepsize μ and momentum factor β, has been proved equivalent to SDG with constant $\mu/(1-\beta)$ per iteration [175]. Since SDG with a large stepsize converges fast at the price of considerable loss in optimality, the momentum methods naturally inherit these attributes.

To see the influence of the momentum term, consider expanding the iteration (4.19) as

$$\begin{aligned}\boldsymbol{\lambda}_{t+1} &= \boldsymbol{\lambda}_t + \mu\nabla\mathcal{D}_t(\boldsymbol{\lambda}_t) + \beta(\boldsymbol{\lambda}_t - \boldsymbol{\lambda}_{t-1})\\ &= \boldsymbol{\lambda}_t + \mu\nabla\mathcal{D}_t(\boldsymbol{\lambda}_t) + \beta\left[\mu\nabla\mathcal{D}_{t-1}(\boldsymbol{\lambda}_{t-1}) + \beta(\boldsymbol{\lambda}_{t-1} - \boldsymbol{\lambda}_{t-2})\right]\\ &= \boldsymbol{\lambda}_t + \underbrace{\mu\textstyle\sum_{\tau=1}^{t}\beta^{t-\tau}\nabla\mathcal{D}_\tau(\boldsymbol{\lambda}_\tau)}_{\text{accumulated gradient}} + \underbrace{\beta^t(\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_0)}_{\text{initial state}}. \end{aligned}\tag{4.20}$$

The stochastic heavy-ball method will accelerate convergence in the initial stage thanks to the accumulated gradients, and it will gradually forget the initial state. As t increases however, the algorithm also incurs a worst-case oscillation $\mathcal{O}(\mu/(1-\beta))$, which degrades performance in terms of objective values when compared to SDG with stepsize μ. This is in agreement with the theoretical analysis in [175, Theorem 11].

Different from standard momentum methods, LA-SDG nicely inherits the fast convergence in the initial stage, while reducing the oscillation of stochastic momentum methods in the steady state. To see this, consider two consecutive iterations (4.17b)

$$\gamma_{t+1} = \hat{\boldsymbol{\lambda}}_{t+1} + \mu\mathbf{q}_{t+1} - \boldsymbol{\theta} \tag{4.21a}$$

$$\gamma_t = \hat{\boldsymbol{\lambda}}_t + \mu\mathbf{q}_t - \boldsymbol{\theta} \tag{4.21b}$$

and subtract them, to arrive at

$$\begin{aligned}\gamma_{t+1} &= \gamma_t + \mu\left(\mathbf{q}_{t+1} - \mathbf{q}_t\right) + (\hat{\boldsymbol{\lambda}}_{t+1} - \hat{\boldsymbol{\lambda}}_t)\\ &= \gamma_t + \mu\nabla\mathcal{D}_t(\gamma_t) + (\hat{\boldsymbol{\lambda}}_{t+1} - \hat{\boldsymbol{\lambda}}_t), \quad \forall t.\end{aligned}\tag{4.22}$$

Here the equalities in (4.22) follows from $\nabla\mathcal{D}_t(\gamma_t) = \mathbf{A}\mathbf{x}_t(\gamma_t) + \mathbf{c}_t$ in $\mathbf{q}_t$ recursion (4.16), and with a sufficiently large $\boldsymbol{\theta}$, the projection in (4.16) rarely (with sufficiently low probability) takes effect since the steady-state $\mathbf{q}_t$ will hover around $\boldsymbol{\theta}/\mu$; see the details of Theorem 7 and the proof thereof.

Comparing the LA-SDG iteration (4.22) with the stochastic heavy-ball iteration (4.19), both of them correct the iterates using the stochastic gradient $\nabla\mathcal{D}_t(\boldsymbol{\gamma}_t)$ or $\nabla\mathcal{D}_t(\boldsymbol{\lambda}_t)$. However, LA-SDG incorporates the variation of a learning sequence (also known as a reference sequence) $\{\hat{\boldsymbol{\lambda}}_t\}$ into the recursion of the main iterate $\boldsymbol{\gamma}_t$, other than heavy-ball's momentum term $\beta(\boldsymbol{\lambda}_t - \boldsymbol{\lambda}_{t-1})$. Since the variation of learning iterate $\hat{\boldsymbol{\lambda}}_t$ eventually diminishes as t increases, keeping the learning sequence enables LA-SDG to enjoy accelerated convergence in the initial (transient) stage compared to SDG, while avoiding large oscillation in the steady state compared to the stochastic heavy-ball method. We formally remark this obervation next.

Remark 2. LA-SDG offers a fresh approach to designing stochastic optimization algorithms in a dynamic environment. While directly applying the momentum-based iteration to a stochastic setting may lead to unsatisfactory steady-state performance, it is promising to carefully design a reference sequence that exactly converges to the optimal argument. Therefore, algorithms with improved convergence (e.g., the second-order method in [178]) can also be incorporated as a reference sequence to further enhance the performance of LA-SDG.

4.4.3 Complexity and distributed implementation of LA-SDG

This section introduces a fully distributed implementation of LA-SDG by exploiting the problem structure of network resource allocation. For notational brevity, collect the variables representing outgoing links from node i in $\mathbf{x}_t^i := \{x_t^{ij}, \forall j \in \mathcal{N}_i\}$ with $\mathcal{N}_i$ denoting the index set of outgoing neighbors of node i. Let also $\mathbf{s}_t^i := [\boldsymbol{\phi}_t^i; c_t^i]$ denote the random state at node i. It will be shown that the learning and allocation decision per time slot t is processed locally per node i based on its local state $\mathbf{s}_t^i$.

To this end, rewrite the Lagrangian minimization for a general dual variable $\boldsymbol{\lambda} \in \mathbb{R}_+^I$ at time t as [cf. (4.17a) and (4.18b)]

$$\min_{\mathbf{x}_t \in \mathcal{X}} \sum_{i \in \mathcal{I}} \Psi^i(\mathbf{x}_t^i; \boldsymbol{\phi}_t^i) + \sum_{i \in \mathcal{I}} \lambda^i (\mathbf{A}_{(i,:)}\mathbf{x}_t + c_t^i) \tag{4.23}$$

where λ^i is the i-th entry of vector $\boldsymbol{\lambda}$, and $\mathbf{A}_{(i,:)}$ denotes the i-th row of the node-incidence matrix $\mathbf{A}$. Clearly, $\mathbf{A}_{(i,:)}$ selects entries of $\mathbf{x}_t$ associated with the in- and out-links of node i. Therefore, the subproblem at node i is

$$\min_{\mathbf{x}_t^i \in \mathcal{X}^i} \Psi^i(\mathbf{x}_t^i; \boldsymbol{\phi}_t^i) + \sum_{j \in \mathcal{N}_i} (\lambda^j - \lambda^i) x_t^{ji} \tag{4.24}$$

where $\mathcal{X}^i$ is the feasible set of primal variable $\mathbf{x}_t^i$. In the case of (4.3d), the feasible set $\mathcal{X}$ can be written as a Cartesian product of sets $\{\mathcal{X}^i, \forall i\}$, so that the projection of $\mathbf{x}_t$ to $\mathcal{X}$ is equivalent to separate projections of $\mathbf{x}_t^i$ onto $\mathcal{X}^i$. Note that $\{\lambda^j, \forall j \in \mathcal{N}_i\}$ will be available at node i by exchanging information with the neighbors per time t. Hence, given the effective multipliers γ_t^j (j-th entry of $\boldsymbol{\gamma}_t$) from its outgoing neighbors in $j \in \mathcal{N}_i$, node i is able to form an allocation decision $\mathbf{x}_t^i(\boldsymbol{\gamma}_t)$ by solving the convex programs (4.24) with $\lambda^j = \gamma_t^j$; see also (4.17a). Needless to mention, q_t^i can be locally updated via (4.16), that is

$$q_{t+1}^i = \left[q_t^i + \Big(\sum_{j:i\in\mathcal{N}_j} x_t^{ji}(\boldsymbol{\gamma}_t) - \sum_{j\in\mathcal{N}_i} x_t^{ij}(\boldsymbol{\gamma}_t) + c_t^i \Big) \right]^+ \tag{4.25}$$

where $\{x_t^{ji}(\boldsymbol{\gamma}_t)\}$ are the local measurements of arrival (departure) workloads from (to) its neighbors.

Likewise, the tentative primal variable $\mathbf{x}_t^i(\hat{\boldsymbol{\lambda}}_t)$ can be obtained at each node locally by solving (4.24) using the current sample $\mathbf{s}_t^i$ again with $\lambda^i = \hat{\lambda}_t^i$. By sending $\mathbf{x}_t^i(\hat{\boldsymbol{\lambda}}_t)$ to its outgoing neighbors, node i can update the empirical multiplier $\hat{\lambda}_{t+1}^i$ via

$$\hat{\lambda}_{t+1}^i = \left[\hat{\lambda}_t^i + \eta_t \Big(\sum_{j:i\in\mathcal{N}_j} x_t^{ji}(\hat{\boldsymbol{\lambda}}_t) - \sum_{j\in\mathcal{N}_i} x_t^{ij}(\hat{\boldsymbol{\lambda}}_t) + c_t^i \Big) \right]^+ \tag{4.26}$$

which, together with the local queue length q_{t+1}^i, also implies that the next γ_{t+1}^i can be obtained locally.

Compared with the classic SDG recursion (4.13a)-(4.13b), the distributed implementation of LA-SDG incurs only a factor of two increase in computational complexity. Next, we will further analytically establish that it can improve the delay of SDG by an order of magnitude with the same order of optimality gap.

4.5 Optimality and stability of LA-SDG

This section presents performance analysis of LA-SDG, which will rely on the following four assumptions.

Assumption 1. *The state $\mathbf{s}_t$ is bounded and i.i.d. over time t.*

Assumption 2. $\Psi_t(\mathbf{x}_t)$ *is proper, σ-strongly convex, lower semi-continuous, and has L_p-Lipschitz continuous gradient. Also, $\Psi_t(\mathbf{x}_t)$ is non-decreasing w.r.t. all entries of $\mathbf{x}_t$ over $\mathcal{X}$.*

Assumption 3. *There exists a stationary policy $\boldsymbol{\chi}(\cdot)$ satisfying $\boldsymbol{\chi}(\mathbf{s}_t) \in \mathcal{X}$ for all $\mathbf{s}_t$, and $\mathbb{E}[\mathbf{A}\boldsymbol{\chi}(\mathbf{s}_t) + \mathbf{c}_t] \leq -\boldsymbol{\zeta}$, where $\boldsymbol{\zeta} > \mathbf{0}$ is a slack vector constant.*

Assumption 4. *For any time t, the magnitude of the constraint is bounded, that is, $\|\mathbf{A}\mathbf{x}_t + \mathbf{c}_t\| \leq M,\ \forall \mathbf{x}_t \in \mathcal{X}$.*

Assumption 1 is typical in stochastic network resource allocation [70, 69, 51], and can be relaxed to an ergodic and stationary setting following [130, 49]. Assumption 2 requires the primal objective to be well behaved, meaning that it is bounded from below and has a unique optimal solution. Note that non-decreasing costs with increased resources are easily guaranteed with e.g., exponential and quadratic functions in our simulations. In addition, Assumption 2 ensures that the dual function has favorable properties, which are important for the ensuring stability analysis. Assumption 3 is Slater's condition, which guarantees the existence of a bounded optimal Lagrange multiplier [14], and is also necessary for queue stability [116]. Assumption 4 guarantees boundedness of the gradient of the instantaneous dual function, which is common in performance analysis of stochastic gradient-type algorithms [118].

Building upon the desirable properties of the primal problem, we next show that the corresponding dual function satisfies both smoothness and quadratic growth properties [66, 77], which will be critical to the subsequent analysis.

Lemma 14. *Under Assumption 2, the dual function $\mathcal{D}(\boldsymbol{\lambda})$ in* (4.11) *is $L_{\rm d}$-smooth, where $L_{\rm d} = \rho(\mathbf{A}^\top\mathbf{A})/\sigma$, and $\rho(\mathbf{A}^\top\mathbf{A})$ denotes the spectral radius of $\mathbf{A}^\top\mathbf{A}$. In addition, if $\boldsymbol{\lambda}$ lies in a compact set, there always exists a constant ϵ such that $\mathcal{D}(\boldsymbol{\lambda})$ satisfies the following quadratic growth property*

$$\mathcal{D}(\boldsymbol{\lambda}^*) - \mathcal{D}(\boldsymbol{\lambda}) \geq \frac{\epsilon}{2}\|\boldsymbol{\lambda}^* - \boldsymbol{\lambda}\|^2 \tag{4.27}$$

where $\boldsymbol{\lambda}^*$ is the optimal multiplier for the dual problem (4.11).

Proof: See Appendix A in the online version [32].

We start with the convergence of the empirical dual variables $\hat{\boldsymbol{\lambda}}_t$. Note that the update of $\hat{\boldsymbol{\lambda}}_t$ is a standard learning iteration from historical data, and it is not affected by future resource allocation decisions. Therefore, the theoretical result on SDG with diminishing stepsize is directly applicable [118, Sec. 2.2].

Lemma 15. *Let $\hat{\boldsymbol{\lambda}}_t$ denote the empirical dual variable in Algorithm 5, and $\boldsymbol{\lambda}^*$ the optimal argument for the dual problem* (4.11). *If the stepsize is chosen as $\eta_t = \frac{\alpha D}{M\sqrt{t}},\ \forall t$, with a constant*

$\alpha > 0$, a sufficient large constant $D > 0$, and M as in Assumption 4, then it holds that

$$\mathbb{E}\left[\mathcal{D}(\boldsymbol{\lambda}^*) - \mathcal{D}(\hat{\boldsymbol{\lambda}}_t)\right] \leq \max\{\alpha, \alpha^{-1}\}\,\frac{DM}{\sqrt{t}} \tag{4.28}$$

where the expectation is over all the random states $\mathbf{s}_t$ up to t.

Lemma 15 asserts that using a diminishing stepsize, the dual function value converges sub-linearly to the optimal value in expectation. In principle, D is the radius of the feasible set for the dual variable $\boldsymbol{\lambda}$ [118, Sec. 2.2]. However, as the optimal multiplier $\boldsymbol{\lambda}^*$ is bounded according to Assumption 3, one can always estimate a large enough D, and the estimation error will only affect the constant of the sub-optimality bound (4.28) through the scalar α. The sub-optimality bound in Lemma 15 holds in expectation, which averages over all possible sample paths $\{\mathbf{s}_1, \ldots, \mathbf{s}_t\}$.

As a complement to Lemma 15, the almost sure convergence of the empirical dual variables is established next to characterize the performance of each individual sample path.

Theorem 6. *For the sequence of empirical multipliers $\{\hat{\boldsymbol{\lambda}}_t\}$ in Algorithm 5, if the stepsizes are chosen as $\eta_t = \frac{\alpha D}{M\sqrt{t}}, \forall t$, with constants α, M, D defined in Lemma 15, it holds that*

$$\lim_{t\to\infty} \hat{\boldsymbol{\lambda}}_t = \boldsymbol{\lambda}^*, \quad \text{w.p.1} \tag{4.29}$$

where $\boldsymbol{\lambda}^*$ is the optimal dual variable for the expected dual function minimization (4.11).

Proof: The proof follows the steps in [14, Proposition 8.2.13], which is omitted here.

Building upon the asymptotic convergence of empirical dual variables for statistical learning, it becomes possible to analyze the online performance of LA-SDG. Clearly, the online resource allocation $\mathbf{x}_t$ is a function of the effective dual variable $\boldsymbol{\gamma}_t$ and the instantaneous network state $\mathbf{s}_t$ [cf. (4.17a)]. Therefore, the next step is to show that the effective dual variable $\boldsymbol{\gamma}_t$ also converges to the optimal argument of the expected problem (4.11), which would establish that the online resource allocation $\mathbf{x}_t$ is asymptotically optimal. However, directly analyzing the trajectory of $\boldsymbol{\gamma}_t$ is nontrivial, because the queue length $\{\mathbf{q}_t\}$ is coupled with the reference sequence $\{\hat{\boldsymbol{\lambda}}_t\}$ in $\boldsymbol{\gamma}_t$. To address this issue, rewrite the recursion of $\boldsymbol{\gamma}_t$ as

$$\boldsymbol{\gamma}_{t+1} = \boldsymbol{\gamma}_t + (\hat{\boldsymbol{\lambda}}_{t+1} - \hat{\boldsymbol{\lambda}}_t) + \mu(\mathbf{q}_{t+1} - \mathbf{q}_t),\ \forall t \tag{4.30}$$

where the update of $\boldsymbol{\gamma}_t$ depends on the variations of $\hat{\boldsymbol{\lambda}}_t$ and $\mathbf{q}_t$. We will first study the asymptotic behavior of queue lengths $\mathbf{q}_t$, and then derive the analysis of $\boldsymbol{\gamma}_t$ using the convergence of $\hat{\boldsymbol{\lambda}}_t$ in (4.29), and the recursion (4.30).

Define the time-varying target $\tilde{\boldsymbol{\theta}}_t = \boldsymbol{\lambda}^* - \hat{\boldsymbol{\lambda}}_t + \boldsymbol{\theta}$, which is the optimality residual of statistical learning $\boldsymbol{\lambda}^* - \hat{\boldsymbol{\lambda}}_t$ plus the bias-control variable $\boldsymbol{\theta}$. Per Theorem 6, it readily follows that $\lim_{t\to\infty} \tilde{\boldsymbol{\theta}}_t = \boldsymbol{\theta}$, w.p.1. By showing that $\mathbf{q}_t$ is attracted towards the time-varying target $\tilde{\boldsymbol{\theta}}_t/\mu$, we will further derive the stability of queue lengths.

Lemma 16. *With $\mathbf{q}_t$ and μ denoting queue length and stepsize, there exists a constant $B = \Theta(1/\sqrt{\mu})$, and a finite time $T_B < \infty$, such that for all $t \geq T_B$, if $\|\mathbf{q}_t - \tilde{\boldsymbol{\theta}}_t/\mu\| > B$, it holds in LA-SDG that*

$$\mathbb{E}\left[\left\|\mathbf{q}_{t+1} - \tilde{\boldsymbol{\theta}}_t/\mu\right\| \Big| \mathbf{q}_t\right] \leq \left\|\mathbf{q}_t - \tilde{\boldsymbol{\theta}}_t/\mu\right\| - \sqrt{\mu}, \text{ w.p.1.} \tag{4.31}$$

Proof: See Appendix B in the online version [32].

Lemma 16 reveals that when $\mathbf{q}_t$ is large and deviates from the time-varying target $\tilde{\boldsymbol{\theta}}_t/\mu$, it will be bounced back towards the target in the next time slot. Upon establishing this drift behavior of queues, we are on track to establish queue stability.

Theorem 7. *With $\mathbf{q}_t, \boldsymbol{\theta}$, and μ defined in* (4.17b)*, there exists a constant $\tilde{B} = \Theta(1/\sqrt{\mu})$ such that the queue length under LA-SDG converges to a neighborhood of $\boldsymbol{\theta}/\mu$ as*

$$\liminf_{t\to\infty} \; \|\mathbf{q}_t - \boldsymbol{\theta}/\mu\| \leq \tilde{B}, \;\; \text{w.p.1.} \tag{4.32a}$$

In addition, if we choose $\boldsymbol{\theta} = \mathcal{O}(\sqrt{\mu}\log^2(\mu))$, the long-term average expected queue length satisfies

$$\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[\mathbf{q}_t\right] = \mathcal{O}\left(\frac{\log^2(\mu)}{\sqrt{\mu}}\right), \text{ w.p.1.} \tag{4.32b}$$

Proof: See Appendix C in the online version [32].

Theorem 7 in (4.32a) asserts that the sequence of queue iterates converges (in the infimum sense) to a neighborhood of $\boldsymbol{\theta}/\mu$, where the radius of neighborhood region scales as $1/\sqrt{\mu}$. In addition to the sample path result, (4.32b) demonstrates that with a specific choice of $\boldsymbol{\theta}$, the queue length averaged over all sample paths will be $\mathcal{O}\left(\log^2(\mu)/\sqrt{\mu}\right)$. Together with Theorem 6, it suffices to have the effective dual variable converge to a neighborhood of the optimal multiplier $\boldsymbol{\lambda}^*$; that is, $\liminf_{t\to\infty} \boldsymbol{\gamma}_t = \boldsymbol{\lambda}^* + \mu\mathbf{q}_t - \boldsymbol{\theta} = \boldsymbol{\lambda}^* + \mathcal{O}(\sqrt{\mu})$, w.p.1. Notice that the SDG iterate $\boldsymbol{\lambda}_t$ in (4.13a) will also converge to a neighborhood of $\boldsymbol{\lambda}^*$. Therefore, intuitively LA-SDG will behave similar to SDG in the steady state, and its asymptotic performance follows from that of SDG. However, the difference is that through a careful choice of $\boldsymbol{\theta}$, for a sufficiently small μ, LA-SDG can improve the queue length $\mathcal{O}\left(1/\mu\right)$ under SDG by an order of magnitude.

In addition to feasibility, we formally establish in the next theorem that LA-SDG is asymptotically near-optimal.

Theorem 8. *Let* Ψ^* *be the optimal objective value of* (4.3) *under any feasible policy with distribution information about the state fully available. If the control variable is chosen as* $\boldsymbol{\theta} = \mathcal{O}(\sqrt{\mu}\log^2(\mu))$*, then with a sufficiently small* μ*, LA-SDG yields a near-optimal solution for* (4.3) *in the sense that*

$$\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[\Psi_t\left(\mathbf{x}_t(\boldsymbol{\gamma}_t)\right)\right] \leq \Psi^* + \mathcal{O}(\mu), \text{ w.p.1} \tag{4.33}$$

where $\mathbf{x}_t(\boldsymbol{\gamma}_t)$ *denotes the online operations obtained from the Lagrangian minimization* (4.17a).

Proof: See Appendix D in the online version [32].

Combining Theorems 7 and 8, we are ready to state that by setting $\boldsymbol{\theta} = \mathcal{O}(\sqrt{\mu}\log^2(\mu))$, LA-SDG is asymptotically $\mathcal{O}(\mu)$-optimal with an average queue length $\mathcal{O}(\log^2(\mu)/\sqrt{\mu})$. This result implies that LA-SDG is able to achieve a near-optimal cost-delay tradeoff $[\mu, \log^2(\mu)/\sqrt{\mu}]$; see [108, 116]. Comparing with the standard tradeoff $[\mu, 1/\mu]$ under SDG, the learn-and-adapt design of LA-SDG markedly improves the online performance in terms of delay. Note that a better tradeoff $[\mu, \log^2(\mu)]$ has been derived in [69] under the so-termed local polyhedral assumption. Observe though, that the considered setting in [69] is different from the one here. While the network state set $\mathcal{S}$ and the action set $\mathcal{X}$ in [69] are discrete and countable, LA-SDG allows continuous $\mathcal{S}$ and $\mathcal{X}$ with possibly infinite elements, and still be amenable to efficient and scalable online operations.

4.6 Numerical tests

This section presents numerical tests to confirm the analytical claims and demonstrate the merits of the proposed approach. We consider the geographical load balancing network of Section 4.2.2 with $K = 10$ data centers, and $J = 10$ mapping nodes. Performance is tested in terms of the time-averaged instantaneous network cost in (4.4), namely

$$\Psi_t(\mathbf{x}_t) := \sum_{k\in\mathcal{K}} p_t^k \left((x_t^{k0})^2 - e_t^k\right) + \sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{K}} b_t^{jk}(x_t^{jk})^2 \tag{4.34}$$

where the energy price p_t^k is uniformly distributed over $[10, 30]$; samples of the renewable supply $\{e_t^k\}$ are generated uniformly over $[10, 100]$; and the per-unit bandwidth cost is set to $b_t^{jk} =$

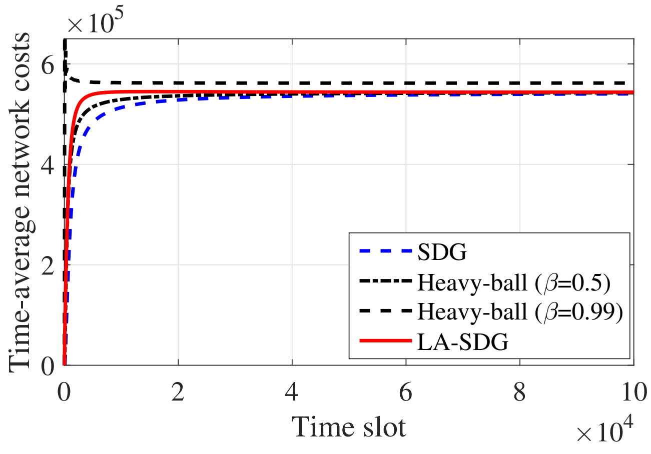

Figure 4.2: Comparison of time-averaged network costs.

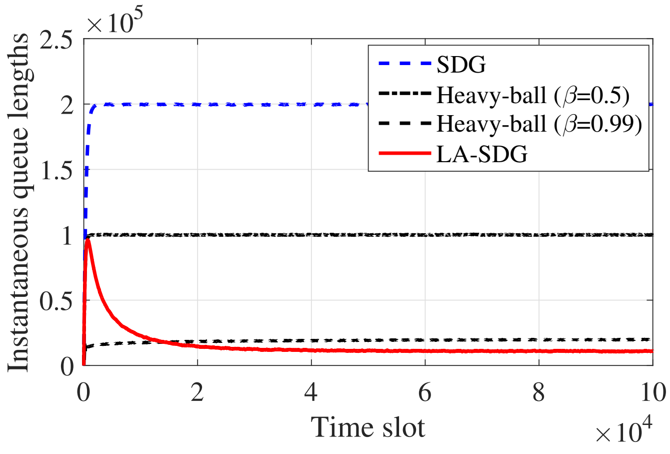

Figure 4.3: Instantaneous queue lengths summed over all nodes.

$40/\bar{x}^{jk}, \forall k, j$, with bandwidth limits $\{\bar{x}^{jk}\}$ generated from a uniform distribution within $[100, 200]$. The capacities at data centers $\{\bar{x}_t^{k0}\}$ are uniformly generated from $[100, 200]$. The delay-tolerant workloads $\{c_t^j\}$ arrive at each mapping node j according to a uniform distribution over $[10, 100]$. Clearly, the cost (4.34) and the state $\mathbf{s}_t$ here satisfy Assumptions 1 and 2. Finally, the stepsize is $\eta_t = 1/\sqrt{t}, \forall t$, the trade-off variable is $\mu = 0.2$, and the bias correction vector is chosen as $\boldsymbol{\theta} = 100\sqrt{\mu}\log^2(\mu)\mathbf{1}$ by default, but manually tuned in Figs. 4.5-4.6. We introduce two benchmarks: SDG in (4.13a) (see e.g., [116]), and the projected stochastic heavy-ball in (4.19) and $\beta = 0.5$ by default (see e.g., [97]). Unless otherwise stated, all simulated results were averaged over 50 Monte Carlo realizations.

Performance is first compared in terms of the time-averaged cost, and the instantaneous queue

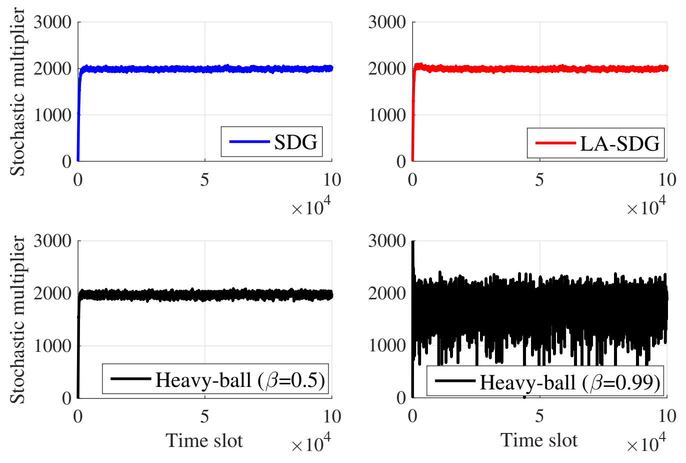

Figure 4.4: The evolution of stochastic multipliers at mapping node 1 ($\mu = 0.2$).

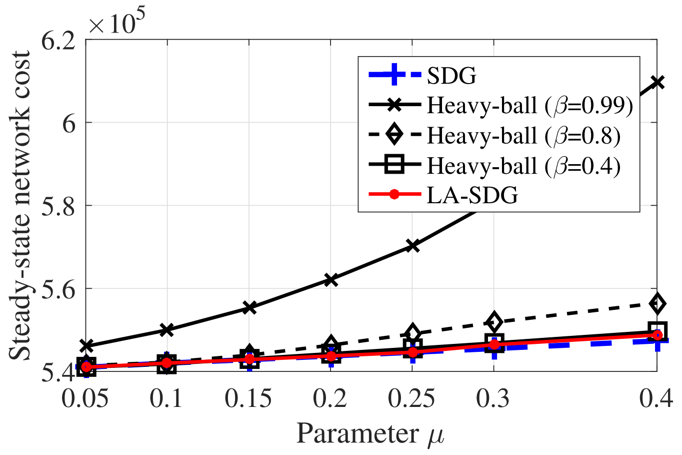

Figure 4.5: Comparison of steady-state network costs (after 10^6 slots).

length in Figs. 4.2 and 4.3. For the network cost, SDG, LA-SDG, and the heavy-ball iteration with $\beta = 0.5$ converge to almost the same value, while the heavy-ball method with a larger momentum factor $\beta = 0.99$ exhibits a pronounced optimality loss. LA-SDG and heavy-ball exhibit faster convergence than SDG as their running-average costs quickly arrive at the optimal operating phase by leveraging the learning process or the momentum acceleration. In this test, LA-SDG exhibits a much lower delay as its aggregated queue length is only 10% of that for the heavy-ball method with $\beta = 0.5$ and 4% of that for SDG. By using a larger β, the heavy-ball method incurs a much lower queue length relative to that of SDG, but still slightly higher than that of LA-SDG. Clearly, our learn-and-adapt procedure improves the delay performance.

Recall that the instantaneous resource allocation can be viewed as a function of the dual

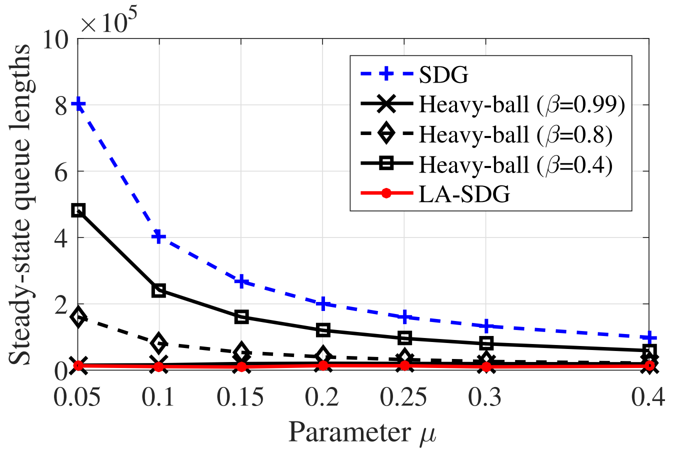

Figure 4.6: Steady-state queue lengths summed over all nodes (after 10^6 slots).

variable; see Proposition 3. Hence, the performance differences in Figs. 4.2-4.3 can be also anticipated by the different behavior of dual variables. In Fig. 4.4, the evolution of stochastic dual variables is plotted for a single Monte Carlo realization; that is the dual iterate in (4.13a) for SDG, the momentum iteration in (4.19) for the heavy-ball method, and the effective multiplier in (4.17b) for LA-SDG. As illustrated in (4.20), the performance of momentum iterations is similar to SDG with larger stepsize $\mu/(1-\beta)$. This is corroborated by Fig. 4.4, where the stochastic momentum iterate with $\beta = 0.5$ behaves similar to the dual iterates of SDG and LA-SDG, but its oscillation becomes prohibitively high with a larger factor $\beta = 0.99$, which nicely explains the higher cost in Fig. 4.2.

Since the cost-delay performance is sensitive to the choice of parameters μ and β, extensive experiments are further conducted among three algorithms using different values of μ and β in Figs. 4.5 and 4.6. The steady-state performance is evaluated by running algorithms for sufficiently long time, up to 10^6 slots. The steady-state costs of all three algorithms increase as μ becomes larger, and the costs of LA-SDG and the heavy-ball with small momentum factor $\beta = 0.4$ are close to that of SDG, while the costs of the heavy-ball with larger momentum factors $\beta = 0.8$ and $\beta = 0.99$ are much larger than that of SDG. Considering steady-state queue lengths (network delay), LA-SDG exhibits an order of magnitude lower amount than those of SDG and the heavy-ball with small β, under all choices of μ. Note that the heavy-ball with a sufficiently large factor $\beta = 0.99$ also has a very low queue length, but it incurs a higher cost than LA-SDG in Fig. 4.5 due to higher steady-state oscillation in Fig. 4.4.

4.7 Proofs of lemmas and theorems

Let us first state a simple but useful property regarding the primal-dual problems (4.7) and (4.11).

Proposition 5. *Under Assumptions 1-3, for the constrained optimization* (4.7) *with the optimal policy $\boldsymbol{\chi}^*(\cdot)$ and its optimal Lagrange multiplier $\boldsymbol{\lambda}^*$, it holds that $\mathbb{E}[\mathbf{A}\mathbf{x}_t^* + \mathbf{c}_t] = \mathbf{0}$ with $\mathbf{x}_t^* = \boldsymbol{\chi}^*(\mathbf{s}_t) \in \mathcal{X}$, and accordingly that $\nabla\mathcal{D}(\boldsymbol{\lambda}^*) = \mathbf{0}$.*

Proof: With $\boldsymbol{\lambda}^*$ denoting the optimal Lagrange multiplier with (4.7b), the Karush-Kuhn-Tucker (KKT) conditions [14] are

$$\left(\mathbb{E}[\nabla\Psi_t(\mathbf{x}_t^*)] + \mathbf{A}^\top\boldsymbol{\lambda}^*\right)^\top (\mathbb{E}[\mathbf{x}_t - \mathbf{x}_t^*]) \geq 0,\ \forall \mathbf{x}_t \in \mathcal{X} \tag{4.35a}$$

$$(\boldsymbol{\lambda}^*)^\top \mathbb{E}[\mathbf{A}\mathbf{x}_t^* + \mathbf{c}_t] = 0 \tag{4.35b}$$

$$\mathbb{E}[\mathbf{A}\mathbf{x}_t^* + \mathbf{c}_t] \leq \mathbf{0};\ \boldsymbol{\lambda}^* \geq \mathbf{0} \tag{4.35c}$$

where (4.35a) is the optimality condition of Lagrangian minimization, (4.35b) is the complementary slackness condition, and (4.35c) are the primal and dual feasibility conditions.

To establish the claim, let us first assume that there exists entry k that the inequality constraint (4.7b) is not active; i.e., $\mathbb{E}[\mathbf{A}_{(k,:)}\mathbf{x}_t^* + c_t^k] = -\zeta$ with the constant $\zeta > 0$, and $\mathbf{A}_{(k,:)}$ denoting the k-th row of $\mathbf{A}$. As each row of $\mathbf{A}$ has at least one entry equal to -1, we collect all indices of entries at k-th row with value -1 in set $\mathcal{E}_k^{-1}$ so that $\mathbf{A}_{(k,e)} = -1, \forall e \in \mathcal{E}_k^{-1}$.

Since $\mathbf{x}_t^*$ is feasible, we have $\mathbf{x}_t^* \geq \mathbf{0}$, and thus

$$\mathbb{E}[\mathbf{A}_{(k,:)}\mathbf{x}_t^* + c_t^k] = \mathbb{E}\left[\sum_{e\in\mathcal{E}} \mathbf{A}_{(k,e)}(x_t^e)^* + c_t^k\right] = -\zeta \tag{4.36}$$

which implies that

$$\mathbb{E}\big[\textstyle\sum_{e\in\mathcal{E}_k^{-1}}(x_t^e)^*\big] = \zeta + \mathbb{E}[c_t^k + \textstyle\sum_{e\in\mathcal{E}\setminus\mathcal{E}_k^{-1}} \mathbf{A}_{(k,e)}(x_t^e)^*] > 0. \tag{4.37}$$

According (4.35b), it further follows that $(\lambda^k)^* = 0$ since $(\lambda^k)^* \cdot \mathbb{E}[\mathbf{A}_{(k,:)}\mathbf{x}_t^* + c_t^k] = -(\lambda^k)^* \cdot \zeta = 0$. Now we are on track to show that it contradicts with (4.35a). Since $\mathbb{E}[\sum_{e\in\mathcal{E}_k^{-1}}(x_t^e)^*] > 0$, there exists at least an index j such that $\mathbb{E}[(x_t^j)^*] > 0,\ j \in \mathcal{E}_k^{-1}$. Choose $\mathbb{E}[\mathbf{x}_t]$ with $\mathbb{E}[x_t^{\tilde{j}}] = \mathbb{E}[(x_t^{\tilde{j}})^*], \forall \tilde{j} \neq j$ and $\mathbb{E}[x_t^j] = 0$, to have $\mathbb{E}[\mathbf{x}_t - \mathbf{x}_t^*] = [0, \ldots, -\mathbb{E}[(x_t^j)^*], \ldots, 0]^\top$. Recall that the feasible set $\mathcal{X}$ in (4.3d) contains only box constraints; i.e., $\mathcal{X} := \{\mathbf{x} \,|\, \mathbf{0} \leq \mathbf{x} \leq \bar{\mathbf{x}}\}$, which implies that the above selection of $\mathbf{x}_t$ is feasible. Hence, we arrive at (with $\nabla_j\Psi_t(\mathbf{x}_t^*)$ denoting j-th entry of gradient)

$$\begin{aligned}
&\Big(\mathbb{E}[\nabla\Psi_t(\mathbf{x}_t^*)] + \mathbf{A}^\top\boldsymbol{\lambda}^*\Big)^\top \mathbb{E}[(\mathbf{x}_t - \mathbf{x}_t^*)] \\
=& -\mathbb{E}\Big[\nabla_j\Psi_t(\mathbf{x}_t^*)(x_t^j)^* - \sum_{i\in\mathcal{I}}(\lambda^i)^*\mathbf{A}_{(i,j)}(x_t^j)^*\Big] \\
\overset{(a)}{=}& \underbrace{-\mathbb{E}[\nabla_j\Psi_t(\mathbf{x}_t^*)(x_t^j)^*]}_{<0} - \sum_{i\in\mathcal{I}\backslash k}\underbrace{(\lambda^i)^*\mathbf{A}_{(i,j)}\mathbb{E}[(x_t^j)^*]}_{\geq 0} < 0
\end{aligned} \tag{4.38}$$

where (a) uses $(\lambda^k)^* = 0$; the first bracket follows from Assumption 2 since $\nabla_j\Psi_t(\mathbf{x}_t^*)$ is monotonically increasing and $\nabla_j\Psi_t(\mathbf{x}_t^*) \geq 0$, thus for $\mathbb{E}[(x_t^j)^*] > 0$ it follows $\mathbb{E}[\nabla_j\Psi_t(\mathbf{x}_t^*)] > 0$; and the second bracket follows that $\boldsymbol{\lambda}^* \geq \mathbf{0}$ and each column of $\mathbf{A}$ has at most one -1 and $\mathbf{A}_{(k,j)} = -1$. The proof is then complete since (4.38) contradicts (4.35a).

4.7.1 Proof of Lemma 14

Proof of Lipschitz continuity: Under Assumption 2, the primal objective $\Psi_t(\mathbf{x}_t)$ is σ-strongly convex, and the smooth constant of the dual function $\mathcal{D}_t(\boldsymbol{\lambda})$, or equivalently, the Lipschitz constant of gradient $\nabla\mathcal{D}_t(\boldsymbol{\lambda})$ directly follows from [12, Lemma II.2], which equals to $L_{\rm d} = \rho(\mathbf{A}^\top\mathbf{A})/\sigma$, with $\rho(\mathbf{A}^\top\mathbf{A})$ denoting the maximum eigenvalue of $\mathbf{A}^\top\mathbf{A}$. We omit the derivations of this result, and refer readers to that in [12].

Supporting lemmas for quadratic growth: To prove the quadratic growth property (4.27), we introduce an error bound, which describes the local property of the dual function $\mathcal{D}(\boldsymbol{\lambda})$.

Lemma 17. *[66, Lemma 2.3] Consider the dual function in* (4.10) *and the feasible set $\mathcal{X}$ in* (4.3d) *with only linear constraints. For any $\boldsymbol{\lambda}$ satisfying $\mathcal{D}(\boldsymbol{\lambda}) > -\infty$ and $\|\nabla\mathcal{D}(\boldsymbol{\lambda})\| \leq \delta$, we have*

$$\|\boldsymbol{\lambda}^* - \boldsymbol{\lambda}\| \leq \xi\|\nabla\mathcal{D}(\boldsymbol{\lambda})\| \tag{4.39}$$

where the scalar ξ depends on the matrix $\mathbf{A}$ as well the constants σ, $L_{\rm p}$ and $L_{\rm d}$ introduced in Assumption 2.

Lemma 17 states a *local* error bound for the dual function $\mathcal{D}(\boldsymbol{\lambda})$. The error bound is "local" since it holds only for $\boldsymbol{\lambda}$ close enough to the optimum $\boldsymbol{\lambda}^*$, i.e., $\|\nabla\mathcal{D}(\boldsymbol{\lambda})\| \leq \delta$. Following the arguments in [66] however, if the dual iterate $\boldsymbol{\lambda}$ is artificially confined to a compact set $\boldsymbol{\Lambda}$ such that $\|\boldsymbol{\lambda}\| \leq D$ with D denoting the radius of $\boldsymbol{\Lambda}$,[5] then for the case $\|\nabla\mathcal{D}(\boldsymbol{\lambda})\| \geq \delta$, the ratio

[5]Since the optimal multiplier is bounded per Assumption 3, one can safely find a large set $\boldsymbol{\Lambda}$ with radius D to project dual iterates during optimization.

$\|\boldsymbol{\lambda}^* - \boldsymbol{\lambda}\|/\|\nabla\mathcal{D}(\boldsymbol{\lambda})\| \leq D/\delta$, which implies the existence of ξ satisfying (4.39) for any $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$. Lemma 17 is important for establishing linear convergence rate without strong convexity [66]. Remarkably, we will show next that this error bound is also critical to characterize the steady-state behavior of our LA-SDG scheme.

Building upon Lemma 17, we next show that the ensemble dual function $\mathcal{D}(\boldsymbol{\lambda})$ also satisfies the so-termed Polyak-Lojasiewicz (PL) condition [77].

Lemma 18. *Under Assumption 2, the local error-bound in* (4.39) *implies the following PL condition, namely*

$$\mathcal{D}(\boldsymbol{\lambda}^*) - \mathcal{D}(\boldsymbol{\lambda}) \leq \frac{L_{\rm d}\xi^2}{2}\|\nabla\mathcal{D}(\boldsymbol{\lambda})\|^2 \tag{4.40}$$

where $L_{\rm d}$ is the Lipschitz constant of the dual gradient and ξ is as in (4.39).

Proof: Using the $L_{\rm d}$-smoothness of the dual function $\mathcal{D}(\boldsymbol{\lambda})$, we have for any $\boldsymbol{\lambda}$ and $\boldsymbol{\varphi} \in \mathbb{R}_+^I$ that

$$\mathcal{D}(\boldsymbol{\varphi}) \leq \mathcal{D}(\boldsymbol{\lambda}) - \langle\nabla\mathcal{D}(\boldsymbol{\varphi}), \boldsymbol{\lambda} - \boldsymbol{\varphi}\rangle + \frac{L_{\rm d}}{2}\|\boldsymbol{\lambda} - \boldsymbol{\varphi}\|^2. \tag{4.41}$$

Choosing $\boldsymbol{\varphi} = \boldsymbol{\lambda}^*$, and using Proposition 5 such that $\nabla\mathcal{D}(\boldsymbol{\lambda}^*) = \mathbf{0}$, we have

$$\mathcal{D}(\boldsymbol{\lambda}^*) \leq \mathcal{D}(\boldsymbol{\lambda}) + \frac{L_{\rm d}}{2}\|\boldsymbol{\lambda} - \boldsymbol{\lambda}^*\|^2 \overset{(a)}{\leq} \mathcal{D}(\boldsymbol{\lambda}) + \frac{L_{\rm d}\xi^2}{2}\|\nabla\mathcal{D}(\boldsymbol{\lambda})\|^2 \tag{4.42}$$

where inequality (a) uses the local error-bound in (4.39).

Proof of quadratic growth: The proof follows the main steps of that in [77]. Building upon Lemma 18, we next prove Lemma 14. Define a function of the dual variable $\boldsymbol{\lambda}$ as $g(\boldsymbol{\lambda}) := \sqrt{\mathcal{D}(\boldsymbol{\lambda}^*) - \mathcal{D}(\boldsymbol{\lambda})}$. With the PL condition in (4.40), and $\boldsymbol{\Lambda}^*$ denoting the set of optimal multipliers for (4.11), we have for any $\boldsymbol{\lambda} \notin \boldsymbol{\Lambda}^*$ that

$$\|\nabla g(\boldsymbol{\lambda})\|^2 = \frac{\|\nabla\mathcal{D}(\boldsymbol{\lambda})\|^2}{\mathcal{D}(\boldsymbol{\lambda}^*) - \mathcal{D}(\boldsymbol{\lambda})} \geq \frac{2}{L_{\rm d}\xi^2} \tag{4.43}$$

which implies that $\|\nabla g(\boldsymbol{\lambda})\| \geq \sqrt{2/(L_{\rm d}\xi^2)}$.

For any $\boldsymbol{\lambda}_0 \notin \boldsymbol{\Lambda}^*$, consider the following differential equation[6]

$$\begin{cases} \dfrac{\mathbf{d}\boldsymbol{\lambda}(\tau)}{\mathbf{d}\tau} = -\nabla g(\boldsymbol{\lambda}(t)) & \text{(4.44a)} \\ \boldsymbol{\lambda}(\tau = 0) = \boldsymbol{\lambda}_0 & \text{(4.44b)} \end{cases}$$

[6]The time index in the proof of Lemma 1 is not related to the online optimization process, but it is useful to find the structure of the dual function.

which describes the continuous trajectory of $\{\boldsymbol{\lambda}(\tau)\}$ starting from $\boldsymbol{\lambda}_0$ along the direction of $-\nabla g(\boldsymbol{\lambda}(\tau))$. By using $\|\nabla g(\boldsymbol{\lambda})\| \geq \sqrt{2/(L_{\mathrm{d}}\xi^2)}$, it follows that $\nabla g(\boldsymbol{\lambda})$ is bounded below; thus, the differential equation (4.44) guarantees that we sufficiently reduce the value of function $g(\boldsymbol{\lambda})$, and $\boldsymbol{\lambda}(\tau)$ will eventually reach $\boldsymbol{\Lambda}^*$.

In other words, there exists a time T such that $\boldsymbol{\lambda}(T) \in \boldsymbol{\Lambda}^*$. Formally, for $\tau > T$, we have

$$\begin{aligned} g(\boldsymbol{\lambda}_0) - g(\boldsymbol{\lambda}_\tau) &= \int_{\boldsymbol{\lambda}_\tau}^{\boldsymbol{\lambda}_0} \langle \nabla g(\boldsymbol{\lambda}), \mathbf{d}\boldsymbol{\lambda} \rangle \\ &= -\int_{\boldsymbol{\lambda}_0}^{\boldsymbol{\lambda}_\tau} \langle \nabla g(\boldsymbol{\lambda}), \mathbf{d}\boldsymbol{\lambda} \rangle = -\int_0^T \left\langle \nabla g(\boldsymbol{\lambda}), \frac{\mathbf{d}\boldsymbol{\lambda}(\tau)}{\mathbf{d}\tau} \right\rangle \mathbf{d}\tau \\ &= \int_0^T \|\nabla g(\boldsymbol{\lambda}(\tau))\|^2 \mathbf{d}\tau \geq \int_0^T \frac{2}{L_{\mathrm{d}}\xi^2} \mathbf{d}\tau = \frac{2T}{L_{\mathrm{d}}\xi^2}. \end{aligned} \tag{4.45}$$

Since $g(\boldsymbol{\lambda}) \geq 0,\ \forall \boldsymbol{\lambda}$, we have $T \leq g(\boldsymbol{\lambda}_0) L_{\mathrm{d}}\xi^2/2$, which implies that there exists a finite time T such that $\boldsymbol{\lambda}_\tau \in \boldsymbol{\Lambda}^*$. On the other hand, the path length of trajectory $\{\boldsymbol{\lambda}(\tau)\}$ will be longer than the projection distance between $\boldsymbol{\lambda}_\mathbf{0}$ and the closest point in $\boldsymbol{\Lambda}^*$ denoted as $\boldsymbol{\lambda}^*$, that is,

$$\int_0^T \left\| \frac{\mathbf{d}\boldsymbol{\lambda}(\tau)}{\mathbf{d}\tau} \right\| \mathbf{d}\tau = \int_0^T \|\nabla g(\boldsymbol{\lambda}(\tau))\| \, \mathbf{d}\tau \geq \|\boldsymbol{\lambda}_\mathbf{0} - \boldsymbol{\lambda}^*\| \tag{4.46}$$

and thus we have from (4.45) that

$$\begin{aligned} g(\boldsymbol{\lambda}_0) - g(\boldsymbol{\lambda}_\tau) &= \int_0^T \|\nabla g(\boldsymbol{\lambda}(\tau))\|^2 \mathbf{d}\tau \\ &\geq \int_0^T \|\nabla g(\boldsymbol{\lambda}(\tau))\| \sqrt{\frac{2}{L_{\mathrm{d}}\xi^2}} \mathbf{d}\tau \overset{(b)}{\geq} \sqrt{\frac{2}{L_{\mathrm{d}}\xi^2}} \|\boldsymbol{\lambda}_\mathbf{0} - \boldsymbol{\lambda}^*\| \end{aligned} \tag{4.47}$$

where (b) follows from (4.46). Choosing T such that $g(\boldsymbol{\lambda}_T) = 0$, we have

$$g(\boldsymbol{\lambda}_0) \geq \sqrt{\frac{2}{L_{\mathrm{d}}\xi^2}} \|\boldsymbol{\lambda}_\mathbf{0} - \boldsymbol{\lambda}^*\|. \tag{4.48}$$

Squaring both sides of (4.48), the proof is complete, since ϵ is defined as $\epsilon := 2/(L_{\mathrm{d}}\xi^2)$ and $\boldsymbol{\lambda}_\mathbf{0}$ can be any point outside the set of optimal multipliers.

4.7.2 Proof of Lemma 16

Since $\hat{\boldsymbol{\lambda}}_t$ converges to $\boldsymbol{\lambda}^*,\ \text{w.p.1}$ according to Theorem 6, there exists a finite time T_θ such that for $t > T_\theta$, we have $\|\boldsymbol{\lambda}^* - \hat{\boldsymbol{\lambda}}_t\| \leq \|\boldsymbol{\theta}\|$. In such case, it follows that $\tilde{\boldsymbol{\theta}}_t = \boldsymbol{\lambda}^* - \hat{\boldsymbol{\lambda}}_t + \boldsymbol{\theta} \geq \mathbf{0}$, since

$\boldsymbol{\theta} \geq \mathbf{0}$. Therefore, we have

$$\begin{aligned}\|\mathbf{q}_{t+1}-\tilde{\boldsymbol{\theta}}_t/\mu\|^2 &= \|[\mathbf{q}_t + \mathbf{A}\mathbf{x}_t + \mathbf{c}_t]^+ - [\tilde{\boldsymbol{\theta}}_t/\mu]^+\|^2 \\ &\overset{(a)}{\leq} \|\mathbf{q}_t + \mathbf{A}\mathbf{x}_t + \mathbf{c}_t - \tilde{\boldsymbol{\theta}}_t/\mu\|^2 \\ &\overset{(b)}{\leq} \|\mathbf{q}_t - \tilde{\boldsymbol{\theta}}_t/\mu\|^2 + 2(\mathbf{q}_t - \tilde{\boldsymbol{\theta}}_t/\mu)^\top(\mathbf{A}\mathbf{x}_t + \mathbf{c}_t) + M^2\end{aligned} \tag{4.49}$$

where (a) comes from the non-expansive property of the projection operator, and (b) is due to the upper bound M in Assumption 4.

The RHS of (4.49) can be upper bounded by

$$\begin{aligned}&\|\mathbf{q}_t - \tilde{\boldsymbol{\theta}}_t/\mu\|^2 + 2(\mathbf{q}_t - \tilde{\boldsymbol{\theta}}_t/\mu)^\top(\mathbf{A}\mathbf{x}_t + \mathbf{c}_t) + M^2 \\ &\overset{(c)}{=} \|\mathbf{q}_t - \tilde{\boldsymbol{\theta}}_t/\mu\|^2 + \frac{2}{\mu}\left(\boldsymbol{\gamma}_t - \boldsymbol{\lambda}^*\right)^\top \left(\mathbf{A}\mathbf{x}_t + \mathbf{c}_t\right) + M^2\end{aligned} \tag{4.50}$$

where (c) uses the definitions $\tilde{\boldsymbol{\theta}}_t := \boldsymbol{\lambda}^* - \hat{\boldsymbol{\lambda}}_t + \boldsymbol{\theta}$, and $\boldsymbol{\gamma}_t := \hat{\boldsymbol{\lambda}}_t + \mu\mathbf{q}_t - \boldsymbol{\theta}$. Since $\mathbf{A}\mathbf{x}_t + \mathbf{c}_t$ is the stochastic subgradient of the concave function $\mathcal{D}(\boldsymbol{\lambda})$ at $\boldsymbol{\lambda} = \boldsymbol{\gamma}_t$ [cf. (4.17a)], we have

$$\mathbb{E}\left[\left(\boldsymbol{\gamma}_t - \boldsymbol{\lambda}^*\right)^\top \left(\mathbf{A}\mathbf{x}_t + \mathbf{c}_t\right)\right] \leq \mathcal{D}(\boldsymbol{\gamma}_t) - \mathcal{D}(\boldsymbol{\lambda}^*). \tag{4.51}$$

Taking expectations on (4.49)-(4.50) over the random state $\mathbf{s}_t$ conditioned on $\mathbf{q}_t$ and using (4.51), we arrive at

$$\mathbb{E}\left[\|\mathbf{q}_{t+1}-\tilde{\boldsymbol{\theta}}_t/\mu\|^2\right] \leq \|\mathbf{q}_t - \tilde{\boldsymbol{\theta}}_t/\mu\|^2 + \frac{2}{\mu}\left(\mathcal{D}(\boldsymbol{\gamma}_t) - \mathcal{D}(\boldsymbol{\lambda}^*)\right) + M^2 \tag{4.52}$$

where we use the fact that $\mathcal{D}(\boldsymbol{\lambda}) := \mathbb{E}\left[\mathcal{D}_t(\boldsymbol{\lambda})\right]$ in (4.11). Using the quadratic growth property of $\mathcal{D}(\boldsymbol{\lambda})$ in (4.27) of Lemma 14, the recursion (4.52) further leads to

$$\begin{aligned}\mathbb{E}\left[\|\mathbf{q}_{t+1}-\tilde{\boldsymbol{\theta}}_t/\mu\|^2\right] &\leq \|\mathbf{q}_t - \tilde{\boldsymbol{\theta}}_t/\mu\|^2 - \frac{2\epsilon}{\mu}\|\boldsymbol{\gamma}_t - \boldsymbol{\lambda}^*\|^2 + M^2 \\ &\overset{(d)}{=} \|\mathbf{q}_t - \tilde{\boldsymbol{\theta}}_t/\mu\|^2 - 2\mu\epsilon\|\mathbf{q}_t - \tilde{\boldsymbol{\theta}}_t/\mu\|^2 + M^2\end{aligned} \tag{4.53}$$

where equality (d) uses the definitions $\tilde{\boldsymbol{\theta}}_t := \boldsymbol{\lambda}^* - \hat{\boldsymbol{\lambda}}_t + \boldsymbol{\theta}$ and $\boldsymbol{\gamma}_t := \hat{\boldsymbol{\lambda}}_t + \mu\mathbf{q}_t - \boldsymbol{\theta}$, implying that $\boldsymbol{\gamma}_t - \boldsymbol{\lambda}^* = \mu\mathbf{q}_t - \tilde{\boldsymbol{\theta}}_t$.

Now considering (cf. (4.53))

$$-2\mu\epsilon\|\mathbf{q}_t - \tilde{\boldsymbol{\theta}}_t/\mu\|^2 + M^2 \leq -2\sqrt{\mu}\|\mathbf{q}_t - \tilde{\boldsymbol{\theta}}_t/\mu\| + \mu \tag{4.54}$$

and plugging it back into (4.53) yields

$$\mathbb{E}\big[\|\mathbf{q}_{t+1} - \tilde{\boldsymbol{\theta}}_t/\mu\|^2\big] \leq \|\mathbf{q}_t - \tilde{\boldsymbol{\theta}}_t/\mu\|^2 - 2\sqrt{\mu}\|\mathbf{q}_t - \tilde{\boldsymbol{\theta}}_t/\mu\| + \mu$$
$$= \big(\|\mathbf{q}_t - \tilde{\boldsymbol{\theta}}_t/\mu\| - \sqrt{\mu}\big)^2. \tag{4.55}$$

By the convexity of $(\,\cdot\,)^2$, we further arrive at

$$\mathbb{E}\left[\|\mathbf{q}_{t+1} - \tilde{\boldsymbol{\theta}}_t/\mu\|\right]^2 \leq \mathbb{E}\big[\|\mathbf{q}_{t+1} - \tilde{\boldsymbol{\theta}}_t/\mu\|^2\big]$$
$$\leq \left(\|\mathbf{q}_t - \tilde{\boldsymbol{\theta}}_t/\mu\| - \sqrt{\mu}\right)^2 \tag{4.56}$$

which directly implies the argument (4.31) in the lemma. By checking Vieta's formulas for second-order equations, there exists $B = \Theta(\frac{1}{\sqrt{\mu}})$ such that for $\|\mathbf{q}_t - \tilde{\boldsymbol{\theta}}_t/\mu\| > B$, inequality (4.54) holds, and thus the lemma follows readily.

4.7.3 Proof of Theorem 7

Proof of (4.32a) **in Theorem 7:** Theorem 6 asserts that $\hat{\boldsymbol{\lambda}}_t$ eventually converges to the optimum $\boldsymbol{\lambda}^*$, w.p.1. Hence, there always exists a finite time T_ρ and an arbitrarily small ρ such that for $t > T_\rho$, it holds that $\|\boldsymbol{\lambda}^*/\mu - \hat{\boldsymbol{\lambda}}_t/\mu\| \leq \rho$. Using the definition $\tilde{\boldsymbol{\theta}}_t = \boldsymbol{\lambda}^* - \hat{\boldsymbol{\lambda}}_t + \boldsymbol{\theta}$, it then follows by the triangle inequality that

$$\Big|\|\mathbf{q}_t - \tilde{\boldsymbol{\theta}}_t/\mu\| - \|\mathbf{q}_t - \boldsymbol{\theta}/\mu\|\Big| \leq \|\boldsymbol{\lambda}^*/\mu - \hat{\boldsymbol{\lambda}}_t/\mu\| \leq \rho \tag{4.57}$$

which also holds for $\mathbf{q}_{t+1}$.

Using (4.57) and the conditional drift (4.31) in Lemma 16, for $t > T_\rho$ and $\|\mathbf{q}_t - \tilde{\boldsymbol{\theta}}_t/\mu\| > B = \Theta(1/\sqrt{\mu})$, it holds that

$$\mathbb{E}\left[\|\mathbf{q}_{t+1} - \boldsymbol{\theta}/\mu\| \,\Big|\mathbf{q}_t\right] \leq \mathbb{E}\left[\Big\|\mathbf{q}_{t+1} - \tilde{\boldsymbol{\theta}}_t/\mu\Big\| \,\Big|\mathbf{q}_t\right] + \rho$$
$$\leq \Big\|\mathbf{q}_t - \tilde{\boldsymbol{\theta}}_t/\mu\Big\| - \sqrt{\mu} + \rho \leq \|\mathbf{q}_t - \boldsymbol{\theta}/\mu\| - \sqrt{\mu} + 2\rho. \tag{4.58}$$

Choosing ρ such that $\sqrt{\tilde{\mu}} := \sqrt{\mu} - 2\rho < 0$, then for $t > T_\rho$ and $\|\mathbf{q}_t - \boldsymbol{\theta}/\mu\| > \tilde{B} := B + \rho = \Theta(1/\sqrt{\mu})$, we have

$$\mathbb{E}\left[\|\mathbf{q}_{t+1} - \boldsymbol{\theta}/\mu\| \,\Big|\mathbf{q}_t\right] \leq \|\mathbf{q}_t - \boldsymbol{\theta}/\mu\| - \sqrt{\tilde{\mu}}. \tag{4.59}$$

Leveraging (4.59), we first show (4.32a) by constructing a super-martingale. Define the stochastic process a_t as

$$a_t := \|\mathbf{q}_t - \boldsymbol{\theta}/\mu\| \cdot \mathbb{1}\left\{\min_{\tau \leq t} \|\mathbf{q}_\tau - \boldsymbol{\theta}/\mu\| > \tilde{B}\right\}, \forall t \tag{4.60a}$$

and likewise the stochastic process b_t as

$$b_t := \sqrt{\tilde{\mu}} \cdot \mathbb{1}\left\{\min_{\tau \leq t} \|\mathbf{q}_\tau - \boldsymbol{\theta}/\mu\| > \tilde{B}\right\}, \forall t. \tag{4.60b}$$

Clearly, a_t tracks the distance between $\mathbf{q}_t$ and $\boldsymbol{\theta}/\mu$ until the distance becomes smaller than $\tilde{B}$ for the first time; and b_t stops until $\|\mathbf{q}_t - \boldsymbol{\theta}/\mu\| \leq \tilde{B}$ for the first time as well.

With the definitions of a_t and b_t, one can easily show that the recursion (4.59) implies

$$\mathbb{E}\left[a_{t+1}|\mathcal{F}_t\right] \leq a_t - b_t \tag{4.61}$$

where $\mathcal{F}_t$ is the so-termed sigma algebra measuring the history of two processes. As a_t and b_t are both nonnegative, (4.61) allows us to apply the super-martingale convergence theorem [81, Theorem E7.4], which almost surely establishes that: (i) the sequence a_t converges to a limit; and (ii) the summation $\sum_{t=1}^{\infty} b_t < \infty$. Note that (ii) implies that $\lim_{t\to\infty} b_t = 0$, w.p.1. Since $\sqrt{\tilde{\mu}} > 0$, it follows that the indicator function of b_t eventually becomes null and thus

$$\liminf_{t\to\infty} \; \|\mathbf{q}_t - \boldsymbol{\theta}/\mu\| \leq \tilde{B}, \;\; \text{w.p.1} \tag{4.62}$$

which establishes that $\mathbf{q}_t$ will eventually visit and then hover around a neighborhood of the reference point $\boldsymbol{\theta}/\mu$.

Proof of (4.32b) **in Theorem 7:** In complement to the sample-path result in (4.62), we next derive (4.32b), which captures the long-term queue lengths averaged over all sample paths.

Similar to (4.49), we have

$$\begin{aligned}&\|\mathbf{q}_{t+1} - \boldsymbol{\lambda}^*/\mu\|^2 \leq \\ &\qquad \|\mathbf{q}_t - \boldsymbol{\lambda}^*/\mu\|^2 + 2(\mathbf{q}_t - \boldsymbol{\lambda}^*/\mu)^\top(\mathbf{A}\mathbf{x}_t + \mathbf{c}_t) + M^2.\end{aligned} \tag{4.63}$$

Using the definition $\boldsymbol{\gamma}_t := \hat{\boldsymbol{\lambda}}_t + \mu\mathbf{q}_t - \boldsymbol{\theta}$, (4.64) can be written as

$$\|\mathbf{q}_{t+1} - \boldsymbol{\lambda}^*/\mu\|^2 \leq \|\mathbf{q}_t - \boldsymbol{\lambda}^*/\mu\|^2 \tag{4.64}$$

$$+\frac{2}{\mu}(\boldsymbol{\gamma}_t-\boldsymbol{\lambda}^*)^\top(\mathbf{A}\mathbf{x}_t+\mathbf{c}_t)+\frac{2}{\mu}(\boldsymbol{\theta}-\hat{\boldsymbol{\lambda}}_t)^\top(\mathbf{A}\mathbf{x}_t+\mathbf{c}_t)+M^2.$$

Defining the Lyapunov drift as $\Delta(\mathbf{q}_t):=\frac{1}{2}(\|\mathbf{q}_{t+1}-\boldsymbol{\lambda}^*/\mu\|^2-\|\mathbf{q}_t-\boldsymbol{\lambda}^*/\mu\|^2)$ and taking expectations on (4.64) over $\mathbf{s}_t$ conditioned on $\mathbf{q}_t$, we have

$$\begin{aligned}\mu\mathbb{E}\left[\Delta(\mathbf{q}_t)\right] &\le \mathbb{E}\Big[(\boldsymbol{\gamma}_t-\boldsymbol{\lambda}^*)^\top(\mathbf{A}\mathbf{x}_t+\mathbf{c}_t)\Big]\\ &\qquad+\mathbb{E}\Big[(\boldsymbol{\theta}-\hat{\boldsymbol{\lambda}}_t)^\top(\mathbf{A}\mathbf{x}_t+\mathbf{c}_t)\Big]+\mu M^2/2\\ &\overset{(b)}{\le} \mathcal{D}(\boldsymbol{\gamma}_t)-\mathcal{D}(\boldsymbol{\lambda}^*)+\mathbb{E}\Big[(\boldsymbol{\theta}-\hat{\boldsymbol{\lambda}}_t)^\top(\mathbf{A}\mathbf{x}_t+\mathbf{c}_t)\Big]+\mu M^2/2\end{aligned}\tag{4.65}$$

where (b) follows from (4.51).

Summing both sides over $t=1,\dots,T$, taking expectations over all possible $\mathbf{q}_t$, and dividing both sides by T, we arrive at

$$\begin{aligned}&\frac{\mu}{2T}\left(\mathbb{E}\left[\|\mathbf{q}_{T+1}-\boldsymbol{\lambda}^*/\mu\|^2\right]-\mathbb{E}\left[\|\mathbf{q}_1-\boldsymbol{\lambda}^*/\mu\|^2\right]\right)\le \qquad (4.66)\\ &\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\mathcal{D}(\boldsymbol{\gamma}_t)]-\mathcal{D}(\boldsymbol{\lambda}^*)+\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\Big[(\boldsymbol{\theta}-\hat{\boldsymbol{\lambda}}_t)^\top(\mathbf{A}\mathbf{x}_t+\mathbf{c}_t)\Big]+\frac{\mu M^2}{2}.\end{aligned}$$

First, it is easy to show that

$$\begin{aligned}&\lim_{T\to\infty}\frac{\mu}{2T}\left(\mathbb{E}\left[\|\mathbf{q}_{T+1}-\boldsymbol{\lambda}^*/\mu\|^2\right]-\mathbb{E}\left[\|\mathbf{q}_1-\boldsymbol{\lambda}^*/\mu\|^2\right]\right)\\ &\overset{(c)}{\ge}-\lim_{T\to\infty}\frac{\mu}{2T}\mathbb{E}\left[\|\mathbf{q}_1-\boldsymbol{\lambda}^*/\mu\|^2\right]\overset{(d)}{=}0\end{aligned}\tag{4.67}$$

where (c) holds since $\|\mathbf{q}_{T+1}-\boldsymbol{\lambda}^*/\mu\|^2\ge 0$, and (d) follows from the boundedness of $\|\mathbf{q}_1-\boldsymbol{\lambda}^*/\mu\|^2$.

We next argue that the following equality holds

$$\lim_{T\to\infty}(1/T)\ \textstyle\sum_{t=1}^{T}\mathbb{E}\left[(\boldsymbol{\theta}-\hat{\boldsymbol{\lambda}}_t)^\top(\mathbf{A}\mathbf{x}_t+\mathbf{c}_t)\right]=\mathcal{O}(\mu).\tag{4.68}$$

Rearranging terms in (4.68) leads to

$$\begin{aligned}&\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[(\boldsymbol{\theta}-\hat{\boldsymbol{\lambda}}_t)^\top(\mathbf{A}\mathbf{x}_t+\mathbf{c}_t)\right] \qquad (4.69)\\ &=\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\Big(\boldsymbol{\theta}-\boldsymbol{\lambda}^*+(\boldsymbol{\lambda}^*-\boldsymbol{\theta}-\hat{\boldsymbol{\lambda}}_t+\boldsymbol{\theta})\Big)^\top(\mathbf{A}\mathbf{x}_t+\mathbf{c}_t)\right].\end{aligned}$$

Since $\hat{\boldsymbol{\lambda}}_t$ converges to $\boldsymbol{\lambda}^*$ w.p.1 according to Theorem 6, there always exists a finite time T_ρ such that for $t > T_\rho$, which implies that $\|\boldsymbol{\lambda}^* - \boldsymbol{\theta} - (\hat{\boldsymbol{\lambda}}_t - \boldsymbol{\theta})\| \leq \rho,$ w.p.1. Hence, together with the Cauchy-Schwarz inequality, we have

$$\begin{aligned}&(\boldsymbol{\lambda}^* - \boldsymbol{\theta} - \hat{\boldsymbol{\lambda}}_t + \boldsymbol{\theta})^\top (\mathbf{A}\mathbf{x}_t + \mathbf{c}_t)\\ \leq& \|\boldsymbol{\lambda}^* - \boldsymbol{\theta} - (\hat{\boldsymbol{\lambda}}_t - \boldsymbol{\theta})\| \|\mathbf{A}\mathbf{x}_t + \mathbf{c}_t\| \overset{(d)}{\leq} \rho M = \mathcal{O}(\rho)\end{aligned} \tag{4.70}$$

where (d) follows since $T_\rho < \infty$ and constant M is as in Assumption 4. Plugging (4.70) into (4.69), it follows that

$$\begin{aligned}&\lim_{T\to\infty} (1/T) \textstyle\sum_{t=1}^T \mathbb{E}\left[(\boldsymbol{\theta} - \hat{\boldsymbol{\lambda}}_t)^\top (\mathbf{A}\mathbf{x}_t + \mathbf{c}_t)\right] \\ \leq& \lim_{T\to\infty} (1/T) \textstyle\sum_{t=1}^T \mathbb{E}\left[(\boldsymbol{\theta} - \boldsymbol{\lambda}^*)^\top (\mathbf{A}\mathbf{x}_t + \mathbf{c}_t)\right] + \mathcal{O}(\rho) \\ \overset{(e)}{\leq}& \|\boldsymbol{\lambda}^* - \boldsymbol{\theta}\| \cdot \left\| \lim_{T\to\infty} (1/T) \textstyle\sum_{t=1}^T \mathbb{E}\left[-\mathbf{A}\mathbf{x}_t - \mathbf{c}_t\right] \right\| + \mathcal{O}(\rho)\end{aligned} \tag{4.71}$$

where (e) simply follows from the Cauchy-Schwarz inequality.

Building upon (4.59), one can follow the arguments in [70, Theorem 4] to show that there exist constants $D_1 = \Theta(1/\mu)$, and $D_2 = \Theta(\sqrt{\mu})$, for any d, to obtain a large deviation bound as

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^T \mathbb{P}\left(\|\mathbf{q}_t - \boldsymbol{\theta}/\mu\| > \tilde{B} + d\right) \leq D_1 e^{-D_2 d} \tag{4.72}$$

where $\tilde{B} = \Theta(1/\sqrt{\mu})$ as in (4.59). Intuitively speaking, (4.72) upper bounds the probability that the steady-state $\mathbf{q}_t$ deviates from $\boldsymbol{\theta}/\mu$, and (4.72) implies that the probability that $q_t^i > \theta/\mu + \tilde{B} + d, \ \forall i$ is exponentially decreasing in $D_2 d$.

Using the large deviation bound in (4.72), it follows that

$$\begin{aligned}\mathbf{0} &\overset{(f)}{\leq} \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^T \mathbb{E}[-\mathbf{A}\mathbf{x}_t - \mathbf{c}_t] \\ &\overset{(g)}{\leq} \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^T \mathbf{1} \cdot M \,\mathbb{P}\left(\mathbf{q}_t < M\right) \overset{(h)}{\leq} \mathbf{1} \cdot M D_1 e^{-D_2(\boldsymbol{\theta}/\mu - \tilde{B} - M)}\end{aligned} \tag{4.73}$$

where (f) holds because taking expectation in (4.72) over all d implies that the expected queue length is finite [cf. (4.3c)], which implies the necessary condition in (4.5); (g) follows from [69, Lemma 4] which establishes that negative accumulated service residual $\sum_{t=1}^T \mathbb{E}[\mathbf{A}\mathbf{x}_t + \mathbf{c}_t]$ may happen only when $\mathbf{q}_t < M$ and the maximum value is bounded by $\|\mathbf{A}\mathbf{x}_t + \mathbf{c}_t\| \leq M$ in Assumption 4; and (h) uses the bound in (4.72) by choosing $d = \boldsymbol{\theta}/\mu - \tilde{B} - M$.

Setting $\boldsymbol{\theta} = \sqrt{\mu}\log^2(\mu)$ in (4.73), there exists a sufficiently small μ such that $-D_2\big(\log^2(\mu)/\sqrt{\mu} - \tilde{B} - M\big) \leq 2\log(\mu)$. Together with (4.73) and $D_1 = \Theta(1/\mu)$, the latter implies that

$$\left\| \lim_{T\to\infty} (1/T) \textstyle\sum_{t=1}^{T} \mathbb{E}[-\mathbf{A}\mathbf{x}_t - \mathbf{c}_t] \right\| \leq \|\mathbf{1} \cdot M D_1 \mu^2\| = \mathcal{O}(\mu). \tag{4.74}$$

Plugging (4.74) into (4.71), setting $\rho = \mathbf{o}(\mu)$ in (4.71), and using $\|\boldsymbol{\lambda}^* - \boldsymbol{\theta}\| = \mathcal{O}(1)$, we arrive at (4.68).

Letting $T \to \infty$ in (4.66), it follows from (4.67) and (4.68) that

$$\begin{aligned} 0 &\leq \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[\mathcal{D}(\boldsymbol{\gamma}_t)] - \mathcal{D}(\boldsymbol{\lambda}^*) + \mathcal{O}(\mu) + \frac{\mu M^2}{2} \\ &\overset{(h)}{\leq} \mathcal{D}\left(\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[\boldsymbol{\gamma}_t]\right) - \mathcal{D}(\boldsymbol{\lambda}^*) + \mathcal{O}(\mu) + \frac{\mu M^2}{2}. \end{aligned} \tag{4.75}$$

where inequality (h) uses the concavity of the dual function $\mathcal{D}(\boldsymbol{\lambda})$. Defining $\boldsymbol{\varphi} := \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[\boldsymbol{\gamma}_t]$, and using $\mathcal{D}(\boldsymbol{\lambda}^*) - \mathcal{D}(\boldsymbol{\varphi}) \geq \frac{\epsilon}{2}\|\boldsymbol{\lambda}^* - \boldsymbol{\varphi}\|^2$ in Lemma 14, (4.75) implies that

$$\|\boldsymbol{\lambda}^* - \boldsymbol{\varphi}\|^2 \leq \frac{2}{\epsilon}\Big(\mathcal{D}(\boldsymbol{\lambda}^*) - \mathcal{D}(\boldsymbol{\varphi})\Big) \leq \mathcal{O}(\mu) + \frac{\mu M^2}{\epsilon} \overset{(i)}{=} \mathcal{O}(\mu) \tag{4.76}$$

where (i) follows since constants M and ϵ are independent of μ. From (4.76), we can further conclude that $\|\boldsymbol{\lambda}^* - \boldsymbol{\varphi}\| = \mathcal{O}(\sqrt{\mu})$.

Recalling the definition $\boldsymbol{\gamma}_t := \hat{\boldsymbol{\lambda}}_t + \mu\mathbf{q}_t - \boldsymbol{\theta}$, we have that

$$\begin{aligned} \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[\frac{\boldsymbol{\gamma}_t}{\mu}\right] - \frac{\boldsymbol{\lambda}^*}{\mu} &= \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[\mathbf{q}_t + \frac{\hat{\boldsymbol{\lambda}}_t}{\mu}\right] - \frac{\boldsymbol{\lambda}^*}{\mu} - \frac{\boldsymbol{\theta}}{\mu} \\ &\overset{(j)}{=} \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[\mathbf{q}_t] - \frac{\boldsymbol{\theta}}{\mu} \overset{(k)}{\leq} \frac{1}{\mu}\|\boldsymbol{\lambda}^* - \boldsymbol{\varphi}\| = \mathcal{O}\left(\frac{1}{\sqrt{\mu}}\right) \end{aligned} \tag{4.77}$$

where (j) follows from the convergence of $\hat{\boldsymbol{\lambda}}_t$ in Theorem 6, and inequality (k) uses the definition of $\boldsymbol{\varphi}$ and $\boldsymbol{\varphi} - \boldsymbol{\lambda}^* \leq \|\boldsymbol{\lambda}^* - \boldsymbol{\varphi}\|$. Recalling that $\boldsymbol{\theta} = \sqrt{\mu}\log^2(\mu)$ in (4.74) completes the proof.

4.7.4 Proof of Theorem 8

Defining the Lyapunov drift as $\Delta(\mathbf{q}_t) := \frac{1}{2}(\|\mathbf{q}_{t+1}\|^2 - \|\mathbf{q}_t\|^2)$, and squaring the queue update, we obtain

$$\begin{aligned}\|\mathbf{q}_{t+1}\|^2 =& \|\mathbf{q}_t\|^2 + 2\mathbf{q}_t^\top(\mathbf{A}\mathbf{x}_t + \mathbf{c}_t) + \|\mathbf{A}\mathbf{x}_t + \mathbf{c}_t\|^2 \\ \overset{(a)}{\le}& \|\mathbf{q}_t\|^2 + 2\mathbf{q}_t^\top(\mathbf{A}\mathbf{x}_t + \mathbf{c}_t) + M^2 \end{aligned} \tag{4.78}$$

where (a) follows from the definition of M in Assumption 4. Multiplying by $\mu/2$ and adding $\Psi_t(\mathbf{x}_t)$, yields

$$\begin{aligned}\mu\Delta(\mathbf{q}_t) + \Psi_t(\mathbf{x}_t) \le& \Psi_t(\mathbf{x}_t) + \mu\mathbf{q}_t^\top(\mathbf{A}\mathbf{x}_t + \mathbf{c}_t) + \mu M^2/2 \\ \overset{(b)}{=}& \Psi_t(\mathbf{x}_t) + (\boldsymbol{\gamma}_t - \hat{\boldsymbol{\lambda}}_t + \boldsymbol{\theta})^\top(\mathbf{A}\mathbf{x}_t + \mathbf{c}_t) + \mu M^2/2 \\ \overset{(c)}{=}& \mathcal{L}_t(\mathbf{x}_t, \boldsymbol{\gamma}_t) + (\boldsymbol{\theta} - \hat{\boldsymbol{\lambda}}_t)^\top(\mathbf{A}\mathbf{x}_t + \mathbf{c}_t) + \mu M^2/2 \end{aligned} \tag{4.79}$$

where (b) uses the definition of $\boldsymbol{\gamma}_t$, and (c) is the definition of the instantaneous Lagrangian. Taking expectations on the both sides of (4.79) over $\mathbf{s}_t$ conditioned on $\mathbf{q}_t$, it holds that

$$\begin{aligned}&\mu\mathbb{E}\left[\Delta(\mathbf{q}_t)\middle|\mathbf{q}_t\right] + \mathbb{E}\left[\Psi_t(\mathbf{x}_t)\middle|\mathbf{q}_t\right] \\ \overset{(d)}{=}& \mathcal{D}(\boldsymbol{\gamma}_t) + \mathbb{E}\left[(\boldsymbol{\theta} - \hat{\boldsymbol{\lambda}}_t)^\top(\mathbf{A}\mathbf{x}_t + \mathbf{c}_t)\middle|\mathbf{q}_t\right] + \mu M^2/2 \\ \overset{(e)}{\le}& \Psi^* + \mathbb{E}\left[(\boldsymbol{\theta} - \hat{\boldsymbol{\lambda}}_t)^\top(\mathbf{A}\mathbf{x}_t + \mathbf{c}_t)\middle|\mathbf{q}_t\right] + \mu M^2/2 \end{aligned} \tag{4.80}$$

where (d) follows from the definition of the dual function (4.10), while (e) uses the weak duality that $\mathcal{D}(\boldsymbol{\gamma}_t) \le \tilde{\Psi}^*$, and the fact that $\tilde{\Psi}^* \le \Psi^*$ (cf. the discussion after (4.6)).

Taking expectations on both sides of (4.80) over all possible $\mathbf{q}_t$, summing over $t = 1, \ldots, T$, dividing by T, and letting $T \to \infty$, we arrive at

$$\begin{aligned}&\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\Psi_t(\mathbf{x}_t)\right] \\ \overset{(f)}{\le}& \Psi^* + \lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[(\boldsymbol{\theta} - \hat{\boldsymbol{\lambda}}_t)^\top(\mathbf{A}\mathbf{x}_t + \mathbf{c}_t)\right] + \frac{\mu M^2}{2} + \lim_{T\to\infty}\frac{\mu\|\mathbf{q}_1\|^2}{2T} \\ \overset{(g)}{\le}& \Psi^* + \lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[(\boldsymbol{\theta} - \hat{\boldsymbol{\lambda}}_t)^\top(\mathbf{A}\mathbf{x}_t + \mathbf{c}_t)\right] + \frac{\mu M^2}{2} \end{aligned} \tag{4.81}$$

where (f) comes from $\mathbb{E}[\|\mathbf{q}_{T+1}\|^2] \geq 0$, and (g) follows because $\|\mathbf{q}_1\|$ is bounded. One can follow the derivations in (4.69)-(4.74) to show (4.68), which is the second term in the RHS of (4.81). Therefore, we have from (4.81) that

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\left[\Psi_t(\mathbf{x}_t)\right] \leq \Psi^* + \mathcal{O}(\mu) + \frac{\mu M^2}{2} \tag{4.82}$$

which completes the proof.

Chapter 5

Online learning viewpoint of network resource management

5.1 Introduction

Online convex optimization (OCO) is an emerging methodology for sequential inference with well documented merits especially when the sequence of convex costs varies in an unknown and possibly adversarial manner [185, 65, 141].

5.1.1 Prior art

Starting from the seminal papers [185] and [65], most of the early works evaluate OCO algorithms with a *static regret*, which measures the difference of costs (a.k.a. losses) between the online solution and the overall best static solution in hindsight. If an algorithm incurs static regret that increases sub-linearly with time, then its performance loss averaged over an infinite time horizon goes to zero; see also [141, 64], and references therein.

However, static regret is not a comprehensive performance metric [15]. Take online parameter estimation as an example. When the true parameter varies over time, a static benchmark (time-invariant estimator) itself often performs poorly so that achieving sub-linear static regret is no longer attractive. Recent works [15, 62, 71, 112] extend the analysis of static regret to that of *dynamic regret*, where the performance of an OCO algorithm is benchmarked by the best dynamic solution with a-priori information on the one-slot-ahead cost function. Sub-linear dynamic regret is proved to be possible, if the dynamic environment changes slow enough for the accumulated variation of either costs or per-slot minimizers to be sub-linearly increasing with respect to the

Table 5.1: A summary of related works on discrete time OCO

Reference	Type of benchmark	Long-term constraint	Adversarial constraint
[185]	Static and dynamic	No	No
[65, 141, 64]	Static	No	No
[15, 112, 62, 71, 24, 6]	Dynamic	No	No
[106, 174, 82, 168]	Static	Yes	No
[140]	Dynamic	Yes	No
This work	Dynamic	Yes	Yes

time horizon. When the per-slot costs depend on previous decisions, the so-termed competitive difference can be employed as an alternative of the static regret [24, 6].

The aforementioned works [15, 24, 6, 62, 71, 112] deal with dynamic costs focusing on problems with time-invariant constraints that must be strictly satisfied, but do not allow for instantaneous violations of the constraints. The *long-term* effect of such instantaneous violations was studied in [106], where an online algorithm with sub-linear static regret and sub-linear accumulated constraint violation was also developed. The regret bounds in [106] have been improved in the discrete time domain [174] and the continuous time domain [125], respectively. Decentralized optimization with consensus constraints, as a special case of having long-term but time-invariant constraints, has been studied in [82, 168, 140]. Nevertheless, [106, 125, 82, 174, 168, 140] do not deal with OCO under time-varying adversarial constraints.

5.1.2 Our contributions

In this context, the present paper considers OCO with time-varying constraints that must be satisfied in the long term. Under this setting, the learner first takes an action without knowing a-priori either the adversarial cost or the time-varying constraint, which are revealed by the nature subsequently. Its performance is evaluated by: i) *dynamic regret* that is the optimality loss relative to a sequence of instantaneous minimizers with known costs and constraints; and, ii) *dynamic fit* that accumulates constraint violations incurred by the online learner due to the lack of knowledge about future constraints. We compare the OCO setting here with the existing ones in Table 5.1.

We further introduce a modified online saddle-point (MOSP) method in this novel OCO framework, where the learner deals with time-varying costs as well as time-varying but long-term constraints. We analytically establish that MOSP simultaneously achieves sub-linear dynamic regret and fit, provided that the accumulated variations of both minimizers and constraints grow sub-linearly with time. This result provides valuable insights for OCO with long-term constraints: *When the dynamic environment comprising both costs and constraints does not change on average,* *and the order of variations is known, the online decisions provided by MOSP are as good as the best dynamic solution over a long time horizon.*

To demonstrate the impact of these results, we further apply the proposed MOSP approach to a dynamic network resource allocation task, where online management of resources is sought without knowing future network states. Existing algorithms include first- and second-order methods in the dual domain [99, 173, 55, 12, 170, 35], which are tailored for time-invariant deterministic formulations. To capture the temporal variations of network resources, stochastic formulation of network resource allocation has been extensively pursued since the seminal work of [162]; see also the celebrated stochastic dual gradient method in [116, 108]. These stochastic approximation-based approaches assume that the time-varying costs are i.i.d. or generally samples from a stationary ergodic stochastic process [118, 49]. However, performance of most stochastic schemes is established in an asymptotic sense, considering the ensemble of per slot averages or infinite samples across time. Clearly, stationarity may not hold in practice, especially when the stochastic process involves human participation. Inheriting merits of the OCO framework, the proposed MOSP approach operates in a fully online mode with only information at previous time slots, and further admits finite-sample performance analysis under a sequence of deterministic, or even adversarial costs and constraints within a budget of temporal variation.

Relative to existing works, the main contributions of this paper are summarized as follows.

c1) We generalize the standard OCO framework with only adversarial costs in [185, 65, 141, 64] to account for both adversarial costs and constraints. Different from the regret analysis in [106, 82, 125, 174, 168], performance here is established relative to the best dynamic benchmark, via metrics that we term dynamic regret and fit.

c2) We develop a MOSP algorithm to tackle this novel OCO problem, and analytically establish that MOSP yields simultaneously sub-linear dynamic regret and fit, provided that the accumulated variations of per-slot minimizers and constraints are known to grow sub-linearly with time.

c3) Our novel approach is tailored for online resource allocation tasks, where MOSP is compared with the popular stochastic dual gradient approach. Relative to the latter, MOSP remains operational in a broader practical setting without probabilistic assumptions. Numerical tests demonstrate the gain of MOSP over existing alternatives.

5.2 OCO with long-term time-varying constraints

In this section, we introduce the generic OCO formulation with long-term time-varying constraints, along with pertinent metrics to evaluate an OCO algorithm.

5.2.1 Problem formulation

We begin with the classical OCO setting, where constraints are time-invariant and must be strictly satisfied. OCO can be viewed as a repeated game between a learner and nature [141, 185, 65]. Consider that time is discrete and indexed by t. Per slot t, a learner selects an action $\mathbf{x}_t$ from a convex set $\mathcal{X} \subseteq \mathbb{R}^I$, and subsequently nature chooses a (possibly adversarial) loss function $f_t(\,\cdot\,) : \mathbb{R}^I \rightarrow \mathbb{R}$ through which the learner incurs a loss $f_t(\mathbf{x}_t)$. The convex set $\mathcal{X}$ is a-priori known and fixed over the entire time horizon. Although this standard OCO setting is appealing to various applications such as online regression and classification [185, 65, 141], it does not account for potential variations of (possibly unknown) constraints, and does not deal with constraints that can possibly be satisfied in the long term rather than a slot-by-slot basis. Online optimization with time-varying and long-term constraints is motivated for applications such as navigation, tracking, localization, and resource allocation [116, 125, 108, 25]. Taking resource allocation as an example, time-varying long-term constraints are usually imposed to tolerate instantaneous violations when available resources cannot satisfy user requests, and hence allow flexible adaptation of online decisions to temporal variations of resource availability.

To broaden the applicability of OCO to these scenarios, we consider that per slot t, a learner selects an action $\mathbf{x}_t$ from a known and fixed convex set $\mathcal{X} \subseteq \mathbb{R}^I$, and then nature reveals not only a loss function $f_t(\cdot) : \mathbb{R}^I \rightarrow \mathbb{R}$ but also a time-varying (possibly adversarial) penalty function $\mathbf{g}_t(\cdot) : \mathbb{R}^I \rightarrow \mathbb{R}^I$. This function leads to a time-varying constraint $\mathbf{g}_t(\mathbf{x}) \leq \mathbf{0}$, which is driven by the unknown dynamics in various applications, e.g., on-demand data request arrivals in resource allocation. Different from the known and fixed set $\mathcal{X}$, the time-varying constraint $\mathbf{g}_t(\mathbf{x}) \leq \mathbf{0}$ can vary arbitrarily or even adversarially from slot to slot. It is revealed after the learner makes her/his decision, and is hence hard to be satisfied in every time slot. This is indeed a major difference when comparing to settings where the time-varying constraints are revealed before making decisions. Therefore, the goal in this context is to find a sequence of online solutions $\{\mathbf{x}_t \in \mathcal{X}\}$ that minimize the aggregate loss, and ensures that the constraints $\{\mathbf{g}_t(\mathbf{x}_t) \leq \mathbf{0}\}$ are satisfied in the long-term on average. Specifically, we aim to solve the following online optimization problem

$$\min_{\{\mathbf{x}_t\in\mathcal{X},\forall t\}} \sum_{t=1}^{T} f_t(\mathbf{x}_t) \tag{5.1a}$$

$$\text{s. to} \sum_{t=1}^{T} \mathbf{g}_t(\mathbf{x}_t) \leq \mathbf{0} \tag{5.1b}$$

where T is the time horizon, $\mathbf{x}_t \in \mathbb{R}^I$ is the decision variable, f_t is the cost function, $\mathbf{g}_t := [g_t^1, \dots, g_t^I]^\top$ denotes the constraint function with ith entry $g_t^i : \mathbb{R}^I \to \mathbb{R}$, and $\mathcal{X} \in \mathbb{R}^I$ is a convex set. The formulation (5.1) extends the standard OCO framework to accommodate adversarial time-varying constraints that must be satisfied in the long term. Complemented by algorithm development and performance analysis to be carried in the following sections, the main contribution of the present paper is incorporation of long-term and time-varying constraints to markedly broaden the scope of OCO.

5.2.2 Performance and feasibility metrics

Regarding performance of online decisions $\{\mathbf{x}_t\}_{t=1}^T$, static regret is adopted as a metric by standard OCO schemes, under time-invariant and strictly satisfied constraints. The static regret measures the difference between the online loss of an OCO algorithm and that of the best fixed solution in hindsight [185, 65, 141]. Extending the definition of static regret over T slots to accommodate time-varying constraints, it can be written as

$$\mathrm{Reg}_T^{\mathrm{s}} := \sum_{t=1}^{T} f_t(\mathbf{x}_t) - \sum_{t=1}^{T} f_t(\mathbf{x}^*) \tag{5.2}$$

where the best static solution $\mathbf{x}^*$ is obtained as

$$\mathbf{x}^* \in \arg\min_{\mathbf{x}\in\mathcal{X}} \sum_{t=1}^{T} f_t(\mathbf{x}) \;\; \text{s. to}\; \mathbf{g}_t(\mathbf{x}) \leq \mathbf{0},\; \forall t. \tag{5.3}$$

A desirable OCO algorithm in this case is the one yielding a sub-linear regret [106, 174], meaning $\mathrm{Reg}_T^{\mathrm{s}} = \mathbf{o}(T)$. Consequently, $\lim_{T\to\infty} \mathrm{Reg}_T^{\mathrm{s}}/T = 0$ implies that the algorithm is "on average" no-regret, or in other words, not worse asymptotically than the best fixed solution $\mathbf{x}^*$. Though widely used in various OCO applications, the aforementioned *static regret* has several limitations. For instance, it fails to capture the convergence of online decisions $\{\mathbf{x}_t\}$ relative to the fixed best solution $\mathbf{x}^*$, since small regrets can be also achieved by having $f_t(\mathbf{x}_t)$ oscillate around $f_t(\mathbf{x}^*)$ [125]. Even when the sub-linear static regret does imply $\mathbf{x}_t$ approaching $\mathbf{x}^*$, targeting a rather coarse benchmark may be less useful especially in dynamic settings. For instance, [15, Example 2] shows that the gap between the best static and the best dynamic benchmark can be as large as $\mathcal{O}(T)$. Furthermore, since the time-varying constraint $\mathbf{g}_t(\mathbf{x}_t) \leq \mathbf{0}$ is not observed before making a decision $\mathbf{x}_t$, its feasibility can not be checked instantaneously.

In response to the quest for improved benchmarks in this dynamic setup, two metrics are considered here: *dynamic regret* and *dynamic fit*. The notion of dynamic regret (also termed tracking regret or adaptive regret) has been recently introduced in [15, 62, 71, 112] to offer a competitive performance measure of OCO algorithms under time-invariant constraints. We adopt it in the setting of (5.1) by incorporating time-varying constraints

$$\mathrm{Reg}_T^{\mathrm{d}} := \sum_{t=1}^{T} f_t(\mathbf{x}_t) - \sum_{t=1}^{T} f_t(\mathbf{x}_t^*) \tag{5.4}$$

where the benchmark is now formed via a sequence of best dynamic solutions $\{\mathbf{x}_t^*\}$ for the instantaneous cost minimization problem subject to the instantaneous constraint, namely

$$\mathbf{x}_t^* \in \arg\min_{\mathbf{x}\in\mathcal{X}} \; f_t(\mathbf{x}) \quad \text{s. to } \mathbf{g}_t(\mathbf{x}) \leq \mathbf{0}. \tag{5.5}$$

Clearly, the dynamic regret is always larger than the static regret in (5.2), i.e., $\mathrm{Reg}_T^{\mathrm{s}} \leq \mathrm{Reg}_T^{\mathrm{d}}$, because $\sum_{t=1}^{T} f_t(\mathbf{x}^*)$ is always no smaller than $\sum_{t=1}^{T} f_t(\mathbf{x}_t^*)$ according to the definitions of $\mathbf{x}^*$ and $\mathbf{x}_t^*$. Hence, a sub-linear dynamic regret implies a sub-linear static regret, but not vice versa. The dynamic regret is suitable for cases where the goal is to track the time-varying solutions; e.g., AC power flow [42], and energy management policy [104].

To ensure feasibility of online decisions, the notion of *dynamic fit* is introduced to measure the accumulated violation of constraints; under time-invariant long-term constraints [106, 82] or under time-varying constraints [125]. It is defined as

$$\mathrm{Fit}_T^{\mathrm{d}} := \left\| \left[\sum_{t=1}^{T} \mathbf{g}_t(\mathbf{x}_t) \right]^+ \right\|. \tag{5.6}$$

Observe that the dynamic fit is zero if the accumulated violation $\sum_{t=1}^{T} \mathbf{g}_t(\mathbf{x}_t)$ is entry-wise less than zero. However, enforcing $\sum_{t=1}^{T} \mathbf{g}_t(\mathbf{x}_t) \leq \mathbf{0}$ is different from restricting $\mathbf{x}_t$ to meet $\mathbf{g}_t(\mathbf{x}_t) \leq \mathbf{0}$ in each and every slot. While the latter readily implies the former, the long-term (aggregate) constraint allows adaptation of online decisions to the environment dynamics; as a result, it is tolerable to have $\mathbf{g}_t(\mathbf{x}_t) \geq \mathbf{0}$ and $\mathbf{g}_{t+1}(\mathbf{x}_{t+1}) \leq \mathbf{0}$. Note that the definition of dynamic fit in (5.6) implicitly assumes that the instantaneous constraint violations can be compensated by the later strictly feasible decisions. When this is the case for resource allocation in power and cloud networks (see Section IV), extra modifications are required to account for other type of constraints, which go beyond the scope of the present paper.

An ideal algorithm in this broader OCO framework is the one that achieves both sub-linear dynamic regret and sub-linear dynamic fit. A sub-linear dynamic regret implies "no-regret" relative to the clairvoyant dynamic solution on the long-term average; i.e., $\lim_{T\to\infty} \mathrm{Reg}_T^{\mathrm{d}}/T = 0$; and a sub-linear dynamic fit indicates that the online strategy is also feasible on average; i.e., $\lim_{T\to\infty} \mathrm{Fit}_T^{\mathrm{d}}/T = 0$. Unfortunately, the sub-linear dynamic regret is not achievable in general, even under the special case of (5.1) where the time-varying constraint is absent [15]. For this reason, we aim at designing and analyzing an online strategy that generates a sequence $\{\mathbf{x}_t\}_{t=1}^{T}$ ensuring sub-linear dynamic regret and fit, under mild conditions that must be satisfied by the cost and constraint variations.

5.3 MOSP: Modified online saddle-point method

In this section, a modified online saddle-point method is developed to solve (5.1), and its performance and feasibility are analyzed using the dynamic regret and fit metrics.

5.3.1 Algorithm development

Consider now the per-slot problem (5.5), which contains the current objective $f_t(\mathbf{x})$, the current constraint $\mathbf{g}_t(\mathbf{x}) \leq \mathbf{0}$, and a time-invariant constraint set $\mathcal{X}$. With $\boldsymbol{\lambda} \in \mathbb{R}_+^I$ denoting the Lagrange multiplier associated with the time-varying constraint, the online (partial) Lagrangian of (5.5) can be expressed as

$$\mathcal{L}_t(\mathbf{x}, \boldsymbol{\lambda}) := f_t(\mathbf{x}) + \boldsymbol{\lambda}^\top \mathbf{g}_t(\mathbf{x}) \tag{5.7}$$

where $\mathbf{x} \in \mathcal{X}$ remains implicit. For the online Lagrangian (5.7), we introduce a modified online saddle point (MOSP) approach, which takes a modified descent step in the primal domain, and a dual ascent step at each time slot t in a Gauss-Seidel manner. Specifically, given the previous primal iterate $\mathbf{x}_{t-1}$ and the current dual iterate $\boldsymbol{\lambda}_t$ at each slot t, the current decision $\mathbf{x}_t$ is the minimizer of the following optimization problem

$$\min_{\mathbf{x}\in\mathcal{X}} \nabla f_{t-1}(\mathbf{x}_{t-1})^{\top}(\mathbf{x}-\mathbf{x}_{t-1})+\boldsymbol{\lambda}_t^{\top}\mathbf{g}_{t-1}(\mathbf{x})+\frac{\|\mathbf{x}-\mathbf{x}_{t-1}\|^2}{2\alpha} \tag{5.8}$$

where α is a positive stepsize, and $\nabla f_{t-1}(\mathbf{x}_{t-1})$ is the gradient[1] of primal objective $f_{t-1}(\mathbf{x})$ at $\mathbf{x}=\mathbf{x}_{t-1}$. After the current decision $\mathbf{x}_t$ is made, $f_t(\mathbf{x})$ and $\mathbf{g}_t(\mathbf{x})$ are observed, and the dual update takes the form

$$\boldsymbol{\lambda}_{t+1}=\left[\boldsymbol{\lambda}_t+\mu\nabla_{\boldsymbol{\lambda}}\mathcal{L}_t(\mathbf{x}_t,\boldsymbol{\lambda}_t)\right]^{+}=\left[\boldsymbol{\lambda}_t+\mu\mathbf{g}_t(\mathbf{x}_t)\right]^{+} \tag{5.9}$$

where μ is also a positive stepsize, and $\nabla_{\boldsymbol{\lambda}}\mathcal{L}_t(\mathbf{x}_t,\boldsymbol{\lambda}_t)=\mathbf{g}_t(\mathbf{x}_t)$ is the gradient of online Lagrangian (5.7) with respect to (w.r.t.) $\boldsymbol{\lambda}$ at $\boldsymbol{\lambda}=\boldsymbol{\lambda}_t$. Clearly, updating $\boldsymbol{\lambda}_t$ and $\mathbf{x}_t$ at slot t only requires information of the cost and constraint at the previous slot.

Remark 3. The primal gradient step of the classical saddle-point approach in [82, 106, 125] is tantamount to minimizing a first-order approximation of $\mathcal{L}_{t-1}(\mathbf{x},\boldsymbol{\lambda}_t)$ at $\mathbf{x}=\mathbf{x}_{t-1}$ plus a proximal term. We call the recursion (5.8) and (5.9) as a modified online saddle-point approach, since the primal update (5.8) is not an exact gradient step when the constraint $\mathbf{g}_t(\mathbf{x})$ is nonlinear w.r.t. $\mathbf{x}$. Similar to the primal update of OCO with long-term but time-invariant constraints in [174], the minimization in (5.8) penalizes the exact constraint violation $\mathbf{g}_t(\mathbf{x})$ instead of its first-order approximation, which improves control of constraint violations and facilitates performance analysis of MOSP. Nevertheless, when $\mathbf{g}_t(\mathbf{x})$ is linear, (5.8) and (5.9) reduce to the online saddle-point approach using the Gauss-Seidel update, which is different to those with the Jacobi one in [82, 106, 125].

Remark 4. When $\mathbf{g}_t(\mathbf{x})$ is linear or quadratic, the computational complexity of (5.8) is fairly low, and closed-form solutions are available. When $\mathbf{g}_t(\mathbf{x})$ is generally a convex function, penalizing the exact constraint in (5.8) comes with higher computational complexity than the saddle-point method. However, as (5.8) is strongly convex, iterative solvers can find the minimizer at linear convergence rate. Linearization techniques can be also incorporated to facilitate its implementation under fast dynamics, in which case the accuracy depends on the smoothness of $\mathbf{g}_t(\mathbf{x})$, and the variability of $\{\mathbf{x}_t\}$ (that can be e.g., controlled by the choice of stepsize α).

[1]One can replace the gradient by one of the sub-gradients when $f_t(\mathbf{x})$ is non-differentiable. The performance analysis still holds true for this case.

Algorithm 6 Modified online saddle-point (MOSP) method

```
1: Initialize: primal iterate $\mathbf{x}_0$, dual iterate $\boldsymbol{\lambda}_1$, and proper stepsizes $\alpha$ and $\mu$.
2: for $t = 1, 2\dots$ do
3:     Update primal variable $\mathbf{x}_t$ by solving (5.8).
4:     Observe the current cost $f_t(\mathbf{x})$ and constraint $\mathbf{g}_t(\mathbf{x})$.
5:     Update the dual variable $\boldsymbol{\lambda}_{t+1}$ via (5.9).
6: end for
```

5.3.2 Performance analysis

We proceed to show that for MOSP, the dynamic regret in (5.4) and the fit in (5.6) are both sub-linearly increasing, if the accumulated variations of per-slot minimizers and constraints are known to be sub-linearly growing. Before formally stating this result, we assume that the following conditions are satisfied.

Assumption 5. *For every t, the cost function $f_t(\mathbf{x})$ and the time-varying constraint $\mathbf{g}_t(\mathbf{x})$ in* (5.1) *are convex.*

Assumption 6. *For every t, $f_t(\mathbf{x})$ has bounded gradient on $\mathcal{X}$; i.e., $\|\nabla f_t(\mathbf{x})\| \leq G,\ \forall \mathbf{x} \in \mathcal{X}$; and $\mathbf{g}_t(\mathbf{x})$ is bounded on $\mathcal{X}$; i.e., $\|\mathbf{g}_t(\mathbf{x})\| \leq M,\ \forall \mathbf{x} \in \mathcal{X}$.*

Assumption 7. *The radius of the convex feasible set $\mathcal{X}$ is bounded; i.e., $\|\mathbf{x}-\mathbf{y}\| \leq R,\ \forall \mathbf{x}, \mathbf{y} \in \mathcal{X}$.*

Assumption 8. *There exists a constant $\epsilon > 0$, and an interior point $\tilde{\mathbf{x}}_t \in \mathcal{X}$ such that $\mathbf{g}_t(\tilde{\mathbf{x}}_t) \leq -\epsilon\mathbf{1},\ \forall t$.*

Assumption 9. *The slack constant ϵ in Assumption 8 satisfies $\epsilon > \bar{V}(\mathbf{g})$, where the point-wise maximum variation of consecutive constraints is defined as*

$$\bar{V}(\mathbf{g}) := \max_t \max_{\mathbf{x}\in\mathcal{X}} \left\|[\mathbf{g}_{t+1}(\mathbf{x})-\mathbf{g}_t(\mathbf{x})]^+\right\|. \tag{5.10}$$

Assumption 5 is necessary for regret analysis in the OCO setting. Assumption 6 bounds primal and dual gradients per slot, which is also typical in OCO [141, 174, 82, 62]. Assumption 7 restricts the action set to be bounded. Assumption 8 is Slater's condition, which guarantees the existence of a bounded optimal Lagrange multiplier [13]. Assumption 9 indicates that the slack constant ϵ is larger than the maximum variation of constraints. Although not always satisfied, it is a key assumption in our proof of the bounded dual iterate (the scaled fit). Equivalently, it requires $\min_{i,t} \max_{\mathbf{x}\in\mathcal{X}}[-g_t^i(\mathbf{x})]^+ > \max_t \max_{\mathbf{x}\in\mathcal{X}} \left\|[\mathbf{g}_{t+1}(\mathbf{x})-\mathbf{g}_t(\mathbf{x})]^+\right\|$, which is valid when the feasible region defined by $\mathbf{g}_t(\mathbf{x}) \leq \mathbf{0}$ is large enough, or, the trajectory of $\mathbf{g}_t(\mathbf{x})$ is smooth enough across time. Besides, Assumption 9 is analogous to the assumption of bounded multipliers in prior OCO works involving long-term constraints [168, 82]. One simple example for Assumption 9 to hold is that (cf. $I = 1$)

$$g_t(x) := 10x + \cos(\pi t),\ \text{with}\, x \in \mathcal{X} {:=} \{x| - 2 {\leq} x {\leq} 2\} \tag{5.11}$$

where we have $\epsilon = \min_t \max_{x\in\mathcal{X}}[-g_t(x)]^+ = 19$, and $\bar{V}(\mathbf{g}) \leq 2$, so that $\epsilon > \bar{V}(\mathbf{g})$.

Under these assumptions, we are on track to first provide an upper bound for the dynamic fit.

Theorem 9. *Under Assumptions 5-9 and the dual variable initialization $\boldsymbol{\lambda}_1 = \mathbf{0}$, the dual iterate for the MOSP recursion* (5.8)-(5.9) *is bounded by*

$$\|\boldsymbol{\lambda}_t\| \leq \|\bar{\boldsymbol{\lambda}}\| := \mu M + \frac{2GR + R^2/(2\alpha) + (\mu M^2)/2}{\epsilon - \bar{V}(\mathbf{g})},\ \forall t \tag{5.12}$$

and the dynamic fit in (5.6) *is upper-bounded by*

$$\text{Fit}_T^{\text{d}} \leq \frac{\|\boldsymbol{\lambda}_{T+1}\|}{\mu} \leq \frac{\|\bar{\boldsymbol{\lambda}}\|}{\mu} = M + \frac{2GR/\mu{+}R^2/(2\alpha\mu){+}M^2/2}{\epsilon - \bar{V}(\mathbf{g})} \tag{5.13}$$

where G, M, R, and ϵ are as in Assumptions 6-8.

Proof: See Appendix 5.5.1.

Theorem 9 asserts that under the condition on the time-varying constraints, $\|\boldsymbol{\lambda}_t\|$ is uniformly upper-bounded, and more importantly, its scaled version $\|\boldsymbol{\lambda}_{T+1}\|/\mu$ upper bounds the dynamic fit. Observe that with a fixed primal stepsize α, Fit_T^{d} is in the order of $\mathcal{O}(1/\mu)$, thus a larger dual stepsize essentially enables a better satisfaction of long-term constraints. In addition, a smaller $\bar{V}(\mathbf{g})$ leads to a smaller dynamic fit, which also makes sense intuitively.

In the next theorem, we further bound the dynamic regret.

Theorem 10. *Under Assumptions 5-9 and the dual variable initialization $\boldsymbol{\lambda}_1 = \mathbf{0}$, the MOSP recursion* (5.8)-(5.9) *yields a dynamic regret*

$$\begin{aligned}\text{Reg}_T^{\text{d}} {\leq} \frac{RV(\{\mathbf{x}_t^*\}_{t=1}^T)}{\alpha} + \frac{\alpha G^2 T}{2} {+} \frac{\mu M^2(T+1)}{2} {+} \frac{R^2}{2\alpha} \\ + \|\bar{\boldsymbol{\lambda}}\| V(\{\mathbf{g}_t\}_{t=1}^T)\end{aligned} \tag{5.14}$$

where $V(\{\mathbf{x}_t^*\}_{t=1}^T)$ is the accumulated variation of the per-slot minimizers $\mathbf{x}_t^*$ defined as

$$V(\{\mathbf{x}_t^*\}_{t=1}^T) := \sum_{t=1}^T \underbrace{\|\mathbf{x}_t^* - \mathbf{x}_{t-1}^*\|}_{V(\mathbf{x}_t^*)} \tag{5.15}$$

and $V(\{\mathbf{g}_t\}_{t=1}^T)$ is the accumulated variation of constraints

$$V(\{\mathbf{g}_t\}_{t=1}^T) := \sum_{t=1}^T \underbrace{\max_{\mathbf{x}\in\mathcal{X}} \left\|[\mathbf{g}_{t+1}(\mathbf{x}) - \mathbf{g}_t(\mathbf{x})]^+\right\|}_{V(\mathbf{g}_t)}. \tag{5.16}$$

Proof: See Appendix 5.5.2.

Theorem 10 asserts that MOSP's dynamic regret is upper-bounded by a constant depending on the accumulated variations of per-slot minimizers and time-varying constraints as well as the primal and dual stepsizes. While the dynamic regret in the current form (5.14) is hard to grasp, the next corollary shall demonstrate that $\mathrm{Reg}_T^{\mathrm{d}}$ can be very small.

Based on Theorems 9-10, we can readily arrive at the following corollary regarding the optimal stepsizes.

Corollary 1. *Under the same assumptions of Theorems 9-10, if the primal and dual stepsizes are chosen such that*

$$\alpha = \mu = \max\left\{\sqrt{\frac{V(\{\mathbf{x}_t^*\}_{t=1}^T)}{T}}, \sqrt{\frac{V(\{\mathbf{g}_t\}_{t=1}^T)}{T}}\right\} \tag{5.17}$$

then the dynamic regret is upper-bounded by

$$\mathrm{Reg}_T^{\mathrm{d}} = \mathcal{O}\left(\max\left\{\sqrt{V(\{\mathbf{x}_t^*\}_{t=1}^T)T}, \sqrt{V(\{\mathbf{g}_t\}_{t=1}^T)T}\right\}\right) \tag{5.18}$$

and the dynamic fit is upper-bounded by

$$\mathrm{Fit}_T^{\mathrm{d}} = \mathcal{O}\left(\max\left\{\frac{T}{V(\{\mathbf{x}_t^*\}_{t=1}^T)}, \frac{T}{V(\{\mathbf{g}_t\}_{t=1}^T)}\right\}\right). \tag{5.19}$$

Proof: The corollary follows by plugging (5.12) into (5.14), and optimizing (5.13) and (5.14) over the primal-dual stepsizes.

According to Theorems 9-10 and Corollary 1, two sets of stepsizes are discussed next.

S1) Stepsizes without knowledge of variations: If the primal and dual stepsizes are chosen such that $\alpha = \mu = \mathcal{O}(T^{-\frac{1}{3}})$, then the dynamic fit is upper-bounded by

$$\mathrm{Fit}_T^{\mathrm{d}} = \mathcal{O}(T^{\frac{2}{3}}) \tag{5.20a}$$

and the dynamic regret is bounded by

$$\mathrm{Reg}_T^{\mathrm{d}} = \mathcal{O}\Big(\max\Big\{V(\{\mathbf{x}_t^*\}_{t=1}^T)T^{\frac{1}{3}},\ V(\{\mathbf{g}_t\}_{t=1}^T)T^{\frac{1}{3}},\ T^{\frac{2}{3}}\Big\}\Big). \tag{5.20b}$$

S2) Stepsizes with knowledge of variations: Assume that there exists a constant $\beta \in [0, 1)$ such that the temporal variations satisfy $V(\{\mathbf{x}_t^*\}_{t=1}^T) = \mathbf{o}(T^\beta)$ and $V(\{\mathbf{g}_t\}_{t=1}^T) = \mathbf{o}(T^\beta)$. Corollary 1 then implies that choosing the stepsizes as $\alpha = \mu = \mathcal{O}(T^{\frac{\beta-1}{2}})$ leads to the dynamic fit

$$\mathrm{Fit}_T^{\mathrm{d}} = \mathcal{O}(T^{1-\beta}) = \mathbf{o}(T) \tag{5.21a}$$

and the corresponding dynamic regret

$$\mathrm{Reg}_T^{\mathrm{d}} = \mathcal{O}\left(T^{\frac{\beta+1}{2}}\right) = \mathbf{o}(T). \tag{5.21b}$$

In the case (S1), sub-linear regret and fit can be achieved given that $V(\{\mathbf{x}_t^*\}_{t=1}^T) = \mathbf{o}(T^{\frac{2}{3}})$ and $V(\{\mathbf{g}_t\}_{t=1}^T) = \mathbf{o}(T^{\frac{2}{3}})$. In the case (S2), the necessary conditions for the environment can be relaxed to $V(\{\mathbf{x}_t^*\}_{t=1}^T) = \mathbf{o}(T)$ and $V(\{\mathbf{g}_t\}_{t=1}^T) = \mathbf{o}(T)$, provided that a-priori knowledge of the environment is available. For example, when allocating resources to smart grids, the temporal variations of the best dynamic solutions and instantaneous constraints can be estimated using day-ahead forecasting of electricity loads and prices. Corollary 1 provides valuable insights for choosing optimal stepsizes in non-stationary settings. Specifically, adjusting stepsizes to match the variability of the environment is the key to achieving the optimal dynamic regret and fit. Intuitively, when the variation is fast (a larger β), slowly decaying stepsizes (thus larger stepsizes) can better track the potential changes; and vice versa.

It is instructive to give several cases where sub-linear accumulated variations emerge, so that the bounds in (5.21) hold.

C1) Intermittent switches: With $\mathbf{x}_t^* \neq \mathbf{x}_{t+1}^*$ or $\mathbf{g}_t \neq \mathbf{g}_{t+1}$ defining a switch, the number of switches is sub-linear over T; i.e., $\sum_{t=1}^T \mathbb{1}_{\{\mathbf{x}_t^* \neq \mathbf{x}_{t+1}^*\}} = T^\beta$, and $\sum_{t=1}^T \mathbb{1}_{\{\mathbf{g}_t \neq \mathbf{g}_{t+1}\}} = T^\beta$, $\forall \beta \in [0, 1)$. It then follows that $V(\{\mathbf{x}_t^*\}_{t=1}^T) = \mathcal{O}(T^\beta)$, and $V(\{\mathbf{g}_t\}_{t=1}^T) = \mathcal{O}(T^\beta)$, since the one-slot variation of the minimizer and the constraint is bounded; see Assumptions 6-7.

C2) Decreasing variations: When the one-slot variations are decreasing over time such that $V(\mathbf{x}_t^*) = \mathcal{O}(t^{\beta-1})$ and $V(\mathbf{g}_t) = \mathcal{O}(t^{\beta-1})$, $\forall \beta \in [0,1)$, the accumulated variations of the per-slot minimizers and the consecutive constraints become $V(\{\mathbf{x}_t^*\}_{t=1}^T) = \mathcal{O}(T^\beta)$, and $V(\{\mathbf{g}_t\}_{t=1}^T) = \mathcal{O}(T^\beta)$.

Other cases do exist for which the accumulated variation is sub-linear, including the interplay between (C1) and (C2).

Remark 5. Theorems 9-10 and Corollary 1 are in the spirit of the recent works in [185], [15, 112, 62, 71] and [140], where the regret bounds are established with respect to a dynamic benchmark in OCO without long-term time-varying constraints. Specifically, [15, 112] consider dynamic regret bounds for strongly-convex loss functions. For the general convex loss functions considered here, [185] reports the dynamic regret bound in the form of

$$\mathrm{Reg}_T^{\mathrm{d}} = \mathcal{O}\left(\sqrt{V(\{\mathbf{x}_t^*\}_{t=1}^T)T}\right) \tag{5.22}$$

and [15] states the bound in the form of

$$\mathrm{Reg}_T^{\mathrm{d}} = \mathcal{O}\left(V(\{f_t\}_{t=1}^T)^{\frac{1}{3}} T^{\frac{2}{3}}\right) \tag{5.23}$$

where the accumulated variation of loss functions is defined as $V(\{f_t\}_{t=1}^T) := \sum_{t=1}^T \max_{\mathbf{x}\in\mathcal{X}} \|f_{t+1}(\mathbf{x}) - f_t(\mathbf{x})\|$. The dynamic regret bound in [71] considers a hybrid version of (5.22) and (5.23), when the effect of dynamic models is further accounted for in the dynamic regret bounds of [62, 140]. When the functional variation $V(\{f_t\}_{t=1}^T)$ is not directly comparable to the variation of minimizers $V(\{\mathbf{x}_t^*\}_{t=1}^T)$, our regret bound in (5.18) immediately reduces to (5.22) in [185], by setting $\alpha = \sqrt{V(\{\mathbf{x}_t^*\}_{t=1}^T)/T}$. Note that [185, 15, 112, 62, 71, 140] do not account for long-term and time-varying constraints, while the regret analysis is generalized here to the setting with long-term constraints. Interestingly though, in the considered setting, sub-linear dynamic regret and fit can be achieved when the environment consisting of the per-slot minimizer and the time-varying constraint *does not vary on average*, that is, $V(\{\mathbf{x}_t^*\}_{t=1}^T)$ and $V(\{\mathbf{g}_t\}_{t=1}^T)$ are sub-linearly increasing over T. Selecting the optimal stepsizes requires the knowledge of variations, and thus it is also promising to develop a parameter-free MOSP using the doubling trick [71].

5.3.3 Beyond dynamic regret

Although the dynamic benchmark in (5.4) is more competitive than the static one in (5.2), it is worth noting that the sequence of the per-slot minimizer $\mathbf{x}_t^*$ in (5.5) is not the optimal solution to problem (5.1). Consider the offline optimal solutions to (5.1), i.e.,

$$\{\mathbf{x}_t^{\text{off}}\}_{t=1}^T \in \arg \min_{\{\mathbf{x}_t \in \mathcal{X}, \forall t\}} \sum_{t=1}^T f_t(\mathbf{x}_t) \;\text{ s. to } \sum_{t=1}^T \mathbf{g}_t(\mathbf{x}_t) \leq \mathbf{0}. \tag{5.24}$$

Computing the per-slot minimizer $\mathbf{x}_t^*$ in (5.5) only requires one-slot-ahead information (namely, $f_t(\mathbf{x})$ and $g_t(\mathbf{x})$), while computing each $\mathbf{x}_t^{\text{off}}$ within $\{\mathbf{x}_t^{\text{off}}\}_{t=1}^T$ requires information over the entire time horizon (that is, $\{f_t(\mathbf{x})\}_{t=1}^T$ and $\{g_t(\mathbf{x})\}_{t=1}^T$). For this reason, we use the superscript "off" in $\{\mathbf{x}_t^{\text{off}}\}_{t=1}^T$ to emphasize that this solution comes from offline computation with information over T slots. Note that for the cases without long-term constraints [15, 62, 71, 112], the offline solutions $\{\mathbf{x}_t^{\text{off}}\}_{t=1}^T$ coincides with the sequence of per-slot minimizers $\{\mathbf{x}_t^*\}_{t=1}^T$.

Regarding feasibility, $\{\mathbf{x}_t^{\text{off}}\}_{t=1}^T$ exactly satisfies the long-term constraint (5.1b), while the solution of MOSP satisfies (5.1b) on average under mild conditions (cf. Corollary 1). For optimality, the cost of the online decisions $\{\mathbf{x}_t\}_{t=1}^T$ attained by MOSP is further benchmarked by the offline solutions $\{\mathbf{x}_t^{\text{off}}\}_{t=1}^T$. To this end, define MOSP's *optimality gap* as

$$\text{OptGap}_T^{\text{off}} := \sum_{t=1}^T f_t(\mathbf{x}_t) - \sum_{t=1}^T f_t(\mathbf{x}_t^{\text{off}}). \tag{5.25a}$$

Intuitively, if $\{\mathbf{x}_t^{\text{off}}\}_{t=1}^T$ are close to $\{\mathbf{x}_t^*\}_{t=1}^T$, the dynamic regret Reg_T^{d} is able to provide an accurate performance measure in the sense of $\text{OptGap}_T^{\text{off}}$. Specifically, one can decompose the optimality gap as

$$\text{OptGap}_T^{\text{off}} = \underbrace{\sum_{t=1}^T f_t(\mathbf{x}_t) - \sum_{t=1}^T f_t(\mathbf{x}_t^*)}_{\mathcal{U}_1} + \underbrace{\sum_{t=1}^T f_t(\mathbf{x}_t^*) - \sum_{t=1}^T f_t(\mathbf{x}_t^{\text{off}})}_{\mathcal{U}_2} \tag{5.25b}$$

where $\mathcal{U}_1$ corresponds to the dynamic regret Reg_T^{d} in (5.4) capturing the regret relative to the sequence of per-slot minimizers with one-slot-ahead information, and $\mathcal{U}_2$ is the difference between the performance of per-slot minimizers and the offline optimal solutions. Although the second term appears difficult to quantify, we will show next that $\mathcal{U}_2$ is driven by the accumulated variation of the dual functions associated with (5.5).

To this end, consider the dual function of the instantaneous primal problem (5.5), which can be expressed by minimizing the online Lagrangian in (5.7) at time t, namely [13]

$$\mathcal{D}_t(\boldsymbol{\lambda}) := \min_{\mathbf{x} \in \mathcal{X}} \mathcal{L}_t(\mathbf{x}, \boldsymbol{\lambda}) = \min_{\mathbf{x} \in \mathcal{X}} f_t(\mathbf{x}) + \boldsymbol{\lambda}^\top \mathbf{g}_t(\mathbf{x}). \tag{5.26}$$

Likewise, the dual function of (5.1) over the entire horizon is

$$\begin{aligned}\mathcal{D}(\boldsymbol{\lambda}) &:= \min_{\{\mathbf{x}_t\in\mathcal{X},\forall t\}} \sum_{t=1}^{T}\mathcal{L}_t(\mathbf{x}_t,\boldsymbol{\lambda})\\ &\overset{(a)}{=} \sum_{t=1}^{T}\min_{\mathbf{x}_t\in\mathcal{X}}\mathcal{L}_t(\mathbf{x}_t,\boldsymbol{\lambda}) \overset{(b)}{=} \sum_{t=1}^{T}\mathcal{D}_t(\boldsymbol{\lambda})\end{aligned} \tag{5.27}$$

where equality (a) holds since the minimization is separable across the summand at time t, and equality (b) is due to the definition of the per-slot dual function in (5.26). As the primal problems (5.1) and (5.5) are both convex, Slater's condition in Assumption 8 implies that strong duality holds. Accordingly, $\mathcal{U}_2$ in (5.25b) can be written as

$$\sum_{t=1}^{T} f_t(\mathbf{x}_t^*) - \sum_{t=1}^{T} f_t(\mathbf{x}_t^{\text{off}}) = \sum_{t=1}^{T}\max_{\boldsymbol{\lambda}_t\geq\mathbf{0}}\mathcal{D}_t(\boldsymbol{\lambda}_t) - \max_{\boldsymbol{\lambda}\geq\mathbf{0}}\sum_{t=1}^{T}\mathcal{D}_t(\boldsymbol{\lambda}) \tag{5.28}$$

which is the difference between the dual objective of the static best solution, i.e., $\boldsymbol{\lambda}^*\in\arg\max_{\boldsymbol{\lambda}\geq\mathbf{0}}\sum_{t=1}^{T}\mathcal{D}_t(\boldsymbol{\lambda})$, and that of the per-slot best solution for (5.26), i.e., $\boldsymbol{\lambda}_t^*\in\arg\max_{\boldsymbol{\lambda}_t\geq\mathbf{0}}\mathcal{D}_t(\boldsymbol{\lambda}_t)$. Leveraging this special property of the dual problem, we next establish that $\mathcal{U}_2$ can be bounded by the variation of the dual function, thus providing an estimate of the optimality gap (5.25a).

Proposition 6. *Define the variation of the dual function* (5.26) *from time t to $t+1$ as*

$$V(\mathcal{D}_t) := \max_{\boldsymbol{\lambda}\geq\mathbf{0}}\|\mathcal{D}_{t+1}(\boldsymbol{\lambda}) - \mathcal{D}_t(\boldsymbol{\lambda})\| \tag{5.29}$$

and the total variation over the time horizon T as $V(\{\mathcal{D}_t\}_{t=1}^T) := \sum_{t=1}^{T} V(\mathcal{D}_t)$. Then the cost difference between the best offline solution and the best dynamic solution satisfies

$$\sum_{t=1}^{T} f_t(\mathbf{x}_t^*) - \sum_{t=1}^{T} f_t(\mathbf{x}_t^{\text{off}}) \leq 2TV(\{\mathcal{D}_t\}_{t=1}^T) \tag{5.30}$$

where $\mathbf{x}_t^*$ is the minimizer of the instantaneous problem (5.5)*, and $\mathbf{x}_t^{\text{off}}$ solves* (5.1) *with all future information available. Combined with* (5.25b)*, it readily follows that*

$$\text{OptGap}_T^{\text{off}} \leq \text{Reg}_T^{\text{d}} + 2TV(\{\mathcal{D}_t\}_{t=1}^T) \tag{5.31}$$

where Reg_T^{d} is defined in (5.4)*, and $\text{OptGap}_T^{\text{off}}$ in* (5.25)*.*

Proof: Instead of going to the primal domain, we upper bound $\mathcal{U}_2$ via the dual representation in

(5.28). Letting $\tilde{t}$ denote any slot in $\mathcal{T} := \{1, \ldots, T\}$, we have

$$\begin{aligned}
&\sum_{t\in\mathcal{T}} \max_{\boldsymbol{\lambda}\geq\mathbf{0}} \mathcal{D}_t(\boldsymbol{\lambda}) - \max_{\boldsymbol{\lambda}\geq\mathbf{0}} \sum_{t\in\mathcal{T}} \mathcal{D}_t(\boldsymbol{\lambda}) \\
&\leq \sum_{t\in\mathcal{T}} \big(\mathcal{D}_t(\boldsymbol{\lambda}_t^*) - \mathcal{D}_t(\boldsymbol{\lambda}_{\tilde{t}}^*)\big) \leq T \max_{t\in\mathcal{T}} \big\{\mathcal{D}_t(\boldsymbol{\lambda}_t^*) - \mathcal{D}_t(\boldsymbol{\lambda}_{\tilde{t}}^*)\big\}\,.
\end{aligned} \tag{5.32}$$

The first inequality comes from the definition $\boldsymbol{\lambda}_t^* \in \arg\max_{\boldsymbol{\lambda}\geq\mathbf{0}} \mathcal{D}_t(\boldsymbol{\lambda})$. Note that if $\max_{t\in\mathcal{T}}\{\mathcal{D}_t(\boldsymbol{\lambda}_t^*) - \mathcal{D}_t(\boldsymbol{\lambda}_{\tilde{t}}^*)\} \leq 2V(\{\mathcal{D}_t\}_{t=1}^T)$, the proposition readily follows from (5.32). We will prove this inequality by contradiction. Assume there exists a slot $t_0 \in \mathcal{T}$ such that $\mathcal{D}_{t_0}(\boldsymbol{\lambda}_{t_0}^*) - \mathcal{D}_{t_0}(\boldsymbol{\lambda}_{\tilde{t}}^*) > 2V(\{\mathcal{D}_t\}_{t=1}^T)$, which implies that

$$\begin{aligned}
\mathcal{D}_{\tilde{t}}(\boldsymbol{\lambda}_{\tilde{t}}^*) &\overset{(a)}{\leq} \mathcal{D}_{t_0}(\boldsymbol{\lambda}_{\tilde{t}}^*) + V(\{\mathcal{D}_t\}_{t=1}^T) \overset{(b)}{<} \mathcal{D}_{t_0}(\boldsymbol{\lambda}_{t_0}^*) - V(\{\mathcal{D}_t\}_{t=1}^T) \\
&\overset{(c)}{\leq} \mathcal{D}_{\tilde{t}}(\boldsymbol{\lambda}_{t_0}^*),\ \forall\, \tilde{t} \in \mathcal{T}
\end{aligned} \tag{5.33}$$

where inequalities (a) and (c) come from the fact that $V(\{\mathcal{D}_t\}_{t=1}^T)$ is the accumulated variation over T slots, and hence $\max_{t_1,t_2\in\mathcal{T}} \|\mathcal{D}_{t_1}(\boldsymbol{\lambda}) - \mathcal{D}_{t_2}(\boldsymbol{\lambda})\| \leq V(\{\mathcal{D}_t\}_{t=1}^T)$, while (b) is due to the hypothesis above. Note that $\mathcal{D}_{\tilde{t}}(\boldsymbol{\lambda}_{\tilde{t}}^*) < \mathcal{D}_{\tilde{t}}(\boldsymbol{\lambda}_{t_0}^*)$ in (5.33) contradicts the fact that $\boldsymbol{\lambda}_{\tilde{t}}^*$ is the maximizer of $\mathcal{D}_{\tilde{t}}(\boldsymbol{\lambda})$. Therefore, we have $\mathcal{D}_t(\boldsymbol{\lambda}_{\tilde{t}}^*) - \mathcal{D}_t(\boldsymbol{\lambda}_t^*) \leq 2V(\{\mathcal{D}_t\}_{t=1}^T)$, which completes the proof.

The following remark provides an approach to improving the bound in Proposition 1.

Remark 6. Although the optimality gap in (5.31) appears to be at least linear w.r.t. T, one can use the "restarting" trick for dual variables, similar to that for primal variables in the unconstrained case; see e.g., [15]. Specifically, if the total variation $V(\{\mathcal{D}_t\}_{t=1}^T)$ is known a-priori, one can divide the entire time horizon $\mathcal{T} := \{1, \ldots, T\}$ into $\lceil T/\Delta_T \rceil$ sub-horizons (each with $\Delta_T = \mathbf{o}\big(T/V(\{\mathcal{D}_t\}_{t=1}^T)\big)$ slots), and restart the dual iterate $\boldsymbol{\lambda}$ at the beginning of each sub-horizon. By assuming that $V(\{\mathcal{D}_t\}_{t=1}^T)$ is sub-linear w.r.t. T, one can guarantee that $\Delta_T \geq 1$ always exists. In this case, the optimality gap in (5.31) can be improved by

$$\mathrm{OptGap}_T^{\mathrm{off}} \leq \lceil T/\Delta_T \rceil \mathrm{Reg}_{\Delta_T}^{\mathrm{d}} + 2\Delta_T V(\{\mathcal{D}_t\}_{t=1}^T) \tag{5.34a}$$

and the dynamic fit is the summation over each sub-horizon

$$\mathrm{Fit}_T^{\mathrm{d}} \leq \lceil T/\Delta_T \rceil \mathrm{Fit}_{\Delta_T}^{\mathrm{d}}. \tag{5.34b}$$

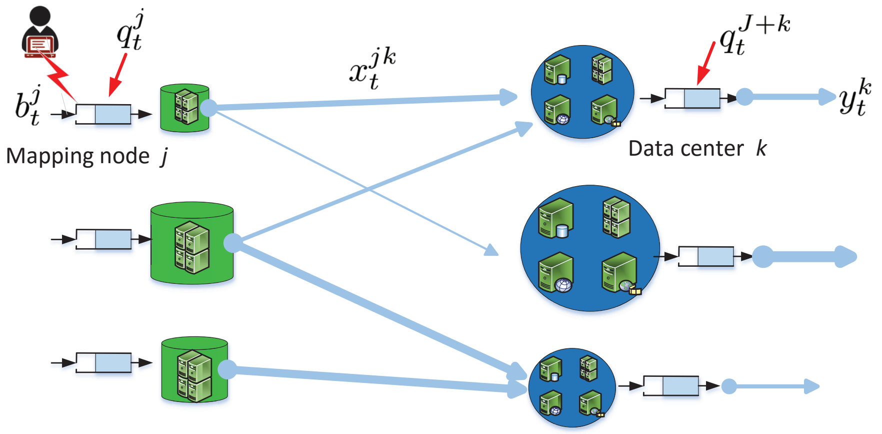

Figure 5.1: A diagram of online network resource allocation. Per time t, mapping node j has an exogenous workload b_t^j plus that stored in the queue q_t^j, and schedules workload x_t^{jk} to data center k. Data center k serves an amount of workload y_t^k out of the assigned $\sum_{j=1}^J x_t^{jk}$ as well as that stored in its queue q_t^{J+k}. The thickness of each edge is proportional to its capacity.

To this end, if the regularity conditions of the environment in (5.21) are satisfied, one can properly set the primal-dual stepsizes to guarantee the sub-linear regret and fit on each sub-horizon. Correspondingly, the optimality gap and the dynamic fit in (5.34) are also both sub-linearly growing with time. Interested readers are referred to [15] for details of this restarting trick, which are omitted here due to space limitation.

5.4 Application to network resource allocation

In this section, we solve the network resource allocation problem within the OCO framework, and present numerical experiments to demonstrate the merits of our MOSP solver.

5.4.1 Online network resource allocation

Consider the resource allocation problem over a cloud network [25], which is represented by a directed graph $\mathcal{G} = (\mathcal{I}, \mathcal{E})$ with node set $\mathcal{I}$ and edge set $\mathcal{E}$, where $|\mathcal{I}| = I$ and $|\mathcal{E}| = E$. Nodes considered here include mapping nodes collected in the set $\mathcal{J} = \{1, \dots, J\}$, and data centers collected in the set $\mathcal{K} = \{1, \dots, K\}$; i.e., we have $\mathcal{I} = \mathcal{J} \bigcup \mathcal{K}$.

Per time t, each mapping node j receives an exogenous data request b_t^j, and forwards the amount x_t^{jk} to each data center k in accordance with bandwidth availability. Each data center k schedules workload y_t^k according to its resource availability. Regarding y_t^k as the weight of a virtual outgoing edge $(k, *)$ from data center k, edge set $\mathcal{E} := \{(j,k), \forall j \in \mathcal{J}, k \in \mathcal{K}\} \bigcup \{(k,*), \forall k \in$

$\mathcal{K}\}$ contains all the links connecting mapping nodes with data centers, and all the "virtual" edges coming out of the data centers. The $I \times E$ node-incidence matrix is formed with the (i,e)-th entry

$$\mathbf{A}_{(i,e)} = \begin{cases} 1, & \text{if link } e \text{ enters node } i \\ -1, & \text{if link } e \text{ leaves node } i \\ 0, & \text{else.} \end{cases} \tag{5.35}$$

For compactness, collect the data workloads across edges $e = (i,j) \in \mathcal{E}$ in a resource allocation vector $\mathbf{x}_t := [x_t^{11}, \ldots, x_t^{JK}, y_t^1, \ldots, y_t^K]^\top \in \mathbb{R}_+^E$, and the exogenous load arrival rates of all nodes in a vector $\mathbf{b}_t := [b_t^1, \ldots, b_t^J, 0 \ldots, 0]^\top \in \mathbb{R}_+^I$. Then, the aggregate (endogenous plus exogenous) workloads of all nodes are given by $\mathbf{A}\mathbf{x}_t + \mathbf{b}_t$. When the i-th entry of $\mathbf{A}\mathbf{x}_t + \mathbf{b}_t$ is positive, there is service residual at node i; otherwise, node i over-serves the current workload arrival. Assume that each data center and mapping node has a local data queue to buffer unserved workloads [116]. With $\mathbf{q}_t := [q_t^1, \ldots, q_t^{J+K}]^\top$ collecting the queue lengths at each mapping node and data center, the queue update is $\mathbf{q}_{t+1} = [\mathbf{q}_t + \mathbf{A}\mathbf{x}_t + \mathbf{b}_t]^+$, where $[\cdot]^+$ ensures that the queue length is always non-negative. The bandwidth limit of link (j,k) is $\bar{x}^{jk}$, and the resource capability of data center k is $\bar{y}^k$, which can be compactly expressed by $\mathbf{x} \in \mathcal{X}$ with $\mathcal{X} := \{\mathbf{0} \leq \mathbf{x} \leq \bar{\mathbf{x}}\}$ and $\bar{\mathbf{x}} := [\bar{x}^{11}, \ldots, \bar{x}^{JK}, \bar{y}^1, \ldots, \bar{y}^K]^\top$. The overall system diagram is depicted in Fig. 5.1.

For each data center, the power cost $f_t^k(y_t^k) := f^k(y_t^k; \theta_t^k)$ depends on a time-varying parameter θ_t^k, which captures the energy price and the renewable generation at data center k during slot t. The bandwidth cost $f_t^{jk}(x_t^{jk}) := f^{jk}(x_t^{jk}; \theta_t^{jk})$ characterizes the transmission delay and is parameterized by a time-varying scalar θ_t^{jk}. Scalars θ_t^k and θ_t^{jk} can be readily extended to vector forms. To keep the exposition simple, we use scalars to represent time-varying factors at nodes and edges.

Per slot t, the instantaneous cost $f_t(\mathbf{x}_t)$ aggregates the costs of power consumed at all data centers plus the bandwidth costs at all links, namely

$$f_t(\mathbf{x}_t) := \sum_{k \in \mathcal{K}} \underbrace{f_t^k(y_t^k)}_{\text{power cost}} + \sum_{j \in \mathcal{J}} \sum_{k \in \mathcal{K}} \underbrace{f_t^{jk}(x_t^{jk})}_{\text{bandwidth cost}} \tag{5.36}$$

where the objective can be also written as $f_t(\mathbf{x}_t) := f(\mathbf{x}_t; \boldsymbol{\theta}_t)$ with $\boldsymbol{\theta}_t := [\theta_t^1, \ldots, \theta_t^K, \theta_t^{11}, \ldots, \theta_t^{JK}]^\top$ concatenating all time-varying parameters. Aiming to minimize the accumulated cost while serving all workloads, the optimal workload routing and allocation strategy in this cloud network is the solution of the following optimization problem

$$\min_{\{\mathbf{x}_t \in \mathcal{X}, \forall t\}} \sum_{t=1}^{T} f_t(\mathbf{x}_t) \quad \text{s. to} \;\; \mathbf{q}_{t+1} = [\mathbf{q}_t + \mathbf{A}\mathbf{x}_t + \mathbf{b}_t]^+ \,, \forall t$$
$$\mathbf{q}_1 \geq \mathbf{0}, \;\mathbf{q}_{T+1} = \mathbf{0} \tag{5.37}$$

where $\mathbf{q}_1$ is the given initial queue length, and $\mathbf{q}_{T+1} = \mathbf{0}$ guarantees that all workloads arrived have been served at the end of the scheduling horizon. Note that (5.37) is time-coupled, and generally challenging to solve without information of future workload arrivals and time-varying cost functions. Therefore, we reformulate (5.37) to fit our OCO formulation (5.1) by relaxing the queue recursion in (5.37), namely

$$\mathbf{q}_{T+1} \geq \mathbf{q}_T + \mathbf{A}\mathbf{x}_T + \mathbf{b}_T \geq \mathbf{q}_1 + \sum_{t=1}^{T} (\mathbf{A}\mathbf{x}_t + \mathbf{b}_t) \tag{5.38}$$

which readily leads to $\sum_{t=1}^{T} (\mathbf{A}\mathbf{x}_t + \mathbf{b}_t) \leq \mathbf{q}_{T+1} - \mathbf{q}_1 \leq \mathbf{0}$, since $\mathbf{q}_1 \geq \mathbf{0}$ and $\mathbf{q}_{T+1} = \mathbf{0}$. Therefore, instead of solving (5.37), we aim to tackle a relaxed problem, given by

$$\min_{\{\mathbf{x}_t \in \mathcal{X}, \forall t\}} \sum_{t=1}^{T} f_t(\mathbf{x}_t) \quad \text{s. to} \;\; \sum_{t=1}^{T} (\mathbf{A}\mathbf{x}_t + \mathbf{b}_t) \leq \mathbf{0} \tag{5.39}$$

where the workload flow conservation constraint $\mathbf{A}\mathbf{x}_t + \mathbf{b}_t \leq \mathbf{0}$ must be satisfied in the long term rather than slot-by-slot. Clearly, (5.39) is in the form of (5.1). Therefore, the MOSP algorithm of Section 5.3 can be leveraged to solve (5.39) in an *online* fashion, with provable performance and feasibility guarantees. Specifically, with $\mathbf{g}_t(\mathbf{x}_t) = \mathbf{A}\mathbf{x}_t + \mathbf{b}_t$, the primal update (5.8) boils down to a simple gradient update $\mathbf{x}_t = \mathcal{P}_{\mathcal{X}} \left(\mathbf{x}_{t-1} - \alpha \nabla f_{t-1}(\mathbf{x}_{t-1}) - \alpha \mathbf{A}^\top \boldsymbol{\lambda}_t\right)$, where $\mathcal{P}_{\mathcal{X}}(\cdot)$ defines projection onto the convex set $\mathcal{X}$. The dual update (5.9) is $\boldsymbol{\lambda}_{t+1} = \left[\boldsymbol{\lambda}_t + \mu(\mathbf{A}\mathbf{x}_t + \mathbf{b}_t)\right]^+$, which can be nicely regarded as a scaled version of the queue dynamics in (5.37), with $\mathbf{q}_t = \boldsymbol{\lambda}_t/\mu$.

In addition to simple closed-form updates, MOSP can also afford a fully decentralized implementation by exploiting the problem structure of network resource allocation, where each mapping node or data center decides the amounts on all its *outgoing links*, and only exchanges information with its *one-hop neighbors*. Per time slot t, the primal update at mapping node j includes variables on all its outgoing links, given by

$$x_t^{jk} = \left[x_{t-1}^{jk} - \alpha \nabla f_{t-1}^{jk}(x_{t-1}^{jk}) - \alpha \left(\lambda_t^k - \lambda_t^j\right)\right]_0^{\bar{x}^{jk}}, \; \forall k \in \mathcal{K} \tag{5.40a}$$

Algorithm 7 Distributed MOSP for network resource allocation

```
1: Initialize: primal iterate $\mathbf{x}_0$, dual iterate $\boldsymbol{\lambda}_1$, and proper stepsizes $\alpha$ and $\mu$.
2: for $t = 1, 2 \ldots$ do
3:     Each mapping node $j$ performs (5.40a) and each data center $k$ runs (5.40c).
4:     Mapping nodes and data centers observe local costs and workload arrivals.
5:     Each mapping node $j$ performs (5.40b) and each data center $k$ performs (5.40d).
6:     Mapping nodes (data centers) send multipliers to all neighboring data centers (nodes).
7: end for
```

and the dual update reduces to

$$\lambda_{t+1}^{j} = \left[\lambda_t^j + \mu\left(b_t^j - \sum_{k\in\mathcal{K}} x_t^{jk}\right)\right]^{+}. \tag{5.40b}$$

Likewise, for data center k, the primal update becomes

$$y_t^k = \left[y_{t-1}^k - \alpha\Big(\nabla f_{t-1}^k(y_{t-1}^k) - \lambda_t^k\Big)\right]_0^{\bar{y}^k} \tag{5.40c}$$

where $[\,\cdot\,]_0^{\bar{y}^k} := \min\{\bar{y}^k, \max\{\cdot\,, 0\}\}$, and the dual recursion is

$$\lambda_{t+1}^{k} = \left[\lambda_t^k + \mu\left(\sum_{j\in\mathcal{J}} x_t^{jk} - y_t^k\right)\right]^{+}. \tag{5.40d}$$

Distributed MOSP for online network resource allocation is summarized in Algorithm 7.

5.4.2 Revisiting stochastic dual (sub)gradient

The dynamic network resource allocation problem in Section 5.4.1 has so far been studied in the stochastic setting [28, 25]. Classical approaches include Lyapunov optimization [162, 116] and the stochastic dual (sub)gradient method [108], both of which rely on stochastic approximation (SA) [118]. In the context of stochastic optimization, the time-varying vectors $\{\boldsymbol{\xi}_t\}$ with $\boldsymbol{\xi}_t := [\boldsymbol{\theta}_t^\top, \mathbf{b}_t^\top]^\top$ appearing in the cost and constraint are assumed to be independent realizations of a random variable Ξ.[2] In an SA-based stochastic optimization algorithm, per time t, a policy first observes a realization $\boldsymbol{\xi}_t$ of the random variable Ξ, and then (stochastically) selects an action $\mathbf{x}_t \in \mathcal{X}$. However, in contrast to minimizing the *observed cost* in the OCO setting, the goal of the stochastic

[2]Extension is also available when $\{\boldsymbol{\xi}_t\}$ constitute a sample path from an ergodic stochastic process $\{\Xi_t\}$, which converges to a stationary distribution; see e.g., [49, 130].

resource allocation is usually to minimize the limiting average of the *expected cost* subject to the so-termed stability constraint, namely

$$\min_{\{\mathbf{x}_t \in \mathcal{X}, \mathbf{q}_t, \forall t\}} \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[f_t(\mathbf{x}_t)] \tag{5.41a}$$

$$\text{s. to } \mathbf{q}_{t+1} = [\mathbf{q}_t + \mathbf{A}\mathbf{x}_t + \mathbf{b}_t]^+, \ \forall t \tag{5.41b}$$

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\left[\mathbf{q}_t\right] \leq \mathbf{0} \tag{5.41c}$$

where he expectation in (5.41a) is taken over Ξ and the randomness of $\mathbf{x}_t$ and $\mathbf{q}_t$ induced by all possible sample paths $\{\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_t\}$ via (5.41b); and the stability constraint (5.41c) implies a finite bound on the accumulated constraint violation. In contrast to the observed costs in (5.37), each decision $\mathbf{x}_t$ is evaluated by all possible realizations in Ξ here. However, as $\mathbf{q}_t$ in (5.41b) couples the optimization variables over an infinite time horizon, (5.41) is intractable in general.

Prior works [116, 53, 108, 25] have demonstrated that (5.41) can be tackled via a tractable stationary relaxation, given by

$$\min_{\{\mathbf{x}_t \in \mathcal{X}, \forall t\}} \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[f_t(\mathbf{x}_t)] \tag{5.42a}$$

$$\text{s. to } \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\left[\mathbf{A}\mathbf{x}_t + \mathbf{b}_t\right] \leq \mathbf{0} \tag{5.42b}$$

where the time-coupling constraints (5.41b) and (5.41c) are relaxed to the limiting average constraint (5.42b). Such a relaxation can be verified similar to the queue relaxation in (5.38); see also [116]. Note that (5.42) is still challenging since it involves expectations in both costs and constraints, and the distribution of Ξ is usually unknown. Even if the joint probability distribution function were available, finding the expectations would not scale with the dimensionality of Ξ. A common remedy is to use the stochastic dual gradient (SDG) iteration (a.k.a. Lyapunov optimization) [162, 116, 25]. Specifically, with $\boldsymbol{\lambda} \in \mathbb{R}_+^I$ denoting the multipliers associated with the expectation constraint (5.42b), the SDG method first observes one realization $\boldsymbol{\xi}_t$ at each slot t, and then performs the dual update as

$$\boldsymbol{\lambda}_{t+1} = \left[\boldsymbol{\lambda}_t + \mu(\mathbf{A}\mathbf{x}_t + \mathbf{b}_t)\right]^+, \ \forall t \tag{5.43}$$

where $\boldsymbol{\lambda}_t$ is the dual iterate at time t, $\mathbf{A}\mathbf{x}_t + \mathbf{b}_t$ is the stochastic dual gradient, and μ is a positive (and typically constant) stepsize. The actual allocation or the primal variable $\mathbf{x}_t$ appearing in (5.43) needs be found by solving the following sub-problems, one per slot t

$$\mathbf{x}_t \in \arg\min_{\mathbf{x}\in\mathcal{X}} f_t(\mathbf{x}) + \boldsymbol{\lambda}_t^\top(\mathbf{A}\mathbf{x}+\mathbf{b}_t). \tag{5.44}$$

For the considered network resource allocation problem, SDG in (5.43)-(5.44) entails a well-known cost-delay tradeoff [116]. Specifically, with f^* denoting the optimal objective (5.42), SDG can achieve an $\mathcal{O}(\mu)$-optimal solution such that $\lim_{T\to\infty}(1/T)\sum_{t=1}^{T}\mathbb{E}\left[f_t\left(\mathbf{x}_t\right)\right]\leq f^*+\mathcal{O}(\mu)$, and guarantee queue lengths[3] satisfying $\lim_{T\to\infty}(1/T)\sum_{t=1}^{T}\mathbb{E}\left[\|\mathbf{q}_t\|\right]=\mathcal{O}(1/\mu)$. Therefore, reducing the optimality gap $\mathcal{O}(\mu)$ will essentially increase the average network delay $\mathcal{O}(1/\mu)$.

Remark 7. The optimality of SDG is established relative to the offline optimal solution of (5.42), which can be thought as the time-average *optimality gap* in (5.25a) under the OCO setting. Interestingly though, the optimality gap under the stochastic setting is equivalent to the (expected) dynamic regret (5.4), since their (expected) difference $V(\{\mathbb{E}[\mathcal{D}_t]\}_{t=1}^T)$ in (5.31) reduces to zero. To see this, note that $\mathbb{E}[f_t(\mathbf{x})]$ and $\mathbb{E}[\mathbf{A}\mathbf{x}+\mathbf{b}_t]$ are time-invariant, hence the dual problem of each per-slot subproblem in (5.42) is time-invariant. This reduction means that the SDG solver of the dynamic problem in (5.41) leverages its inherent stationarity (through the stationary dual problem), in contrast to the non-stationary nature of the OCO framework.

Remark 8. Below we highlight several differences of the novel MOSP in Algorithm 7 with the SDG recursion in (5.43)-(5.44) for the dynamic network resource allocation task.

(D1) From an operational perspective, SDG observes the current state $\boldsymbol{\xi}_t$ first, and then performs the resource allocation decision $\mathbf{x}_t$ accordingly. Therefore, at the beginning of slot t, SDG needs to precisely know the non-causal information $\boldsymbol{\xi}_t$. Inheriting the merits of OCO, on the other hand, MOSP operates in a fully *predictive* mode, which decides $\mathbf{x}_t$ without knowing the cost $f_t(\mathbf{x})$ and the constraint $\mathbf{g}_t(\mathbf{x})$ (or $\boldsymbol{\xi}_t$) at time t. This feature of MOSP is of major practical importance when costs and availability of resources are not available at the point of making decisions; e.g., online demand response in smart grids [78, 104] and resource allocation in wireless networking [160].

(D2) From a computational point of view, MOSP reduces to a simple saddle-point recursion with primal (projected) gradient descent and dual gradient ascent for the network resource allocation problem, both of which incur affordable complexity. However, the primal update of SDG in (5.44) generally requires solving a convex program per time slot t, which leads to much higher

[3]According to Little's law [95], the time-average delay is proportional to the time-average queue length given the arrival rate.

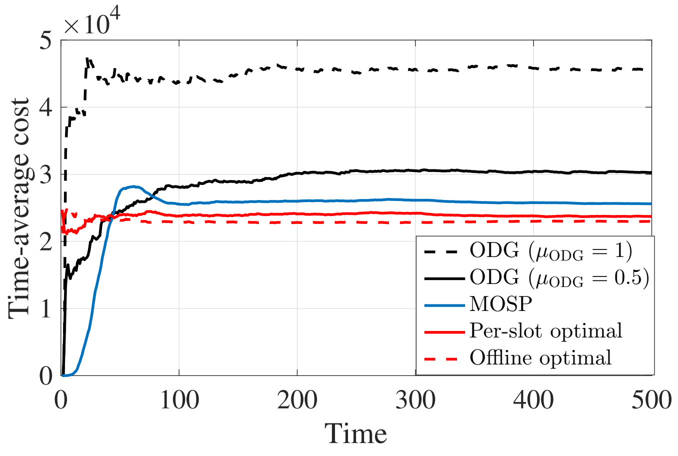

Figure 5.2: Time-average cost for Case 1.

computational complexity in general.

(D3) With regards to the theoretical claims, the time-varying vector $\boldsymbol{\xi}_t$ in SDG typically requires a rather restrictive probabilistic assumption, to establish SDG optimality in either the ensemble average [116] or the limiting ergodic average sense [130]. In contrast, leveraging the OCO framework, MOSP admits finite-sample performance analysis with non-stochastic observed costs and constraints, which can even be adversarial.

5.4.3 Numerical experiments

In this section, we provide numerical tests to demonstrate the merits of the proposed MOSP algorithm in the application of dynamic network resource allocation. Consider the geographical workload routing and allocation task in (5.39) with $J = 10$ mapping nodes and $K = 10$ data centers. The instantaneous network cost in (5.36) is

$$f_t(\mathbf{x}_t) := \sum_{k\in\mathcal{K}} p_t^k (y_t^k)^2 + \sum_{j\in\mathcal{J}} \sum_{k\in\mathcal{K}} c^{jk} (x_t^{jk})^2 \tag{5.45}$$

where p_t^k is the energy price at data center k at time t, and c^{jk} is the per-unit bandwidth cost for transmitting from mapping node j to data center k. With the bandwidth limit $\bar{x}^{jk}$ uniformly randomly generated within $[10, 100]$, we set the bandwidth cost of each link (j, k) as $c^{jk} = 40/\bar{x}^{jk}, \forall j, k$. The resource capacities $\{\bar{y}^k, \forall k\}$ at all data centers are uniformly randomly generated from $[100, 200]$. We consider the following two cases for the time-varying parameters $\{p_t^k, \forall t, k\}$ and $\{b_t^j, \forall t, j\}$:

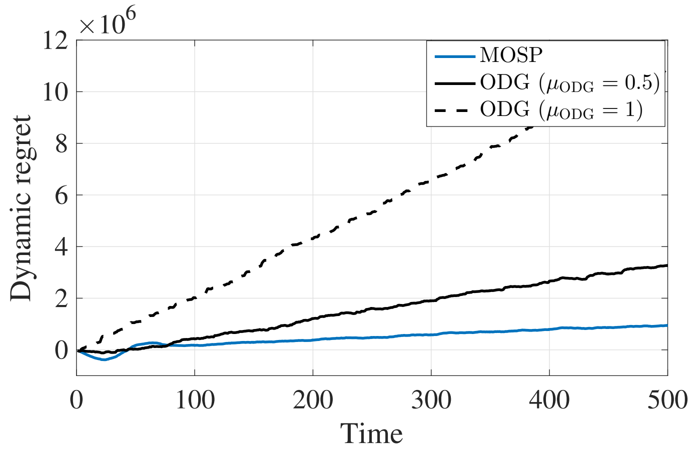

Figure 5.3: Dynamic regret for Case 1.

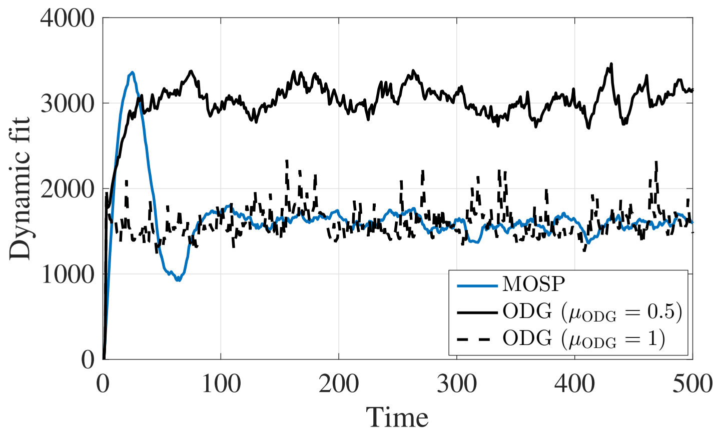

Figure 5.4: Dynamic fit for Case 1.

Case 1) Parameters $\{p_t^k, \forall t, k\}$ and $\{b_t^j, \forall t, j\}$ are independently drawn from time-invariant distributions. Specifically, p_t^k is uniformly distributed over $[1, 3]$, and the delay-tolerant workload b_t^j arrives at each mapping node j according to a uniform distribution over $[50, 150]$.

Case 2) Parameters $\{p_t^k, \forall t, k\}$ and $\{b_t^j, \forall t, j\}$ are generated according to non-stationary stochastic processes. Specifically, $p_t^k = \sin(\pi t/12) + n_t^k$ with i.i.d. noise n_t^k uniformly distributed over $[1, 3]$, while $b_t^j = 50\sin(\pi t/12) + v_t^j$ with i.i.d. noise v_t^j uniformly distributed over $[99, 101]$. One can verify that Assumption 9 is satisfied in this case, as the constraints vary slowly. Intuitively, it means that the network capacity margin is large relative to the temporal variation of arrival rates here.

Finally, with time horizon $T = 500$, the stepsize in (5.40a) and (5.40c) is set to $\alpha = 0.05/T^{1/3}$, and for (5.40b) and (5.40d) to $\mu = 50/T^{1/3}$. MOSP is benchmarked by three strategies: SDG in

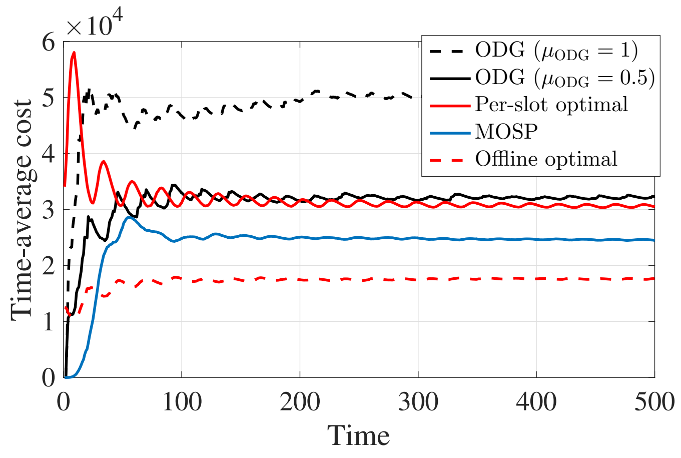

Figure 5.5: Time-average cost for Case 2.

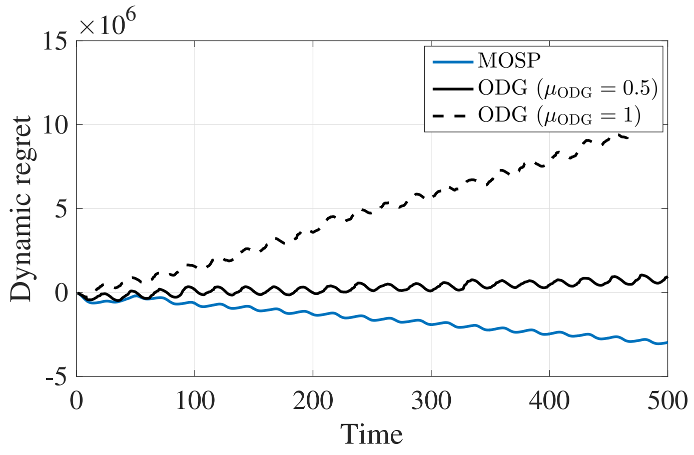

Figure 5.6: Dynamic regret for Case 2.

Section 5.4.2, the sequence of per-slot best minimizers in (5.5), and the offline optimal solution that solves (5.1) at once with all future costs and constraints available. Note that at the beginning of each slot t, the exact prices $\{p_t^k, \forall k\}$ and demands $\{b_t^j, \forall j\}$ for the coming slot are generally not available in practice [1, 78, 160, 68]. Since the original SDG updates (5.43) and (5.44) require non-causal knowledge of $\{p_t^k, \forall k\}$ and $\{b_t^j, \forall j\}$ to decide $\mathbf{x}_t$, we modify them for fairness in this online setting by using the prices and demands at slot $t-1$ to obtain $\mathbf{x}_t$, which we term online dual gradient (ODG). As shown next, different constant stepsizes for ODG's dual update in (5.43) lead to quite different performance and feasibility behaviors; i.e., a larger stepsize results in higher regret but smaller fit, and vice versa. For this reason, ODG is studied under two different stepsizes: $\mu_{\text{ODG}} = 0.5$ balancing the regret and fit of ODG, and $\mu_{\text{ODG}} = 1$ allowing ODG to have similar fit with MOSP.

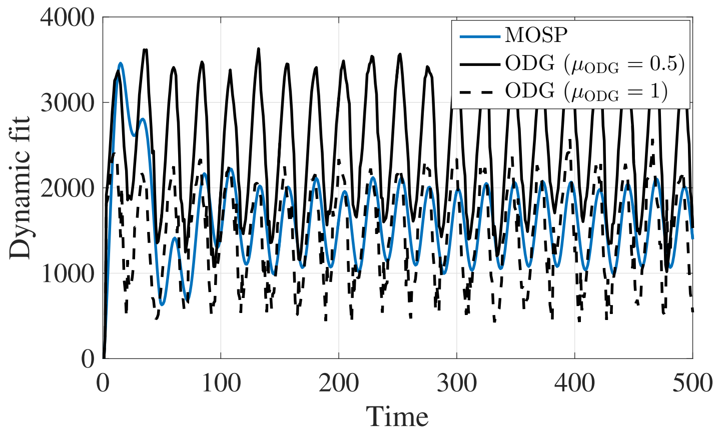

Figure 5.7: Dynamic fit for Case 2.

Figs. 5.2-5.4 show the test results for Case 1 under i.i.d. costs and constraints. Clearly, MOSP in Fig. 5.2 converges to a smaller time-average cost than ODG with the two stepsizes. The time-average cost of MOSP is slightly higher than the per-slot optimal solution, as well as the offline optimal solution with all information of the costs and constraints available over horizon T. Fig. 5.3 confirms the conclusion made from Fig. 5.2, where the dynamic regret (cf. (5.4)) of MOSP grows much slower than that of ODG. Regarding the dynamic fit (cf. (5.6)), Fig. 5.4 demonstrates that ODG with $\mu_{\text{ODG}} = 1$ has a smaller fit than that of $\mu_{\text{ODG}} = 0.5$, and similar to the dynamic fit of MOSP. According to the well-known trade-off between cost (optimality) and delay (constraint violations) in [116], increasing μ_{ODG} will improve the dynamic fit of ODG but degrade its dynamic regret. Therefore, MOSP is favorable in Case 1 since it has much smaller regret when its dynamic fit is similar to that of ODG with $\mu_{\text{ODG}} = 1$. It is worth mentioning that theoretically speaking, the dynamic regret of MOSP may not be sub-linear in this i.i.d. case, since the accumulated cost and constraint variation is not necessarily small enough (cf. Theorem 10). However, MOSP is robust in this aspect at least for the numerical tests we carried.

Simulation tests using non-stationary costs and constraints are shown in Figs. 5.5-5.7. Different from Case 1, the time-average cost of MOSP is not only smaller than ODG, but also smaller than the per-slot optimum obtained via (5.3); see Fig. 5.5. A similar conclusion can be also drawn through the growths of dynamic regret in Fig. 5.6. From a high level, this is because the difference between the cost of the per-slot minimizers and that of the offline solutions is no longer small in the non-stationary case. Regarding Fig. 5.7, both ODG and MOSP have finite dynamic fits in the sense that the accumulated constraint violations do not increase with time. The dynamic fit of MOSP is much smaller than that of ODG with $\mu_{\text{ODG}} = 0.5$, and comparable to that of ODG with $\mu_{\text{ODG}} = 1$. Therefore, in this non-stationary case, MOSP also markedly outperforms ODG in both regret and fit.

5.5 Proofs of lemmas and theorems

Before proving Theorems 9 and 10, we first bound the variation of the dual variable for the MOSP recursion (5.8)-(5.9). With the dual drift defined as $\Delta(\boldsymbol{\lambda}_t) := \left(\|\boldsymbol{\lambda}_{t+1}\|^2 - \|\boldsymbol{\lambda}_t\|^2\right)/2$, we have the following lemma.

Lemma 19. *Per slot t, the dual drift of the MOSP recursion* (5.8)-(5.9) *is upper-bounded as*

$$\Delta(\boldsymbol{\lambda}_t) \leq \mu \boldsymbol{\lambda}_t^\top \mathbf{g}_t(\mathbf{x}_t) + \frac{\mu^2}{2}\|\mathbf{g}_t(\mathbf{x}_t)\|^2. \tag{5.46}$$

Proof: Squaring the dual variable update (5.9), we have

$$\begin{aligned}\|\boldsymbol{\lambda}_{t+1}\|^2 = \left\|\left[\boldsymbol{\lambda}_t + \mu \mathbf{g}_t(\mathbf{x}_t)\right]^+\right\|^2 &\leq \|\boldsymbol{\lambda}_t + \mu \mathbf{g}_t(\mathbf{x}_t)\|^2 \\ &= \|\boldsymbol{\lambda}_t\|^2 + 2\mu\boldsymbol{\lambda}_t^\top \mathbf{g}_t(\mathbf{x}_t) + \mu^2\|\mathbf{g}_t(\mathbf{x}_t)\|^2. \end{aligned} \tag{5.47}$$

The proof is complete after dividing both sides by 2.

5.5.1 Proof of Theorem 9

The proof follows the steps in [174, Theorem 7], but generalizes the result from static regret with time-invariant constraints to dynamic regret with time-varying and long-term constraints. Recall that the primal iterate $\mathbf{x}_{t+1}$ is the optimal solution to the following optimization problem (cf. (5.8))

$$\min_{\mathbf{x}\in\mathcal{X}} h_t(\mathbf{x}) := \nabla f_t(\mathbf{x}_t)^\top(\mathbf{x}-\mathbf{x}_t) + \boldsymbol{\lambda}_{t+1}^\top \mathbf{g}_t(\mathbf{x}) + \frac{1}{2\alpha}\|\mathbf{x}-\mathbf{x}_t\|^2. \tag{5.48}$$

Then for any interior point $\tilde{\mathbf{x}}_t \in \mathcal{X}$ in Assumption 8, it follows that

$$\begin{aligned}&\nabla f_t(\mathbf{x}_t)^\top(\mathbf{x}_{t+1}-\mathbf{x}_t) + \boldsymbol{\lambda}_{t+1}^\top \mathbf{g}_t(\mathbf{x}_{t+1}) + \frac{1}{2\alpha}\|\mathbf{x}_{t+1}-\mathbf{x}_t\|^2 \\ &\leq \nabla f_t(\mathbf{x}_t)^\top(\tilde{\mathbf{x}}_t-\mathbf{x}_t) + \boldsymbol{\lambda}_{t+1}^\top \mathbf{g}_t(\tilde{\mathbf{x}}_t) + \frac{1}{2\alpha}\|\tilde{\mathbf{x}}_t-\mathbf{x}_t\|^2 \\ &\overset{(a)}{\leq} \nabla f_t(\mathbf{x}_t)^\top(\tilde{\mathbf{x}}_t-\mathbf{x}_t) - \epsilon\boldsymbol{\lambda}_{t+1}^\top \mathbf{1} + \frac{1}{2\alpha}\|\tilde{\mathbf{x}}_t-\mathbf{x}_t\|^2 \\ &\overset{(b)}{\leq} \nabla f_t(\mathbf{x}_t)^\top(\tilde{\mathbf{x}}_t-\mathbf{x}_t) - \epsilon\|\boldsymbol{\lambda}_{t+1}\| + \frac{1}{2\alpha}\|\tilde{\mathbf{x}}_t-\mathbf{x}_t\|^2 \end{aligned} \tag{5.49}$$

where (a) follows by choosing $\tilde{\mathbf{x}}_t$ such that $\mathbf{g}_t(\tilde{\mathbf{x}}_t) \leq -\epsilon\mathbf{1}$ and recalling the non-negativity of $\boldsymbol{\lambda}_{t+1}$; inequality (b) is because $\|\boldsymbol{\lambda}_{t+1}\| \leq \boldsymbol{\lambda}_{t+1}^\top\mathbf{1}$ holds for any non-negative vector $\boldsymbol{\lambda}_{t+1}$.

Rearranging terms in (5.49), it follows that

$$\begin{aligned}
\boldsymbol{\lambda}_{t+1}^\top\mathbf{g}_t(\mathbf{x}_{t+1}) &\leq \nabla f_t(\mathbf{x}_t)^\top(\tilde{\mathbf{x}}_t-\mathbf{x}_t)-\nabla f_t(\mathbf{x}_t)^\top(\mathbf{x}_{t+1}-\mathbf{x}_t)-\epsilon\|\boldsymbol{\lambda}_{t+1}\|+\frac{1}{2\alpha}\|\tilde{\mathbf{x}}_t-\mathbf{x}_t\|^2-\frac{1}{2\alpha}\|\mathbf{x}_{t+1}-\mathbf{x}_t\|^2\\
&\overset{(c)}{\leq} \nabla f_t(\mathbf{x}_t)^\top(\tilde{\mathbf{x}}_t-\mathbf{x}_t)-\nabla f_t(\mathbf{x}_t)^\top(\mathbf{x}_{t+1}-\mathbf{x}_t)-\epsilon\|\boldsymbol{\lambda}_{t+1}\|+\frac{R^2}{2\alpha}\\
&\overset{(d)}{\leq} \|\nabla f_t(\mathbf{x}_t)\|\|\tilde{\mathbf{x}}_t-\mathbf{x}_t\|+\|\nabla f_t(\mathbf{x}_t)\|\|\mathbf{x}_{t+1}-\mathbf{x}_t\|-\epsilon\|\boldsymbol{\lambda}_{t+1}\|+\frac{R^2}{2\alpha}\\
&\overset{(e)}{\leq} 2GR-\epsilon\|\boldsymbol{\lambda}_{t+1}\|+\frac{R^2}{2\alpha}
\end{aligned}\tag{5.50}$$

where (c) holds since $\mathcal{X}$ confines $\|\tilde{\mathbf{x}}_t-\mathbf{x}_t\|^2 \leq R^2$ and $\|\mathbf{x}_{t+1}-\mathbf{x}_t\|^2 \geq 0$; (d) uses the Cauchy-Schwartz inequality twice; (e) leverages the bounds in Assumption 7, namely, $\|\nabla f_t(\mathbf{x}_t)\| \leq G$, $\|\tilde{\mathbf{x}}_t-\mathbf{x}_t\| \leq R$, and $\|\mathbf{x}_{t+1}-\mathbf{x}_t\| \leq R$.

Plugging (5.50) into (5.46) in Lemma 19, we have

$$\begin{aligned}
\Delta(\boldsymbol{\lambda}_{t+1}) &\leq \mu\boldsymbol{\lambda}_{t+1}^\top\mathbf{g}_{t+1}(\mathbf{x}_{t+1})+\frac{\mu^2}{2}\|\mathbf{g}_{t+1}(\mathbf{x}_{t+1})\|^2\\
&\overset{(f)}{\leq} \mu\boldsymbol{\lambda}_{t+1}^\top\big(\mathbf{g}_{t+1}(\mathbf{x}_{t+1})-\mathbf{g}_t(\mathbf{x}_{t+1})\big)-\epsilon\mu\|\boldsymbol{\lambda}_{t+1}\|+2\mu GR+\frac{\mu R^2}{2\alpha}+\frac{\mu^2M^2}{2}\\
&\overset{(g)}{\leq} \mu\boldsymbol{\lambda}_{t+1}^\top\big[\mathbf{g}_{t+1}(\mathbf{x}_{t+1})-\mathbf{g}_t(\mathbf{x}_{t+1})\big]^+-\epsilon\mu\|\boldsymbol{\lambda}_{t+1}\|+2\mu GR+\frac{\mu R^2}{2\alpha}+\frac{\mu^2M^2}{2}\\
&\overset{(h)}{\leq} \mu\bar{V}(\mathbf{g})\|\boldsymbol{\lambda}_{t+1}\|-\epsilon\mu\|\boldsymbol{\lambda}_{t+1}\|+2\mu GR+\frac{\mu R^2}{2\alpha}+\frac{\mu^2M^2}{2}
\end{aligned}\tag{5.51}$$

where (f) uses the upper bound in Assumption 6 such that $\|\mathbf{g}_{t+1}(\mathbf{x}_{t+1})\| \leq M$, (g) holds since $\boldsymbol{\lambda}_{t+1} \geq \mathbf{0}$, and (h) follows from the Cauchy-Schwartz inequality and the definition of the maximum variation $\bar{V}(\mathbf{g})$ in Assumption 9.

We prove the dual upper bound (5.12) by contradiction. Without loss of generality, suppose that $t+2$ is the first time that (5.12) does not hold. Therefore, we have

$$\|\boldsymbol{\lambda}_{t+1}\| \leq \|\bar{\boldsymbol{\lambda}}\| = \mu M+\frac{2GR+R^2/(2\alpha)+(\mu M^2)/2}{\epsilon-\bar{V}(\mathbf{g})}\tag{5.52a}$$

and correspondingly

$$\|\boldsymbol{\lambda}_{t+2}\| > \|\bar{\boldsymbol{\lambda}}\| = \mu M+\frac{2GR+R^2/(2\alpha)+(\mu M^2)/2}{\epsilon-\bar{V}(\mathbf{g})}.\tag{5.52b}$$

In this case, it follows that

$$
\begin{aligned}
\|\boldsymbol{\lambda}_{t+1}\| &\geq \|\boldsymbol{\lambda}_{t+2}\| - \|\boldsymbol{\lambda}_{t+2} - \boldsymbol{\lambda}_{t+1}\| \\
&= \|\boldsymbol{\lambda}_{t+2}\| - \|[\boldsymbol{\lambda}_{t+1} + \mu \mathbf{g}_{t+1}(\mathbf{x}_{t+1})]^{+} - \boldsymbol{\lambda}_{t+1}\| \\
&\overset{(i)}{\geq} \|\boldsymbol{\lambda}_{t+2}\| - \|\mu \mathbf{g}_{t+1}(\mathbf{x}_{t+1})\| \\
&\overset{(j)}{>} \frac{2GR + R^2/(2\alpha) + (\mu M^2)/2}{\epsilon - \bar{V}(\mathbf{g})}
\end{aligned}
\tag{5.53}
$$

where (i) is due to the non-expansive property of the projection operator, and inequality (j) uses (5.52b) and $\|\mathbf{g}_{t+1}(\mathbf{x}_{t+1})\| \leq M$ in Assumption 6. However, since $\epsilon > \bar{V}(\mathbf{g})$, (5.51) implies that we have $\Delta(\boldsymbol{\lambda}_{t+1}) < 0$ if (5.53) holds. By definition of the dual drift, $\Delta(\boldsymbol{\lambda}_{t+1}) < 0$ implies that $\|\boldsymbol{\lambda}_{t+2}\| < \|\boldsymbol{\lambda}_{t+1}\|$, which contradicts (5.52a) and (5.52b). In addition, observe that the dual variable is initialized by $\boldsymbol{\lambda}_1 = \mathbf{0}$, and consequently $\|\boldsymbol{\lambda}_2\| \leq \mu M$. Therefore, for every t, we have that $\|\boldsymbol{\lambda}_t\| \leq \|\bar{\boldsymbol{\lambda}}\|$ holds.

Using the dual recursion in (5.9), it follows that $\boldsymbol{\lambda}_{T+1} \geq \boldsymbol{\lambda}_T + \mu \mathbf{g}_T(\mathbf{x}_T) \geq \boldsymbol{\lambda}_1 + \sum_{t=1}^{T} \mu \mathbf{g}_t(\mathbf{x}_t)$. Rearranging terms, we have

$$
\sum_{t=1}^{T} \mathbf{g}_t(\mathbf{x}_t) \leq \frac{\boldsymbol{\lambda}_{T+1}}{\mu} - \frac{\boldsymbol{\lambda}_1}{\mu} \leq \frac{\boldsymbol{\lambda}_{T+1}}{\mu}. \tag{5.54}
$$

With $\boldsymbol{\lambda}_{T+1} \geq \mathbf{0}$, (5.54) implies that $\left[\sum_{t=1}^{T} \mathbf{g}_t(\mathbf{x}_t)\right]^{+} \leq \boldsymbol{\lambda}_{T+1}/\mu$, which completes the proof by taking norms on both sides and using the dual upper bound (5.12).

5.5.2 Proof of Theorem 10

With $h_t(\mathbf{x})$ defining the objective in (5.48), it can be shown that $h_t(\mathbf{x})$ is $1/\alpha$-strongly convex, which implies that for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^I$, we have [119, Theorem 2.1.8]

$$
h_t(\mathbf{y}) \geq h_t(\mathbf{x}) + \nabla h_t(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) + \frac{1}{2\alpha}\|\mathbf{y} - \mathbf{x}\|^2. \tag{5.55}
$$

Since $\mathbf{x}_{t+1}$ is the minimizer of the problem $\min_{\mathbf{x} \in \mathcal{X}} h_t(\mathbf{x})$, the optimality condition [13] implies that

$$
\nabla h_t(\mathbf{x}_{t+1})^\top (\mathbf{y} - \mathbf{x}_{t+1}) \geq 0, \quad \forall \mathbf{y} \in \mathcal{X}. \tag{5.56}
$$

Setting $\mathbf{y} = \mathbf{x}_t^*$ and $\mathbf{x} = \mathbf{x}_{t+1}$ in (5.55), we have that (cf. (5.56))

$$h_t(\mathbf{x}_t^*) \geq h_t(\mathbf{x}_{t+1}) + \frac{1}{2\alpha}\|\mathbf{x}_t^* - \mathbf{x}_{t+1}\|^2. \tag{5.57}$$

Hence, replacing $h_t(\mathbf{x})$ with the objective in (5.48) leads to

$$\begin{aligned}&\nabla f_t(\mathbf{x}_t)^\top(\mathbf{x}_{t+1}-\mathbf{x}_t)+\boldsymbol{\lambda}_{t+1}^\top\mathbf{g}_t(\mathbf{x}_{t+1})+\frac{\|\mathbf{x}_{t+1}-\mathbf{x}_t\|^2}{2\alpha}\\ &\overset{(a)}{\leq}\nabla f_t(\mathbf{x}_t)^\top(\mathbf{x}_t^*-\mathbf{x}_t)+\boldsymbol{\lambda}_{t+1}^\top\mathbf{g}_t(\mathbf{x}_t^*)+\frac{\|\mathbf{x}_t^*-\mathbf{x}_t\|^2}{2\alpha}-\frac{\|\mathbf{x}_{t+1}-\mathbf{x}_t^*\|^2}{2\alpha}\end{aligned} \tag{5.58}$$

where (a) uses the strong convexity of the objective in (5.8); see also [174, Corollary 1]. Adding $f_t(\mathbf{x}_t)$ in (5.58) yields

$$\begin{aligned}&f_t(\mathbf{x}_t)+\nabla f_t(\mathbf{x}_t)^\top(\mathbf{x}_{t+1}-\mathbf{x}_t)+\boldsymbol{\lambda}_{t+1}^\top\mathbf{g}_t(\mathbf{x}_{t+1})+\frac{\|\mathbf{x}_{t+1}-\mathbf{x}_t\|^2}{2\alpha}\\ &\leq f_t(\mathbf{x}_t)+\nabla f_t(\mathbf{x}_t)^\top(\mathbf{x}_t^*-\mathbf{x}_t)+\boldsymbol{\lambda}_{t+1}^\top\mathbf{g}_t(\mathbf{x}_t^*)+\frac{\|\mathbf{x}_t^*-\mathbf{x}_t\|^2}{2\alpha}-\frac{\|\mathbf{x}_t^*-\mathbf{x}_{t+1}\|^2}{2\alpha}\\ &\overset{(b)}{\leq} f_t(\mathbf{x}_t^*)+\boldsymbol{\lambda}_{t+1}^\top\mathbf{g}_t(\mathbf{x}_t^*)+\frac{\|\mathbf{x}_t^*-\mathbf{x}_t\|^2}{2\alpha}-\frac{\|\mathbf{x}_t^*-\mathbf{x}_{t+1}\|^2}{2\alpha}\\ &\overset{(c)}{\leq} f_t(\mathbf{x}_t^*)+\frac{\|\mathbf{x}_t^*-\mathbf{x}_t\|^2}{2\alpha}-\frac{\|\mathbf{x}_t^*-\mathbf{x}_{t+1}\|^2}{2\alpha}\end{aligned} \tag{5.59}$$

where (b) is due to the convexity of $f_t(\mathbf{x})$, and (c) comes from the fact that $\boldsymbol{\lambda}_{t+1} \geq \mathbf{0}$ and the per-slot optimal solution $\mathbf{x}_t^*$ is feasible (i.e., $\mathbf{g}_t(\mathbf{x}_t^*) \leq \mathbf{0}$) such that $\boldsymbol{\lambda}_{t+1}^\top\mathbf{g}_t(\mathbf{x}_t^*) \leq 0$.

Next, we bound the term $\nabla f_t(\mathbf{x}_t)^\top(\mathbf{x}_{t+1}-\mathbf{x}_t)$ by

$$\begin{aligned}&-\nabla f_t(\mathbf{x}_t)^\top(\mathbf{x}_{t+1}-\mathbf{x}_t) \leq \|\nabla f_t(\mathbf{x}_t)\|\|\mathbf{x}_{t+1}-\mathbf{x}_t\|\\ &\leq\frac{\|\nabla f_t(\mathbf{x}_t)\|^2}{2\eta}+\frac{\eta}{2}\|\mathbf{x}_{t+1}-\mathbf{x}_t\|^2 \overset{(d)}{\leq} \frac{G^2}{2\eta}+\frac{\eta}{2}\|\mathbf{x}_{t+1}-\mathbf{x}_t\|^2\end{aligned} \tag{5.60}$$

where η is an arbitrary positive constant, and (d) is from the bound of gradients in Assumption 6. Plugging (5.60) into (5.59), we have

$$\begin{aligned}f_t(\mathbf{x}_t)+\boldsymbol{\lambda}_{t+1}^\top\mathbf{g}_t(\mathbf{x}_{t+1}) &\leq f_t(\mathbf{x}_t^*)+\left(\frac{\eta}{2}-\frac{1}{2\alpha}\right)\|\mathbf{x}_{t+1}-\mathbf{x}_t\|^2+\frac{1}{2\alpha}\left(\|\mathbf{x}_t^*-\mathbf{x}_t\|^2-\|\mathbf{x}_t^*-\mathbf{x}_{t+1}\|^2\right)+\frac{G^2}{2\eta}\\ &\overset{(e)}{=} f_t(\mathbf{x}_t^*)+\frac{1}{2\alpha}\left(\|\mathbf{x}_t^*-\mathbf{x}_t\|^2-\|\mathbf{x}_t^*-\mathbf{x}_{t+1}\|^2\right)+\frac{\alpha G^2}{2}\end{aligned} \tag{5.61}$$

where (e) follows by choosing $\eta = 1/\alpha$ so that $\eta/2-1/(2\alpha)=0$.

Using the dual drift bound (5.46) in Lemma 19 again, we have

$$
\begin{aligned}
&\Delta(\boldsymbol{\lambda}_{t+1})/\mu + f_t(\mathbf{x}_t)\\
\leq & f_t(\mathbf{x}_t) + \boldsymbol{\lambda}_{t+1}^\top \mathbf{g}_t(\mathbf{x}_{t+1}) + \boldsymbol{\lambda}_{t+1}^\top \mathbf{g}_{t+1}(\mathbf{x}_{t+1}) - \boldsymbol{\lambda}_{t+1}^\top \mathbf{g}_t(\mathbf{x}_{t+1}) + \frac{\mu}{2}\|\mathbf{g}_{t+1}(\mathbf{x}_{t+1})\|^2\\
\overset{(f)}{\leq} & f_t(\mathbf{x}_t^*) + \frac{1}{2\alpha}\Big(\|\mathbf{x}_t^* - \mathbf{x}_t\|^2 - \|\mathbf{x}_t^* - \mathbf{x}_{t+1}\|^2\Big) + \boldsymbol{\lambda}_{t+1}^\top(\mathbf{g}_{t+1}(\mathbf{x}_{t+1}) - \mathbf{g}_t(\mathbf{x}_{t+1})) + \frac{\mu\|\mathbf{g}_{t+1}(\mathbf{x}_{t+1})\|^2}{2} + \frac{\alpha G^2}{2}\\
\overset{(g)}{\leq} & f_t(\mathbf{x}_t^*) + \frac{1}{2\alpha}\Big(\|\mathbf{x}_t^* - \mathbf{x}_t\|^2 - \|\mathbf{x}_t^* - \mathbf{x}_{t+1}\|^2\Big) + \boldsymbol{\lambda}_{t+1}^\top\left[\mathbf{g}_{t+1}(\mathbf{x}_{t+1}) - \mathbf{g}_t(\mathbf{x}_{t+1})\right]^+ + \frac{\mu M^2}{2} + \frac{\alpha G^2}{2}\\
\overset{(h)}{\leq} & f_t(\mathbf{x}_t^*) + \frac{1}{2\alpha}\Big(\|\mathbf{x}_t^* - \mathbf{x}_t\|^2 - \|\mathbf{x}_t^* - \mathbf{x}_{t+1}\|^2\Big) + \|\boldsymbol{\lambda}_{t+1}\| V(\mathbf{g}_t) + \frac{\mu M^2}{2} + \frac{\alpha G^2}{2} \qquad (5.62)
\end{aligned}
$$

where (f) follows from (5.61); (g) uses non-negativity of $\boldsymbol{\lambda}_{t+1}$ and the gradient upper bound $\|\mathbf{g}_{t+1}(\mathbf{x})\| \leq M, \forall \mathbf{x} \in \mathcal{X}$; and (h) follows from the Cauchy-Schwartz inequality and the definition of the constraint variation $V(\mathbf{g}_t)$ in (5.16).

By interpolating intermediate terms in $\|\mathbf{x}_t^* - \mathbf{x}_t\|^2 - \|\mathbf{x}_t^* - \mathbf{x}_{t+1}\|^2$, we have that

$$
\begin{aligned}
&\|\mathbf{x}_t^* - \mathbf{x}_t\|^2 - \|\mathbf{x}_t^* - \mathbf{x}_{t+1}\|^2\\
=&\|\mathbf{x}_t^* - \mathbf{x}_t\|^2 - \|\mathbf{x}_t - \mathbf{x}_{t-1}^*\|^2 + \|\mathbf{x}_t - \mathbf{x}_{t-1}^*\|^2 - \|\mathbf{x}_t^* - \mathbf{x}_{t+1}\|^2\\
=&\|\mathbf{x}_t^* - \mathbf{x}_{t-1}^*\|\|\mathbf{x}_t^* - 2\mathbf{x}_t + \mathbf{x}_{t-1}^*\| + \|\mathbf{x}_t - \mathbf{x}_{t-1}^*\|^2 - \|\mathbf{x}_t^* - \mathbf{x}_{t+1}\|^2\\
\overset{(i)}{\leq}&2R\|\mathbf{x}_t^* - \mathbf{x}_{t-1}^*\| + \|\mathbf{x}_t - \mathbf{x}_{t-1}^*\|^2 - \|\mathbf{x}_t^* - \mathbf{x}_{t+1}\|^2 \qquad (5.63)
\end{aligned}
$$

where (i) follows from the radius of $\mathcal{X}$ in Assumption 7 such that $\|\mathbf{x}_t^* - 2\mathbf{x}_t + \mathbf{x}_{t-1}^*\| \leq \|\mathbf{x}_t^* - \mathbf{x}_t\| + \|\mathbf{x}_t - \mathbf{x}_{t-1}^*\| \leq 2R$. Plugging (5.63) into (5.62), it readily leads to

$$
\begin{aligned}
\Delta(\boldsymbol{\lambda}_{t+1})/\mu + f_t(\mathbf{x}_t) \leq &f_t(\mathbf{x}_t^*) + \|\boldsymbol{\lambda}_{t+1}\| V(\mathbf{g}_t) + \frac{\mu M^2}{2} + \frac{\alpha G^2}{2}\\
&+ \frac{1}{2\alpha}\Big(2R\|\mathbf{x}_t^* - \mathbf{x}_{t-1}^*\| + \|\mathbf{x}_t - \mathbf{x}_{t-1}^*\|^2 - \|\mathbf{x}_t^* - \mathbf{x}_{t+1}\|^2\Big). \qquad (5.64)
\end{aligned}
$$

Summing up (5.64) over $t = 1, 2, \ldots, T$, we find

$$
\begin{aligned}
&\sum_{t=1}^T \Delta(\boldsymbol{\lambda}_{t+1})/\mu + \sum_{t=1}^T f_t(\mathbf{x}_t)\\
\leq &\sum_{t=1}^T f_t(\mathbf{x}_t^*) + \frac{1}{2\alpha}\sum_{t=1}^T\big(\|\mathbf{x}_t - \mathbf{x}_{t-1}^*\|^2 - \|\mathbf{x}_t^* - \mathbf{x}_{t+1}\|^2\big) + \frac{RV(\{\mathbf{x}_t^*\}_{t=1}^T)}{\alpha} + \sum_{t=1}^T \|\boldsymbol{\lambda}_{t+1}\| V(\mathbf{g}_t) + \frac{\mu M^2 T}{2} + \frac{\alpha G^2 T}{2}
\end{aligned}
$$

$$\overset{(j)}{\leq}\sum_{t=1}^{T}f_t(\mathbf{x}_t^*)+\frac{1}{2\alpha}\left(\|\mathbf{x}_1-\mathbf{x}_0^*\|^2-\|\mathbf{x}_T^*-\mathbf{x}_{T+1}\|^2\right)+\frac{RV(\{\mathbf{x}_t^*\}_{t=1}^T)}{\alpha}+\|\bar{\boldsymbol{\lambda}}\|\sum_{t=1}^{T}V(\mathbf{g}_t)+\frac{\mu M^2T}{2}+\frac{\alpha G^2T}{2}$$

$$\overset{(k)}{\leq}\sum_{t=1}^{T}f_t(\mathbf{x}_t^*)+\frac{1}{2\alpha}\left(\|\mathbf{x}_1-\mathbf{x}_0^*\|^2\right)+\frac{RV(\{\mathbf{x}_t^*\}_{t=1}^T)}{\alpha}+\|\bar{\boldsymbol{\lambda}}\|V(\{\mathbf{g}_t\}_{t=1}^T)+\frac{\mu M^2T}{2}+\frac{\alpha G^2T}{2} \tag{5.65}$$

where (j) uses the upper bound of $\|\boldsymbol{\lambda}_t\|$ in (5.12) that we define as $\|\bar{\boldsymbol{\lambda}}\|$, and (k) follows from the definition of accumulated variations $V(\{\mathbf{g}_t\}_{t=1}^T)$ in (5.16). The definition of dynamic regret in (5.4) finally implies that

$$\begin{aligned}\mathrm{Reg}_T^{\mathrm{d}} &\leq \frac{RV(\{\mathbf{x}_t^*\}_{t=1}^T)}{\alpha}+\frac{\|\mathbf{x}_1-\mathbf{x}_0^*\|^2}{2\alpha}+\|\bar{\boldsymbol{\lambda}}\|V(\{\mathbf{g}_t\}_{t=1}^T)+\frac{\mu M^2T}{2}+\frac{\alpha G^2T}{2}-\sum_{t=1}^{T}\frac{\Delta(\boldsymbol{\lambda}_{t+1})}{\mu}\\ &=\frac{RV(\{\mathbf{x}_t^*\}_{t=1}^T)}{\alpha}+\frac{\|\mathbf{x}_1-\mathbf{x}_0^*\|^2}{2\alpha}+\|\bar{\boldsymbol{\lambda}}\|V(\{\mathbf{g}_t\}_{t=1}^T)+\frac{\mu M^2T}{2}+\frac{\alpha G^2T}{2}-\frac{\|\boldsymbol{\lambda}_{T+2}\|^2}{2\mu}+\frac{\|\boldsymbol{\lambda}_2\|^2}{2\mu}\\ &\overset{(l)}{\leq}\frac{RV(\{\mathbf{x}_t^*\}_{t=1}^T)}{\alpha}+\frac{R^2}{2\alpha}+\|\bar{\boldsymbol{\lambda}}\|V(\{\mathbf{g}_t\}_{t=1}^T)+\frac{\mu M^2T}{2}+\frac{\alpha G^2T}{2}+\frac{\mu M^2}{2}\end{aligned} \tag{5.66}$$

where (l) follows since: i) $\|\mathbf{x}_1-\mathbf{x}_0^*\|\leq R$ due to the compactness of $\mathcal{X}$; ii) $\|\boldsymbol{\lambda}_{T+2}\|^2\geq 0$; and, iii) $\|\boldsymbol{\lambda}_2\|^2\leq \mu^2M^2$ if $\boldsymbol{\lambda}_1=\mathbf{0}$. This completes the proof.

Chapter 6

Model-free interactive optimization for mobile edge computing

6.1 Introduction

Internet-of-Things (IoT) envisions an intelligent infrastructure of networked smart devices offering task-specific monitoring and control services [134]. Leveraging advances in embedded systems, contemporary IoT devices are featured with *small-size* and *low-power* designs, but their computation and communication capabilities are limited. A prevalent solution during the past decade was to move computing, control, and storage resources to the remote cloud (a.k.a. data centers). Yet, the cloud-based IoT architecture is challenged by high latency due to directly communications with the cloud, which certainly prevents real-time applications [38]. Along with other features of IoT, such as *extreme heterogeneity* and *unpredictable dynamics*, the need arises for innovations in network design and management to allow for adaptive online service provisioning, subject to stringent delay constraints [86].

From the network design vantage point, *fog* is viewed as a promising architecture for IoT that distributes computation, communication, and storage closer to the end IoT users, along the cloud-to-things continuum [38]. In the fog computing paradigm, service provisioning starts at the network edge, e.g., smartphones, and high-tech routers, and only a portion of tasks will be offloaded to the powerful cloud for further processing (a.k.a. computation offloading) [133, 105, 166]. Existing approaches for computation offloading either focus on time-invariant static settings, or, rely on stochastic optimization approaches such as Lyapunov optimization to deal with time-varying cases; see [107] and references therein. Nevertheless, static settings cannot capture the changing

IoT environment, and the stationarity commonly assumed in stochastic optimization literature may not hold in practice, especially when the stochastic process involves human participation as in IoT. From the management perspective, online network control, which is robust to non-stationary dynamics and amenable to low-complexity implementations, remains an uncharted territory [105, 107].

Indeed, the *primary goal* of this paper is an algorithmic pursuit of online network optimization suitable for emerging tasks in IoT. Focusing on such algorithmic challenges, online convex optimization (OCO) is a promising methodology for sequential tasks with well-documented merits, especially when the sequence of convex costs varies in an unknown and possibly adversarial manner [185]. Aiming to empower traditional fog management policies with OCO, most available OCO works benchmark algorithms with a *static regret*, which measures the difference of costs (a.k.a. losses) between the online solution and the best static solution in hindsight [65, 48]. However, static regret is not a comprehensive performance metric in dynamic settings such as those encountered with IoT [71].

6.1.1 Prior art

Recent works extend the analysis of static to that of *dynamic regret* [62, 71], but they deal with time-invariant constraints that cannot be violated instantaneously. Tailored for fog computing setups that need flexible adaptation of online decisions to dynamic resource availability, OCO with time-varying constraints was first studied in [25], along with its adaptive variant in [27], and the optimal regret bound in this setting was first established in [117]. Yet, the approaches in [27, 25, 117] remain operational under the premise that the loss functions are *explicitly known*, or, their gradients are readily available. Clearly, none of these two assumptions can be easily satisfied in IoT settings, because i) the loss function capturing user dissatisfaction, e.g., service latency or reliability, is hard to model in dynamic environments; and, ii) even if modeling is possible in theory, the low-power IoT devices may not afford the complexity of running statistical learning tools such as deep neural networks online.

In this context, targeting a gradient-free efficient solution, alternative online schemes have been advocated leveraging point-wise values of loss functions (partial-information feedback) rather than their gradients (full-information feedback). They are termed bandit convex optimization (BCO) in machine learning [52, 3, 143, 22], or referred as zeroth-order schemes in optimization circles [50, 120]. While [52, 3, 50, 143, 22, 120] employed on BCO with time-invariant constraints that cannot be violated instantaneously, the *long-term* effect of such instantaneous violations

Table 6.1: A summary of related works on OCO/BCO

Reference	Benchmark	Constraints	Feedback
[185, 65, 48]	Static	Fixed and strict	Gradient
[62, 71]	Dynamic	Fixed and strict	Gradient
[117]	Static	Varying and long-term	Gradient
[25, 27]	Dynamic	Varying and long-term	Gradient
[106]	Static	Fixed and long-term	Grad./Fun. value
[52, 3, 50, 120, 143, 22]	Static	Fixed and strict	Function value
This work	Dynamic	Varying and long-term	Function value

was studied in [106], where the focus is still on static regret and time-invariant constraints. Nevertheless, [52, 3, 50, 143, 22, 120] cannot be implemented without knowing the instantaneous constraints, and the performance guarantees relative to the best dynamic benchmark have not been characterized in [52, 3, 50, 143, 22, 120, 106].

6.1.2 Our contributions

Building on full-information precursors [25, 27, 117], the present paper broadens the scope of BCO to the regime with *time-varying constraints*, and proposes a class of online algorithms termed online bandit saddle-point (BanSaP) approaches. Also worth mentioning is that the regret-fit tradeoff of BanSaP markedly improves that in [27] for the special case with full-information feedback, and that in [106] for the special case with time-invariant constraints. With an eye on managing IoT with limited information, our contribution is the incorporation of long-term and time-varying constraints to expand the scope of BCO; see a summary in Table 6.1.

In a nutshell, relative to existing works, the main contributions of the present paper are summarized as follows.

c1) We generalize the standard BCO framework with only time-varying costs [52, 3], to account for both time-varying costs and constraints. Performance here is established relative to the best dynamic benchmark, via metrics that we term dynamic regret and fit (Section III).

c2) We develop a class of BanSaP algorithms to tackle this novel BCO problem, and analytically establish that BanSaP solvers yield simultaneously optimal sub-linear dynamic regret and fit, given that the accumulated variations of per-slot minimizers are known to grow sub-linearly with time (Section IV).

c3) Our BanSaP algorithms are applied to computation offloading tasks emerging in IoT management, and simulations under various network sizes further demonstrate that the BanSaP solvers outperform the popular algorithm with bandit feedback, and have comparable performance relative to full-information alternatives (Section V).

Notation. $(\cdot)^\top$ stands for vector and matrix transposition, and $\|\mathbf{x}\|$ denotes the ℓ_2-norm of a vector $\mathbf{x}$. Inequalities for vectors $\mathbf{x} > \mathbf{0}$, and the projection $[\mathbf{a}]^+ := \max\{\mathbf{a}, \mathbf{0}\}$ are entry-wise.

6.2 Bandit online learning with constraints

In this section, a generic BCO formulation with long-term and time-varying constraints will be introduced, along with its real-world application in IoT management.

6.2.1 Online learning with constraints under partial feedback

Before introducing BCO with long-term constraints, we begin with the classical BCO setting, where constraints are time-invariant, and must be strictly satisfied [52, 3, 22]. Akin to its full-information counterpart [185, 65], BCO can be viewed as a repeated game between a learner and nature. Consider that time is discrete and indexed by t. Per slot t, a learner selects an action $\mathbf{x}_t$ from a convex set $\mathcal{X} \subseteq \mathbb{R}^d$, and subsequently nature chooses a loss function $f_t(\cdot) : \mathbb{R}^d \to \mathbb{R}$ through which the learner incurs a loss $f_t(\mathbf{x}_t)$. The convex feasible set $\mathcal{X}$ is a-priori known and fixed over the entire time horizon. Different from the OCO setup, at the end of each slot, only the value of $f_t(\mathbf{x}_t)$ rather than the form of $f_t(\mathbf{x})$ is revealed to the learner in BCO. Although this standard BCO setting is appealing to various applications such as online end-to-end routing [8] and task assignment [75], it does not account for potential variations of (possibly unknown) constraints, and does not deal with constraints that can possibly be satisfied in the long term rather than a slot-by-slot basis [106, 25, 117].

Online optimization with time-varying and long-term constraints is well motivated for applications from power control in wireless communication [116], geographical load balancing in cloud networks [26, 25], to computation offloading in fog computing [135, 37]. Motivated by these dynamic network management tasks, our recent works [25, 27] studied OCO with time-varying constraints in *full information* setting, where the gradient feedback is available. Complementing [25] and [27], the present paper broadens the applicability of BCO to the regime with time-varying long-term constraints.

Specifically, we consider that per slot t, a learner selects an action $\mathbf{x}_t$ from a known and fixed convex set $\mathcal{X} \subseteq \mathbb{R}^d$, and then nature chooses not only a loss function $f_t(\cdot) : \mathbb{R}^d \to \mathbb{R}$, but also a time-varying penalty function $\mathbf{g}_t(\cdot) : \mathbb{R}^d \to \mathbb{R}^N$. The later gives rise to the time-varying constraint $\mathbf{g}_t(\mathbf{x}_t) \leq \mathbf{0}$, which is driven by the unknown application-specific dynamics. Similar to the standard BCO setting, only the value of $f_t(\mathbf{x}_t)$ at the queried point $\mathbf{x}_t$ is revealed to the learner here; but different from the standard BCO setting, besides $\mathcal{X}$, the constraint $\mathbf{g}_t(\mathbf{x}_t) \leq \mathbf{0}$ needs to be carefully taken care of. And the fact that $\mathbf{g}_t$ is unknown to the learner when performing her/his decision, makes it impossible to satisfy in every time slot. Hence, a more realistic goal here is to find a sequence of solutions $\{\mathbf{x}_t\}$ that minimizes the aggregate loss, and ensures that the constraints $\{\mathbf{g}_t(\mathbf{x}_t) \leq \mathbf{0}\}$ are satisfied in the long term on average. Specifically, extending the BCO framework [52, 3, 143] to accommodate such time-varying constraints, we consider the following online optimization problem

$$\boxed{\min_{\{\mathbf{x}_t \in \mathcal{X}, \forall t\}} \sum_{t=1}^{T} f_t(\mathbf{x}_t) \quad \text{s. to} \sum_{t=1}^{T} \mathbf{g}_t(\mathbf{x}_t) \leq \mathbf{0}} \tag{6.1}$$

where T is the entire time horizon, $\mathbf{x}_t \in \mathbb{R}^d$ is the decision variable, f_t represents the cost function, $\mathbf{g}_t := [g_t^1, \dots, g_t^N]^\top$ denotes the constraint function with nth entry $g_t^n(\cdot) : \mathbb{R}^d \to \mathbb{R}$, and $\mathcal{X} \in \mathbb{R}^d$ is a convex set. In the current setting, we assume that only the values of loss function are available at queried points since e.g., its complete form related to user experience is hard to approximate, but the constraint function is revealed to the learner as it represents measurable physical requirements e.g., power budget, and data flow conservation constraints. Before the algorithm development in Section 6.3 and performance analysis in Section 6.4, we will introduce a motivating example of fog computing in IoT.

6.2.2 Motivating setup: mobile fog computing in IoT

The online computational offloading task of fog computing in IoT [107, 105, 133] takes the form of BCO with long-term constraints (6.1). Consider a mobile network with a sensor layer, a fog layer, and a cloud layer [86, 38]. The sensor layer contains heterogeneous low-power IoT devices (e.g., wearable watches and smart cameras), which do not have enough computational capability, and usually offload their collected data to the local fog nodes (e.g., smartphones and high-tech routers) in the fog layer for further processing [67]. The fog layer consists of N nodes in the set $\mathcal{N} := \{1, \dots, N\}$ with moderate processing capability; thus, part of workloads will be collaboratively processed by the local fog servers to meet the stringent latency requirement, and the rest will be offloaded to the remote data center in the cloud layer [105]; also see Fig. 6.1.

Per time t, each fog node n collects data requests b_t^n from all its nearby sensors. Once receiving these requests, node n has *three options*: i) offloading the amount z_t^n to the remote data center; ii) offloading the amount y_t^{nk} to each of its nearby node k for collaborative computing; and, iii)

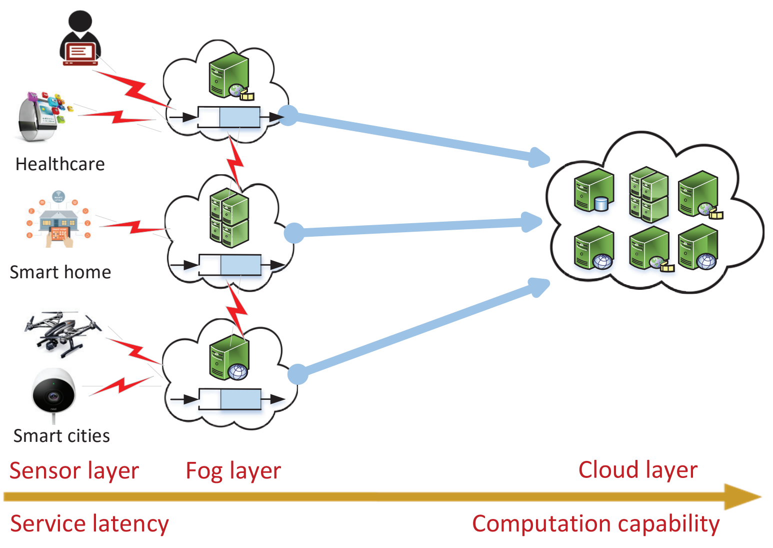

Figure 6.1: A diagram of hierarchical fog computing framework.

locally processing the amount y_t^{nn} according to its resource availability. The optimization variable $\mathbf{x}_t$ in this case consists of the cloud offloading, local offloading, and local processing amounts; i.e., $\mathbf{x}_t := [z_t^1, \ldots, z_t^N, y_t^{11}, \ldots, y_t^{1N}, \ldots, y_t^{N1}, \ldots, y_t^{NN}]^\top$. Assuming that each fog node has a data queue to buffer unserved workloads, the instantaneously served workloads (offloading plus processing) is not necessarily equal to the data arrival rate. Instead, a long-term constraint is common to ensure that the cumulative amount of served workloads is no less than the arrived amount at each node n over time [116]

$$\sum_{t=1}^{T} g_t^n(\mathbf{x}_t) := \sum_{t=1}^{T} \left(b_t^n + \sum_{k \in \mathcal{N}_n^{\text{in}}} y_t^{kn} - \sum_{k \in \mathcal{N}_n^{\text{out}}} y_t^{nk} - z_t^n - y_t^{nn} \right) \leq 0 \tag{6.2}$$

where $\mathcal{N}_n^{\text{in}}$ and $\mathcal{N}_n^{\text{out}}$ represent the sets of fog nodes with in-coming links to node n and those with out-going links from node n, respectively. The bandwidth limit of communication link (e.g., wireline) from fog node n to the remote cloud is $\bar{z}^n$; the limit of the transmission link (e.g., wireless) from node n to its neighbor k is $\bar{y}^{nk}$, and the computation capability of node n is $\bar{y}^{nn}$. With $\bar{\mathbf{x}}$ collecting all the aforementioned limits, the feasible region can be expressed by $\mathbf{x}_t \in \mathcal{X} := \{\mathbf{0} \leq \mathbf{x}_t \leq \bar{\mathbf{x}}\}$.

Performance is assessed by the user dissatisfaction of the online processing and offloading decisions, e.g., aggregate delay [134, 86]. Specifically, as the computation delay is usually negligible for data centers with thousands of high-performance servers, the latency for cloud offloading amount z_t^n is mainly due to the communication delay, which is denoted as a time-varying cost $c_t^n(z_t^n)$ depending on the unpredictable network congestion during slot t. Likewise, the communication delay of the local offloading decision y_t^{nk} from node n to a nearby node k is denoted as $c_t^{nk}(y_t^{nk})$, but its magnitude is much lower than that of cloud offloading. Regarding the processing amount y_t^{nn}, its latency comes from the computation delay due to its limited computational capability, which is presented as a time-varying function $h_t^n(y_t^{nn})$ capturing the dynamic CPU capability during the computing processes. Per slot t, the network delay $f_t(\mathbf{x}_t)$ aggregates the computation delay at all nodes plus the communication delay at all links, namely

$$f_t(\mathbf{x}_t) := \sum_{n\in\mathcal{N}} \Bigg(\underbrace{c_t^n(z_t^n) + \textstyle\sum_{k\in\mathcal{N}_n^{\rm out}} c_t^{nk}(y_t^{nk})}_{\rm communication} + \underbrace{h_t^n(y_t^{nn})}_{\rm computation} \Bigg). \tag{6.3}$$

Clearly, the explicit form of functions $c_t^n(\cdot)$, $c_t^{nk}(\cdot)$, and $h_t^n(\cdot)$ is *unknown* to the network operator due to the unpredictable traffic patterns [8]; but they are convex (thus $f_t(\mathbf{x}_t)$ is convex) with respect to their arguments, which implies that the marginal computation/communication latency is increasing as the offloading/processing amount grows.

Aiming to minimize the accumulated network delay while serving all the IoT workloads in the long term, the optimal offloading strategy in this mobile network is the solution of the following online optimization problem (cf. (6.3))

$$\min_{\{\mathbf{x}_t\in\mathcal{X},\forall t\}} \sum_{t=1}^{T} f_t(\mathbf{x}_t), \text{ s. to (6.2) for } n=1,\ldots,N. \tag{6.4}$$

Comparing to the generic form (6.1), we consider an online fog computing problem in (6.4), where the loss (network latency) function $f_t(\cdot)$ and the data requests $\{b_t^n\}$ within slot t are not known when making the offloading and local processing decision $\mathbf{x}_t$; after performing $\mathbf{x}_t$, only the value of $f_t(\mathbf{x}_t)$ (a.k.a. loss) as well as the measurements $\{b_t^n\}$ are revealed to the network operator. In other words, knowledge of the network operator is fully causal, meaning that before deciding $\mathbf{x}_t$ at time t, the operator knows only $\{f_\tau(\mathbf{x}_\tau), \{b_\tau^n\}\}_{\tau=1}^{t-1}$. Note that in this example, measuring $\{b_t^n\}$ is tantamount to knowing the function $g_t^n(\cdot)$ in (6.2). Therefore, (6.4) is in the form of (6.1).

6.3 BanSaP: Bandit saddle-point methods

To solve the problem in Section 6.2, an online saddle-point method is revisited first, before developing its bandit variants for network optimization with only partial feedback.

6.3.1 Online saddle-point approach with gradient feedback

Several works have studied the OCO setup with time-varying long-term constraints (cf. (6.1)), including [25, 117], and the recent variant [27] incorporating with adaptive stepsizes. Consider now the per-slot problem (6.1), which contains the current objective $f_t(\mathbf{x})$, the current constraint $\mathbf{g}_t(\mathbf{x}) \leq \mathbf{0}$, and a time-invariant feasible set $\mathcal{X}$. With $\boldsymbol{\lambda} \in \mathbb{R}_+^N$ denoting the Lagrange multiplier associated with the time-varying constraint, the online Lagrangian of (6.1) can be expressed as

$$\mathcal{L}_t(\mathbf{x}, \boldsymbol{\lambda}) := f_t(\mathbf{x}) + \boldsymbol{\lambda}^\top \mathbf{g}_t(\mathbf{x}). \tag{6.5}$$

Serving as a basis for developing the bandit approaches, we next revisit the online saddle-point scheme with full-information [117]. Specifically, given the primal iterate $\mathbf{x}_t$ and the dual iterate $\boldsymbol{\lambda}_t$ at each slot t, the next decision $\mathbf{x}_{t+1}$ is generated by

$$\mathbf{x}_{t+1} \in \arg\min_{\mathbf{x}\in\mathcal{X}} \nabla_{\mathbf{x}}^\top \mathcal{L}_t(\mathbf{x}_t, \boldsymbol{\lambda}_t)(\mathbf{x} - \mathbf{x}_t) + \frac{1}{2\alpha} \left\|\mathbf{x} - \mathbf{x}_t\right\|^2 \tag{6.6}$$

where α is a pre-defined constant, and $\nabla_{\mathbf{x}} \mathcal{L}_t(\mathbf{x}_t, \boldsymbol{\lambda}_t) = \nabla f_t(\mathbf{x}_t) + \nabla^\top \mathbf{g}_t(\mathbf{x}_t)\boldsymbol{\lambda}_t$ is the gradient of $\mathcal{L}_t(\mathbf{x}, \boldsymbol{\lambda}_t)$ with respect to (w.r.t.) the primal variable $\mathbf{x}$ at $\mathbf{x} = \mathbf{x}_t$. The minimization (6.6) admits the closed-form solution, given by

$$\mathbf{x}_{t+1} = \mathcal{P}_{\mathcal{X}}(\mathbf{x}_t - \alpha \nabla_{\mathbf{x}} \mathcal{L}_t(\mathbf{x}_t, \boldsymbol{\lambda}_t)) \tag{6.7}$$

where $\mathcal{P}_{\mathcal{X}}(\mathbf{y}) := \arg\min_{\mathbf{x}\in\mathcal{X}} \|\mathbf{x} - \mathbf{y}\|^2$ denotes the projection operator. In addition, the dual update takes the modified online gradient ascent form

$$\boldsymbol{\lambda}_{t+1} = \Big[\boldsymbol{\lambda}_t + \mu(\mathbf{g}_t(\mathbf{x}_t) + \nabla^\top \mathbf{g}_t(\mathbf{x}_t)(\mathbf{x}_{t+1} - \mathbf{x}_t))\Big]^+ \tag{6.8}$$

where μ is a positive stepsize, $[\,\cdot\,]^+$ represents projection to the positive orthant, and $\nabla_{\boldsymbol{\lambda}} \mathcal{L}_t(\mathbf{x}_t, \boldsymbol{\lambda}_t) = \mathbf{g}_t(\mathbf{x}_t)$ is the gradient of $\mathcal{L}_t(\mathbf{x}_t, \boldsymbol{\lambda})$ w.r.t. $\boldsymbol{\lambda}$ at $\boldsymbol{\lambda} = \boldsymbol{\lambda}_t$. Note that (6.8) is a modified gradient update since the dual variable is updated along the first-order approximation of $\mathbf{g}_t(\mathbf{x}_{t+1})$ at the previous iterate $\mathbf{x}_t$ rather than commonly used $\mathbf{g}_t(\mathbf{x}_t)$, which will be critical in our subsequent analytical derivations.

To perform the online saddle-point recursion (6.7)-(6.8) however, the gradient $\nabla f_t(\mathbf{x})$ and the constraint $\mathbf{g}_t(\mathbf{x})$ should be known to the learner at each slot t. When the gradient of $f_t(\mathbf{x})$ (or its explicit form) is unknown as it is in our setup, additional effort is needed. In this context, the systematic design of the online *bandit* saddle-point (BanSaP) methods will be leveraged to

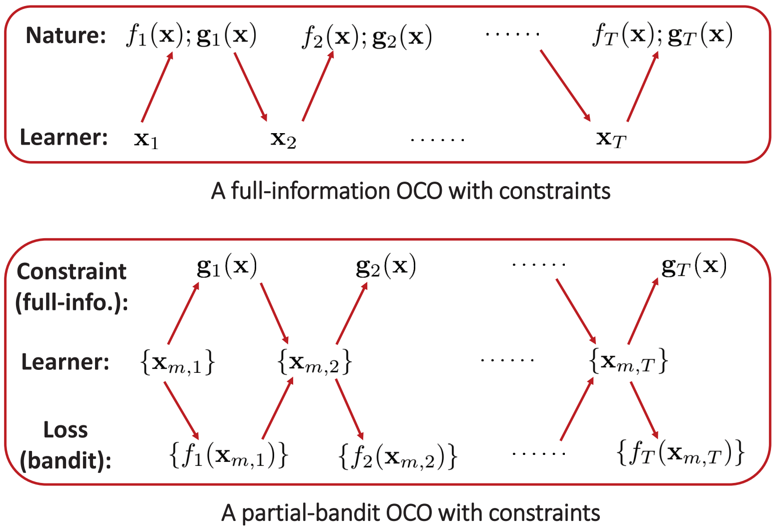
Nature: $f_1(\mathbf{x}); \mathbf{g}_1(\mathbf{x})$ $f_2(\mathbf{x}); \mathbf{g}_2(\mathbf{x})$ ······ $f_T(\mathbf{x}); \mathbf{g}_T(\mathbf{x})$
Learner: $\mathbf{x}_1$ $\mathbf{x}_2$ ······ $\mathbf{x}_T$
A full-information OCO with constraints
Constraint (full-info.): $\mathbf{g}_1(\mathbf{x})$ $\mathbf{g}_2(\mathbf{x})$ ······ $\mathbf{g}_T(\mathbf{x})$
Learner: $\{\mathbf{x}_{m,1}\}$ $\{\mathbf{x}_{m,2}\}$ ······ $\{\mathbf{x}_{m,T}\}$
Loss (bandit): $\{f_1(\mathbf{x}_{m,1})\}$ $\{f_2(\mathbf{x}_{m,2})\}$ ······ $\{f_T(\mathbf{x}_{m,T})\}$
A partial-bandit OCO with constraints

Figure 6.2: A comparison of OCO with full/partial-bandit feedback.

extend the online saddle-point method to the regime where gradient information is unavailable or computationally costly.

6.3.2 BanSaP with one-point partial feedback

The key idea behind BCO is to construct (possibly stochastic) gradient estimates using the limited *function value* information [52, 3, 50, 120, 143]. Depending on system variability, the online learner can afford one or multiple loss function evaluations (partial-information feedback) per time slot [120, 3, 50]. Intuitively, the performance of a bandit algorithm will improve if multiple evaluations are available per time slot; see Fig. 6.2 for a comparison of full- versus partial-information feedback settings.

To begin with, we consider the case where the learner can only observe the function value of $f_t(\mathbf{x})$ at a single point per slot t. The crux here is to construct a (possibly unbiased) estimate of the gradient using this single piece of feedback. Interestingly though, a stochastic gradient estimate of $f_t(\mathbf{x})$ can be obtained by one point *random* function evaluation [52]. The intuition can be readily revealed from the one-dimensional case ($d = 1$): For a binary random variable u taking values $\{-1, 1\}$ equiprobable, and a small constant $\delta > 0$, the idea of forward differentiation implies that the derivative f_t' at x can be approximated by

$$f_t'(x) \approx \frac{f_t(x+\delta) - f_t(x-\delta)}{2\delta} = \mathbb{E}_u \left[\frac{u}{\delta} f_t(x+\delta u)\right] \tag{6.9}$$

where the approximation is due to $\delta > 0$, and the equality follows from the definition of expectation. Hence, $f_t(x+\delta u)u/\delta$ can serve as a stochastic estimator of $f_t'(x)$ based only single function evaluation $f_t(x+\delta u)$. Generalizing this approximation to high dimensions, with a random vector $\mathbf{u}$ drawn from the unit sphere (a.k.a. the surface of a unit ball), the scaled function evaluation at a perturbed point $\mathbf{x}+\delta\mathbf{u}$ yields an estimate of the gradient $\nabla f_t(\mathbf{x})$, given by [52]

$$\nabla f_t(\mathbf{x}) \approx \mathbb{E}_{\mathbf{u}}\left[\frac{d}{\delta} f_t(\mathbf{x}+\delta\mathbf{u})\mathbf{u}\right] := \mathbb{E}_{\mathbf{u}}\left[\hat{\nabla}^1 f_t(\mathbf{x})\right] \tag{6.10}$$

where we define one-point gradient $\hat{\nabla}^1 f_t(\mathbf{x}) := \frac{d}{\delta} f_t(\mathbf{x}+\delta\mathbf{u})\mathbf{u}$.

Building upon this intuition, consider a bandit version of the online saddle-point iteration, for which the primal update becomes (cf. (6.7))

$$\hat{\mathbf{x}}_{t+1} = \mathcal{P}_{(1-\gamma)\mathcal{X}}\Big(\hat{\mathbf{x}}_t - \alpha\hat{\nabla}^1_{\mathbf{x}}\mathcal{L}_t(\hat{\mathbf{x}}_t, \boldsymbol{\lambda}_t)\Big) \tag{6.11}$$

where $(1-\gamma)\mathcal{X} := \{(1-\gamma)\mathbf{x} : \mathbf{x} \in \mathcal{X}\}$ is a subset of $\mathcal{X}$, $\gamma \in [0,1)$ is a pre-selected constant depending on δ, and the one-point Langragian gradient is given by (cf. (6.10))

$$\hat{\nabla}^1_{\mathbf{x}}\mathcal{L}_t(\hat{\mathbf{x}}_t, \boldsymbol{\lambda}_t) := \hat{\nabla}^1 f_t(\hat{\mathbf{x}}_t) + \nabla^\top \mathbf{g}_t(\hat{\mathbf{x}}_t)\boldsymbol{\lambda}_t. \tag{6.12}$$

In the full-information case, $\mathbf{x}_t$ in (6.7) is the learner's action, but in the bandit case the learner's action is $\mathbf{x}_{1,t} := \hat{\mathbf{x}}_t + \delta\mathbf{u}_t$, which is the point for function evaluation but not $\hat{\mathbf{x}}_t$ in (6.11). Note that while the random perturbation $\mathbf{u}_t$ is assumed to lie on the surface of a unit ball, we do not confine the actual IoT decision $\mathbf{x}_t$ to follow any specific distribution.

Furthermore, the projection is performed on a smaller convex set $(1-\gamma)\mathcal{X}$ in (6.11), which ensures feasibility of the perturbed $\mathbf{x}_{1,t} \in \mathcal{X}$. Similar to the full-information case (6.8), the dual update of BanSaP is given by

$$\boldsymbol{\lambda}_{t+1} = \Big[\boldsymbol{\lambda}_t + \mu(\mathbf{g}_t(\hat{\mathbf{x}}_t) + \nabla^\top \mathbf{g}_t(\hat{\mathbf{x}}_t)(\hat{\mathbf{x}}_{t+1} - \hat{\mathbf{x}}_t))\Big]^+ \tag{6.13}$$

where μ is again the stepsize, and the learning iterate $\hat{\mathbf{x}}_t$ rather than the actual decision $\mathbf{x}_t$ is used in this update. Compared with the gradient-based recursions (6.7)-(6.8), the updates (6.11)-(6.13) with one-point bandit feedback do not increase computation or memory requirements, and thus provide a light-weight surrogate for gradient-free online bandit network optimization.

Algorithm 8 BanSaP for OCO with time-varying constraints

```
1: Initialize: primal iterate $\hat{\mathbf{x}}_1$, dual iterate $\boldsymbol{\lambda}_1$, parameters $\delta$ and $\gamma$, and stepsizes $\alpha$ and $\mu$.
2: for $t = 1, 2 \ldots$ do
3:     The learner plays the perturbed actions $\{\mathbf{x}_{m,t}\}_{m=1}^{M}$ based on the learning iterate $\hat{\mathbf{x}}_t$.
4:     The nature reveals the losses $\{f_t(\mathbf{x}_{m,t})\}_{m=1}^{M}$ at queried points, and the constraint $\mathbf{g}_t(\mathbf{x})$.
5:     The learner updates the primal variable $\hat{\mathbf{x}}_{t+1}$ by (6.11) with the gradient estimated by (6.12) for $M = 1$, or, (6.15) for $M = 2$, otherwise, by (6.17).
6:     The learner updates the dual variable $\boldsymbol{\lambda}_{t+1}$ via (6.13).
7: end for
```

6.3.3 BanSaP with multipoint partial feedback

Featuring a simple update given minimal information, the BanSaP with one-point bandit feedback is suitable for fast-varying environments, where multiple function evaluations are impossible. As shown later in Sections 6.4 and 6.5, the theoretical and empirical performance of BanSaP with single-point evaluation is degraded relative to the full-information case.

To improve the performance of BanSaP with one-point feedback, we will first rely on two-point function evaluation at each slot [50], and then generalize to multipoint evaluation. Intuitively, this approach is justified when the underlying dynamics are slow, e.g., when the load and price profiles in power grids are piece-wise stationary. In this case, each slot can be further divided into multiple mini-slots, and one query is performed per mini-slot, over which the loss function and the constraints do not change. Compared to (6.11)-(6.13), the key difference is that the one-point estimate in (6.12) is replaced by

$$\hat{\nabla}^2 f_t(\hat{\mathbf{x}}_t) := \frac{d}{2\delta}\big(f_t(\hat{\mathbf{x}}_t + \delta\mathbf{u}_t) - f_t(\hat{\mathbf{x}}_t - \delta\mathbf{u}_t)\big)\mathbf{u}_t \tag{6.14}$$

where the function values are evaluated on two points around the learning iterate $\hat{\mathbf{x}}_t$, namely, $\mathbf{x}_{1,t} := \hat{\mathbf{x}}_t + \delta\mathbf{u}_t$ and $\mathbf{x}_{2,t} := \hat{\mathbf{x}}_t - \delta\mathbf{u}_t$ with $\mathbf{u}_t$ again drawn uniformly from the unit sphere $\mathbb{S} := \{\mathbf{u} \in \mathbb{R}^d : \|\mathbf{u}\| = 1\}$. The primal update becomes $\hat{\mathbf{x}}_{t+1} = \mathcal{P}_{(1-\gamma)\mathcal{X}}\big(\hat{\mathbf{x}}_t - \alpha\hat{\nabla}^2_{\mathbf{x}}\mathcal{L}_t(\hat{\mathbf{x}}_t, \boldsymbol{\lambda}_t)\big)$, with Lagrangian gradient

$$\hat{\nabla}^2_{\mathbf{x}}\mathcal{L}_t(\hat{\mathbf{x}}_t, \boldsymbol{\lambda}_t) := \hat{\nabla}^2 f_t(\hat{\mathbf{x}}_t) + \nabla^\top \mathbf{g}_t(\hat{\mathbf{x}}_t)\boldsymbol{\lambda}_t. \tag{6.15}$$

Similar to the one-point case, it is instructive to consider the two-point gradient estimate in the one-dimensional case ($d = 1$), where the expectation of the differentiation term in (6.14) approximates well the derivative of f_t at $\hat{x}_t$; that is,

$$\begin{aligned}&\mathbb{E}_u\left[\frac{u_t}{2\delta}\left(f_t(\hat{x}_t+\delta u_t)-f_t(\hat{x}_t-\delta u_t)\right)\right]\\=&\frac{1}{2\delta}\left(f_t(\hat{x}_t+\delta)-f_t(\hat{x}_t-\delta)\right)\approx f_t'(\hat{x}_t)\end{aligned} \tag{6.16}$$

where the equality follows because the random variable u_t takes values $\{-1,1\}$ equiprobable.

Relative to the one-point feedback case, the advantage of the two-point feedback is variance reduction in the gradient estimator. Specifically, the second moment of the stochastic gradient can be uniformly bounded, $\mathbb{E}[\|\frac{d}{2\delta}\big(f_t(\hat{\mathbf{x}}_t+\delta\mathbf{u}_t)-f_t(\hat{\mathbf{x}}_t-\delta\mathbf{u}_t)\big)\mathbf{u}_t\|^2]\leq d^2G^2$, where G is the Lipschitz constant of $f_t(\mathbf{x})$. This is in contrast to the one-point feedback where the second moment is inversely proportional to δ, since $\mathbb{E}[\frac{d}{\delta}\|f_t(\hat{\mathbf{x}}_t+\delta\mathbf{u}_t)\mathbf{u}_t\|^2]\leq d^2F^2/\delta^2$, with F denoting an upper-bound of $f_t(\mathbf{x})$. The proof of this argument can be found in the [33, Lemma 2]. In fact, a bias-variance tradeoff emerges in the one-point case, but not in the two-point case. This subtle yet critical difference will be responsible for an improved performance of BanSaP with two-point feedback, and its stable empirical performance, as will be seen later.

With the insights gained so far, the next step is to endow the BanSaP with more than two function evaluations [3]. With $M>2$ points, the gradient estimator is obtained by querying the function values over M points in the neighborhood of $\hat{\mathbf{x}}_t$. These points include $\mathbf{x}_{m,t}:=\hat{\mathbf{x}}_t+\delta\mathbf{u}_{m,t}$, $1\leq m\leq M-1$, and the learning iterate $\mathbf{x}_{m,t}:=\hat{\mathbf{x}}_t$, where $\mathbf{u}_{m,t}$ is independently drawn from $\mathbb{S}$. Specifically, the gradient becomes (cf. (6.11))

$$\hat{\nabla}_{\mathbf{x}}^M\mathcal{L}_t(\hat{\mathbf{x}}_t,\boldsymbol{\lambda}_t):=\frac{d}{\delta(M-1)}\sum_{m=1}^{M-1}\big(f_t(\hat{\mathbf{x}}_t+\delta\mathbf{u}_{m,t})-f_t(\hat{\mathbf{x}}_t)\big)\mathbf{u}_{m,t}+\nabla^\top\mathbf{g}_t(\hat{\mathbf{x}}_t)\boldsymbol{\lambda}_t \tag{6.17}$$

where we define the M-point stochastic gradient as $\hat{\nabla}^M f_t(\hat{\mathbf{x}}_t):=\frac{d}{\delta(M-1)}\sum_{m=1}^{M-1}\big(f_t(\hat{\mathbf{x}}_t+\delta\mathbf{u}_{m,t})-f_t(\hat{\mathbf{x}}_t)\big)\mathbf{u}_{m,t}$. At the price of extra computations, simulations will validate that the BanSaP with multipoint feedback enjoys improved performance. The family of the BanSaP approaches with one- or multiple-point feedback is summarized in Algorithm 8.

Remark 9 (Sampling schemes). The BanSaP solvers here adopt uniform sampling for gradient estimation, meaning $\mathbf{u}$ is drawn uniformly from the unit sphere. However, other sampling rules can be incorporated without affecting the order of regret bounds derived later. For example, one can sample $\mathbf{u}$ from the canonical basis of a d-dimensional space uniformly at random [3], or, sample $\mathbf{u}$ from a normal distribution [120]. The effectiveness of these schemes will be tested using simulations.

6.4 Performance analysis

In this section, we will introduce pertinent metrics to evaluate BanSaP algorithms in the online bandit learning with long-term constraints, and rigorously analyze the performance of the proposed algorithms.

6.4.1 Optimality and feasibility metrics

With regard to performance of BCO schemes, static regret is a common metric, under time-invariant and strictly satisfied constraints, which measures the difference between the aggregate loss and that of the best fixed solution in hindsight [3, 52]. Extending the definition of static regret to accommodate M-point function evaluations and time-varying constraints, let us first consider

$$\mathrm{Reg}_T^{\mathrm{s}} := \frac{1}{M}\sum_{t=1}^{T}\sum_{m=1}^{M}\mathbb{E}\left[f_t(\mathbf{x}_{m,t})\right] - \sum_{t=1}^{T} f_t(\mathbf{x}^*) \tag{6.18}$$

where the actual loss per slot is averaged over the losses of M actions (queried points), $\mathbb{E}$ is taken over the sequence of the random actions $\mathbf{x}_{m,t}$ with randomness induced by $\{\mathbf{u}_{m,t}\}$ perturbations, and the best static solution is $\mathbf{x}^* \in \arg\min_{\mathbf{x}\in\mathcal{X}} \sum_{t=1}^{T} f_t(\mathbf{x});$ s. to $\mathbf{g}_t(\mathbf{x}) \leq \mathbf{0}, \ \forall t$. A BCO algorithm yielding a sub-linear regret implies that the algorithm is "on average" no-regret [106]; or, in other words, asymptotically not worse than the best fixed solution $\mathbf{x}^*$. Though widely used, the *static regret* relies on a rather coarse benchmark, which is not as useful in dynamic IoT settings. Specifically, the gap between the loss of the best static and that of the best dynamic benchmark is as large as $\mathcal{O}(T)$ [15].

In response to the quest for improved benchmarks in this dynamic setup with constraints, two metrics are considered here: *dynamic regret* and *dynamic fit*. The notion of dynamic regret has been recently adopted in [71, 62] to assess performance of online algorithms under time-invariant constraints. For our BCO setting of (6.1), we adopt

$$\mathrm{Reg}_T^{\mathrm{d}} := \frac{1}{M}\sum_{t=1}^{T}\sum_{m=1}^{M}\mathbb{E}\left[f_t(\mathbf{x}_{m,t})\right] - \sum_{t=1}^{T} f_t(\mathbf{x}_t^*) \tag{6.19}$$

where $\mathbb{E}$ is again taken over the sequence of random actions, and the benchmark is now formed via a sequence of best dynamic solutions $\{\mathbf{x}_t^*\}$ for the instantaneous cost minimization problem subject to the instantaneous constraint, namely

$$\mathbf{x}_t^* \in \arg\min_{\mathbf{x}\in\mathcal{X}} \; f_t(\mathbf{x}) \quad \text{s. to} \quad \mathbf{g}_t(\mathbf{x}) \leq \mathbf{0}. \tag{6.20}$$

Comparing (6.19) with (6.18), if $\mathbf{x}_t^* = \mathbf{x}^*, \forall t$, then the static regret is equivalent to the dynamic regret. In general, the dynamic regret is larger than the static regret, i.e., $\mathrm{Reg}_T^{\mathrm{s}} \leq \mathrm{Reg}_T^{\mathrm{d}}$, since $\sum_{t=1}^T f_t(\mathbf{x}^*)$ is always no smaller than $\sum_{t=1}^T f_t(\mathbf{x}_t^*)$ according to the definitions of $\mathbf{x}^*$ and $\mathbf{x}_t^*$. Hence, a sub-linear dynamic regret implies a sub-linear static regret, but not vice versa.

Regarding feasibility of decisions generated by a BCO algorithm, the notion of *dynamic fit* will be used to measure the accumulated violation of constraints [106], that is

$$\mathrm{Fit}_T^{\mathrm{d}} := \left\| \left[\frac{1}{M} \sum_{t=1}^{T} \sum_{m=1}^{M} \mathbf{g}_t(\mathbf{x}_{m,t}) \right]^+ \right\|. \tag{6.21}$$

Note that the dynamic fit is zero if the accumulated violation $\frac{1}{M}\sum_{t=1}^T \sum_{m=1}^M \mathbf{g}_t(\mathbf{x}_{m,t})$ is entry-wise less than zero. Hence, enforcing $\frac{1}{M}\sum_{t=1}^T \sum_{m=1}^M \mathbf{g}_t(\mathbf{x}_{m,t}) \leq \mathbf{0}$ is different from restricting $\mathbf{x}_t$ to meet $\frac{1}{M}\sum_{m=1}^M \mathbf{g}_t(\mathbf{x}_{m,t}) \leq \mathbf{0}$ in every slot. While the latter implies the former, the long-term constraint implicitly assumes that the instantaneous constraint violations can be compensated by the later strictly feasible decisions.

Under this broader BCO setup, an ideal online algorithm is the one that achieves both sub-linear dynamic regret and sub-linear dynamic fit. A sub-linear dynamic regret implies "no-regret" relative to the clairvoyant dynamic solution on the long-term average; i.e., $\lim_{T\to\infty} \mathrm{Reg}_T^{\mathrm{d}}/T = 0$; and a sub-linear dynamic fit indicates that the online strategy is also feasible on average; i.e., $\lim_{T\to\infty} \mathrm{Fit}_T^{\mathrm{d}}/T = 0$. Unfortunately, the sub-linear dynamic regret is not achievable under *arbitrary* underlying dynamics, even when the time-varying constraint in (6.1) is absent [15]. Therefore, we aim at designing an online strategy that generates a sequence $\{\mathbf{x}_{m,t}\}$ ensuring sub-linear dynamic regret and fit, under the suitable *regularity conditions* on the underlying dynamics.

6.4.2 Main results

Before formally analyzing the dynamic regret and fit for BanSaP, we assume that the following conditions are satisfied.

(as1) *For every t, the functions $f_t(\mathbf{x})$ and $\mathbf{g}_t(\mathbf{x})$ are convex.*

(as2) *Function $f_t(\mathbf{x})$ is bounded over the set $\mathcal{X}$, meaning $|f_t(\mathbf{x})| \leq F$, $\forall \mathbf{x} \in \mathcal{X}$; while $f_t(\mathbf{x})$ and $g_t^n(\mathbf{x})$ have bounded gradients; that is, $\|\nabla f_t(\mathbf{x})\| \leq G$, and $\max_n \|\nabla g_t^n(\mathbf{x})\| \leq G$.*

(as3) *For a small constant γ, there exists a constant $\eta > 0$, and an interior point $\tilde{\mathbf{x}} \in (1-\gamma)\mathcal{X}$ such that $\mathbf{g}_t(\tilde{\mathbf{x}}) \leq -\eta\mathbf{1}$, $\forall t$.*
(as4) *With $\mathbb{B} := \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\| \leq 1\}$ denoting the unit ball, there exist constants $0 < r \leq R$ such that $r\mathbb{B} \subseteq \mathcal{X} \subseteq R\mathbb{B}$.*

Assumptions (as1)-(as2) are typical in OCO with both full- and partial-information feedback [65, 106, 52]; (as3) is Slater's condition modified for our BCO setting, which guarantees the existence of a bounded Lagrange multiplier [13] in the constrained optimization; and, (as4) requires the action set to be bounded within a ball that contains the origin. When (as4) appears to be restrictive, it is tantamount to assuming $\mathcal{X}$ is compact and has a nonempty interior, because one can always apply an affine transformation (a.k.a. reshaping) on $\mathcal{X}$ to satisfy (as4); see [52, Section 3.2].

Under these assumptions, we are on track to first provide upper bounds for the dynamic regret, and the dynamic fit of the BanSaP solver with one-point feedback.

Theorem 11 (one-point feedback)**.** *Suppose that (as1)-(as4) are satisfied, and consider the parameters α, μ, δ, γ defined in* (6.11)-(6.13)*, and constants F, G, r, R defined in (as2)-(as4). If the dual variable is initialized by $\boldsymbol{\lambda}_1 = \mathbf{0}$, then the BanSaP with one-point feedback in* (6.7)-(6.8) *has dynamic regret bounded by*

$$\mathrm{Reg}_T^{\mathrm{d}} \leq \frac{R}{\alpha}V(\mathbf{x}^*_{1:T}) + \frac{R^2}{2\alpha} + \frac{d^2G^2R^2\alpha T}{\delta^2} + 2G\delta T + \gamma GRT\left(1+\|\bar{\boldsymbol{\lambda}}\|\right) + 2\mu G^2R^2T \tag{6.22}$$

where $\|\bar{\boldsymbol{\lambda}}\| := \max_t \|\boldsymbol{\lambda}_t\|$, and the accumulated variation of the per-slot minimizers $\mathbf{x}^*_t$ in (6.20) *is given by*

$$V(\mathbf{x}^*_{1:T}) := \sum_{t=1}^{T} \|\mathbf{x}^*_t - \mathbf{x}^*_{t-1}\|. \tag{6.23}$$

In addition, the dynamic fit defined in (6.21) *is bounded by*

$$\mathrm{Fit}_T^{\mathrm{d}} \leq \frac{\|\bar{\boldsymbol{\lambda}}\|}{\mu} + \frac{G^2\sqrt{N}T}{2\beta} + \delta G\sqrt{N}T + \beta\sqrt{N}T\left(\frac{\alpha^2d^2F^2}{\delta^2} + \alpha^2G^2\|\bar{\boldsymbol{\lambda}}\|^2\right) \tag{6.24}$$

where $\beta > 0$ is a pre-selected constant. Furthermore, if we choose the stepsizes as $\alpha = \mu = \mathcal{O}(T^{-\frac{3}{4}})$, and the parameters $\delta = \mathcal{O}(T^{-\frac{1}{4}})$, $\beta = T^{\frac{1}{4}}$ and $\gamma = \delta/r$, then the online decisions generated by BanSaP are feasible, i.e., $\mathbf{x}_{1,t} \in \mathcal{X}$; and also yield the following dynamic regret and fit

$$\boxed{\mathrm{Reg}_T^{\mathrm{d}} = \mathcal{O}\Big(V(\mathbf{x}^*_{1:T})T^{\frac{3}{4}}\Big) \text{ and } \mathrm{Fit}_T^{\mathrm{d}} = \mathcal{O}\big(T^{\frac{3}{4}}\big).} \tag{6.25}$$

For BanSaP with one-point feedback, Theorem 11 asserts that its dynamic regret and fit are upper-bounded by some constants depending on the those parameters, the time horizon, and the accumulated variation of per-slot minimizers. Interestingly, the crucial constant δ controlling the perturbation of random actions appears in both the denominator and numerator of (6.22) and (6.24), which correspond to the variance and bias of the gradient estimator. Therefore, simply setting a small δ will not only reduce the bias, but it will also boost the variance - a clear manifestation of the that is known as bias-variance tradeoff in BCO [143]. Optimally choosing parameters implies that the dynamic fit is sub-linearly growing, and the dynamic regret is sub-linear given that the variation of the per-slot minimizer is slow enough; i.e., $V(\mathbf{x}^*_{1:T}) = \mathbf{o}(T^{\frac{1}{4}})$.

Regarding BanSaP with two-point feedback, we can prove the following result that parallels Theorem 11.

Theorem 12 (two-point feedback). *Consider the assumptions and the definitions of constants in Theorem 11. If the dual variable is initialized by $\boldsymbol{\lambda}_1 = \mathbf{0}$, then BanSaP with two-point feedback has dynamic regret bounded by*

$$\mathrm{Reg}^{\mathrm{d}}_T \leq \frac{R}{\alpha}V(\mathbf{x}^*_{1:T}) + \frac{R^2}{2\alpha} + 2\mu G^2R^2T + \alpha d^2G^2T + \gamma GRT(1+\|\bar{\boldsymbol{\lambda}}\|) + 2\delta GT \tag{6.26}$$

and has dynamic fit in (6.21) *bounded by*

$$\mathrm{Fit}^{\mathrm{d}}_T \leq \frac{\|\bar{\boldsymbol{\lambda}}\|}{\mu} + \frac{G^2\sqrt{N}T}{2\beta} + \delta G\sqrt{N}T + \beta\sqrt{N}T\big(\alpha^2d^2G^2 + \alpha^2G^2\|\bar{\boldsymbol{\lambda}}\|^2\big). \tag{6.27}$$

In this case, if we choose the stepsizes as $\alpha = \mu = \mathcal{O}(T^{-\frac{1}{2}})$, and set the parameters as $\beta = T^{\frac{1}{2}}$, $\delta = \mathcal{O}(T^{-1})$, and $\gamma = \delta/r$, then the online decisions generated by BanSaP are feasible, and its dynamic regret and fit are bounded by

$$\boxed{\mathrm{Reg}^{\mathrm{d}}_T = \mathcal{O}\Big(V(\mathbf{x}^*_{1:T})T^{\frac{1}{2}}\Big) \text{ and } \mathrm{Fit}^{\mathrm{d}}_T = \mathcal{O}\big(T^{\frac{1}{2}}\big)} \tag{6.28}$$

where $V(\mathbf{x}^*_{1:T})$ is the accumulated variation of the per-slot minimizers $\mathbf{x}^*_t$ in (6.23).

Comparing with the bounds in (6.22) and (6.24), the perturbation constant δ only appears in the numerator of (6.26) and (6.27) because our gradient estimator here replies on two points. In this case, the additional function evaluation allows BanSaP to choose an arbitrarily small δ to minimize the bias of stochastic gradient, without increasing its variance. This observation is aligned with those in BCO without long-term constraints [3, 143]. Furthermore, Theorem 12 establishes that the dynamic regret and fit are sub-linear if $V(\mathbf{x}^*_{1:T}) = \mathbf{o}(T^{\frac{1}{2}})$, which markedly improves those in Theorem 11 under one-point feedback.

For the case of BanSaP with $M > 2$ points, slightly improved bounds can be proved without changing the order of regret and fit, but they are omitted here for brevity. In addition, the bounds in Theorems 11 and 12 can be achieved without any knowledge of $V(\mathbf{x}^*_{1:T})$. When the order of $V(\mathbf{x}^*_{1:T})$ is known, or, can be estimated a-priori, tighter regret and fit bounds can be obtained by adjusting stepsizes accordingly. Formally, we can arrive at the following corollary.

Corollary 2. *Under the conditions of Theorems 11 and 12, suppose that there exists a constant $\rho \in [0, 1)$ such that the variation satisfies $V(\mathbf{x}^*_{1:T}) = \mathbf{o}(T^{\rho})$. If the stepsizes of BanSaP with one-point feedback are chosen as $\alpha = \mu = \mathcal{O}\big(T^{\frac{3}{4}(\rho-1)}\big)$, and the parameters are $\delta = \mathcal{O}(T^{\frac{1}{4}(\rho-1)})$, $\beta = T^{\frac{3}{4}(1-\rho)}$, and $\gamma = \delta/r$, then the dynamic regret and fit in* (6.25) *become*

$$\mathrm{Reg}^{\mathrm{d}}_T = \mathcal{O}\Big(T^{\frac{1}{4}(\rho+3)}\Big) \ \text{ and } \ \mathrm{Fit}^{\mathrm{d}}_T = \mathcal{O}\left(T^{\frac{1}{4}(\rho+3)}\right). \tag{6.29}$$

Likewise, if the stepsizes of BanSaP with two-point feedback are chosen such that $\alpha = \mu = \mathcal{O}\big(T^{\frac{1}{2}(\rho-1)}\big)$, and the parameters are $\delta = \mathcal{O}(T^{\frac{1}{2}(\rho-1)})$, $\beta = T^{\frac{1}{2}(1-\rho)}$, and $\gamma = \delta/r$, then the dynamic regret and fit in (6.25) *become*

$$\mathrm{Reg}^{\mathrm{d}}_T = \mathcal{O}\Big(T^{\frac{1}{2}(\rho+1)}\Big) \ \text{ and } \ \mathrm{Fit}^{\mathrm{d}}_T = \mathcal{O}\left(T^{\frac{1}{2}(\rho+1)}\right). \tag{6.30}$$

Apparently, Corollary 2 implies that sub-linear dynamic regret and fit are both possible, provided that the accumulated variation of the minimizers is growing sub-linearly ($\rho < 1$), and it is available to the learner in advance. It provides valuable insights for choosing optimal stepsizes in dynamic environments. Specifically, adjusting stepsizes to match the variability of the environment is the key to achieving the optimal dynamic regret and fit. Intuitively, when the variation is fast (large ρ), slowly decaying stepsizes (thus larger stepsizes) can better track the potential changes; and vice versa.

Remark 10 (Optimal regret). As a special case of Theorems 11 and 12, by confining $\mathbf{x}^*_1 = \cdots = \mathbf{x}^*_T$ so that $V(\mathbf{x}^*_{1:T}) = 0$, the dynamic regret bounds (6.25) and (6.28) reduce to the static ones, which correspond to $\mathcal{O}(T^{\frac{3}{4}})$ in the one-point feedback case, and to $\mathcal{O}(\sqrt{T})$ in the two-point case. This pair of bounds markedly improves the *regret versus fit tradeoff* in [106], and matches the order of regret in [52], and [3, 50], which are the best possible ones that can be achieved by *efficient* algorithms even in the BCO setup without the long-term constraints. Considering the full-information setting in [27] as a special case, the regret and fit of BanSaP outperform those in

Algorithm 9 BanSaP for edge computing

```
1: Initialize: primal iterates $\{\hat{y}_1^{nk}\}$ and $\{\hat{z}_1^n\}$, dual iterate $\boldsymbol{\lambda}_1$, parameters $\delta$ and $\gamma$, and proper stepsizes $\alpha$ and $\mu$.
2: for $t = 1, 2 \ldots$ do
3:     for $m = 1, \ldots, M$ do
4:         Fog nodes perform perturbed offloading decisions to cloud $\{z_{m,t}^n\}$, to neighbor edges $\{y_{m,t}^{nk}\}$, and locally process $\{y_{m,t}^{nn}\}$ based on $\hat{\mathbf{x}}_t$.
5:     end for
6:     Fog nodes observe the (possibly multiple) losses to update (6.31) with stochastic gradients obtained via (6.32).
7:     Fog nodes observe the actual demands from devices to update the dual variables (6.33).
8: end for
```

[27].

Remark 11 (Dynamic regret). Theorems 11, 12 and Corollary 2 extend the dynamic regret analysis in [62, 71, 25] to the regime of *bandit* online learning with long-term *time-varying* constraints. Interestingly though, in the BCO setting of our interest, sub-linear dynamic regret and fit are possible to achieve when the per-slot minimizer *does not vary on average*, that is, $V(\mathbf{x}_{1:T}^*)$ is sub-linearly growing with T.

6.5 Numerical tests

In this section, we demonstrate how the edge computing task can benefit from BanSaP.

6.5.1 BanSaP for edge computing

Recall that the computation offloading problem (6.4) is in the form of (6.1). Therefore, the BanSaP solver of Section 6.3 can be customized to solve (6.4) in an *online* fashion, with provable performance and feasibility guarantees.

Specifically, with $\mathbf{g}_t(\mathbf{x}_t)$ as in (6.2) and $f_t(\mathbf{x}_t)$ as in (6.3), the primal update (6.7) boils down to a simple closed-form gradient update amenable to decentralized implementation; the cloud offloading amount at node n is

$$\hat{z}_{t+1}^n = \left[\hat{z}_t^n - \alpha\left(\hat{\nabla} c_t^n(\hat{z}_t^n) - \lambda_t^n\right)\right]_0^{\bar{z}^n} \tag{6.31a}$$

and the offloading amount from node n to node k is given by

$$\hat{y}_{t+1}^{nk} = \left[\hat{y}_t^{nk} - \alpha\left(\hat{\nabla} c_t^{nk}(\hat{y}_t^{nk}) - \lambda_t^n + \lambda_t^k\right)\right]_0^{\bar{y}^{nk}} \tag{6.31b}$$

while the local processing decision at node n is generated by

$$\hat{y}_{t+1}^{nn} = \left[\hat{y}_t^{nn} - \alpha\big(\hat{\nabla} h_t^n(\hat{y}_t^{nn}) - \lambda_t^n\big)\right]_0^{\bar{y}^{nn}} \tag{6.31c}$$

where α is chosen according to Theorems 11 and 12. Using two-point feedback $(M = 2)$ as an example, the gradients involved in (6.31) can be estimated as

$$\hat{\nabla}^2 c_t^n(\hat{z}_t^n) := \frac{d}{2\delta}\Big(f_t\big(\underbrace{\hat{\mathbf{x}}_t + \delta \mathbf{u}_t}_{\mathbf{x}_{1,t}}\big) - f_t\big(\underbrace{\hat{\mathbf{x}}_t - \delta \mathbf{u}_t}_{\mathbf{x}_{2,t}}\big)\Big) u_t(\hat{z}^n) \tag{6.32a}$$

and with respect to the offloading variable, as

$$\hat{\nabla}^2 c_t^{nk}(\hat{y}_t^{nk}) := \frac{d}{2\delta}\big(f_t(\hat{\mathbf{x}}_t + \delta \mathbf{u}_t) - f_t(\hat{\mathbf{x}}_t - \delta \mathbf{u}_t)\big) u_t(\hat{y}^{nk}) \tag{6.32b}$$

and with respect to the local processing variable, as

$$\hat{\nabla}^2 h_t^n(\hat{y}_t^{nn}) := \frac{d}{2\delta}\big(f_t(\hat{\mathbf{x}}_t + \delta \mathbf{u}_t) - f_t(\hat{\mathbf{x}}_t - \delta \mathbf{u}_t)\big) u_t(\hat{y}^{nn}) \tag{6.32c}$$

where $u_t(\hat{z}^n)$, $u_t(\hat{y}^{nk})$, and $u_t(\hat{y}^{nn})$ represent the corresponding entries of the random vector $\mathbf{u}_t \in \mathbb{R}^{|\mathcal{E}|}$ at slot t.

The dual update (6.8) at each node n reduces to

$$\lambda_{t+1}^n = \left[\lambda_t^n + \mu\left(b_t^n + \sum_{k \in \mathcal{N}_n^{\text{in}}} \hat{y}_{t+1}^{kn} - \sum_{k \in \mathcal{N}_n^{\text{out}}} \hat{y}_{t+1}^{nk} - \hat{z}_{t+1}^n - \hat{y}_{t+1}^{nn}\right)\right]^+ \tag{6.33}$$

where μ is chosen according to Theorems 11 and 12. Intuitively, to guarantee completion of the service requests, the dual variable increases (increasing penalty) when there is instantaneous service residual, and decreases when over-serving incurs in the mobile-edge computing systems. Following its generic form in Algorithm 8, BanSaP for online edge computing tasks, is summarized in Algorithm 9.

6.5.2 Numerical experiments

Consider the fog computing task in a smart home setting with $N = 10$ fog nodes (e.g., smart home gateways), and a remote cloud center [184]. Each fog node has an outgoing link to the cloud, and two outgoing links to two nearby fog nodes for local collaborative computing. For a communication link offloading loads from fog node n to k, we consider the low-power wireless

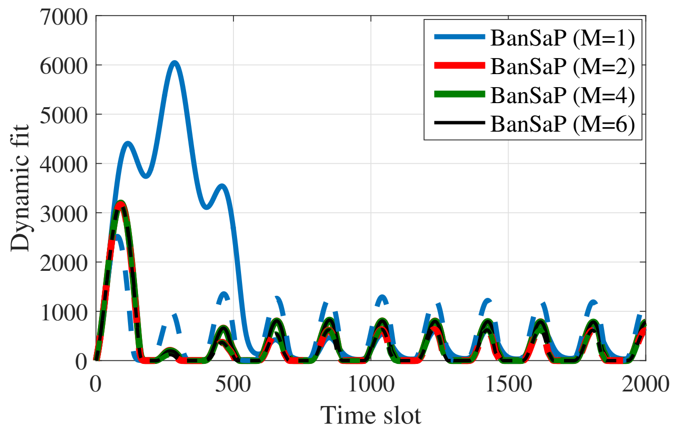

Figure 6.3: Effect of sampling schemes and number of feedback on dynamic fit. Solid lines: BanSaP with uniformly sampling from a unit sphere (uniform sampling). Dashed lines: BanSaP with randomly sampling from standard basis (coordinate sampling).

connection such as Bluetooth or ZigBee [134], whose the offloading limit is $\bar{y}^{nk} = 10$; and for all the fog-cloud offloading communication links, we consider the high-bandwidth WiFi connection with the offloading limits $\{\bar{z}^n\}$ being 100. For each fog node n, the local computation limit is assumed to be $\bar{y}^{nn} = 50$. Regarding communication and computation latency, a linear function is employed to model the communication latency between fog nodes under a constant transmission rate; a quadratic function is adopted for the local computation latency to account for the potential queueing time due to other active services in the fog node; and an exponential function is assumed for the fog-cloud communication latency to characterize the unpredictable network latency due to involved wide-area routing through the Internet backbone [181]. Therefore, the online cost (a.k.a. aggregate service latency) in (6.3) is specified by

$$f_t(\mathbf{x}_t) := \sum_{n\in\mathcal{N}} \left(e^{p_t^n z_t^n} + \textstyle\sum_{k\in\mathcal{N}_n^{\text{out}}} l^{nk} y_t^{nk} + l^{nn}(y_t^{nn})^2 \right) \tag{6.34}$$

where $p_t^n = 0.015\sin(\pi t/96) + 0.05$, $n \in \mathcal{N} \backslash \{4,5\}$, $p_t^n = 0.045\sin(\pi t/96) + 0.15$, $n \in \{4,5\}$, and the local coefficients are set to $l^{nk} = 8/\bar{y}^{nk}$ and $l^{nn} = 8/\bar{y}^{nn}$. Regarding the data arrival rate b_t^n, it is generated according to $b_t^n = q^n \sin(\pi t/96) + \nu_t^n$, with q^n and ν_t^n uniformly distributed over $[40, 50]$ and $[45, 55]$ for $n \in \mathcal{N} \backslash \{1,2,3\} \bigcup \{4,5\}$, and $q^n \in [32, 40], \nu_t^n \in [36, 44]$, $n \in \{1,2,3\}$, and $q^n \in [20, 25], \nu_t^n \in [22.5, 27.5]$, $n \in \{4,5\}$. Notice that the periods of p_t^n and b_t^n correspond to a 24-hour interval, following the periodic patterns of human activities in a home sensor network [102]; while the scales of p_t^n and b_t^n vary between nodes, mimicking heterogeneity of IoT sensors such as motion sensors and thermostat sensors [2]. It is also worth mentioning that our BanSaP

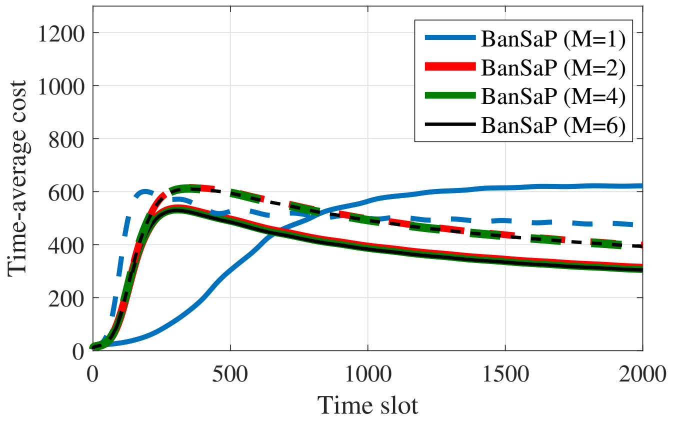

Figure 6.4: Effect of sampling schemes and number of feedback on average cost. Solid lines: BanSaP with sampling from a unit sphere (uniform sampling). Dashed lines: BanSaP with randomly sampling from standard basis (coordinate sampling).

algorithm and its performance analysis are model-free, which means they can incorporate more complex models so long as the loss function is convex and its point-wise value can be evaluated.

Finally, BanSaP is benchmarked by: i) the *full-information* modified online saddle-point method (MOSP) in [25] that takes gradient-based update for primal-dual variables; ii) the *heuristic* cloud-only approach that offloads all data requests to the remote cloud; iii) the *heuristic* fog-only approach that processes all data requests locally without collaboration; and, iv) the *partial-information* perturbed online primal-dual method in [106]. For both cloud-only and fog-only approaches, unoffloaded and unprocessed requests are buffered at the fog nodes for later processing; thus, these amounts are measured by their fit. Regarding the perturbed primal-dual method in [106], it comes with two-point bandit feedback, and the perturbation constant is chosen as 0.06 to satisfy the technical conditions therein. As different stepsizes of BanSaP and MOSP lead to different behaviors, we manually optimized stepsizes in each test so that they have similar fit, and focus on their cost comparison. When the parameters of BanSaP need to be slightly adjusted in each test, they are set to $\gamma = 0.05$, and $\delta = 4$ for with $M = 1$, and $\delta = 0.05$ for $M \geq 2$. All tests were averaged over 500 Monte Carlo realizations.

Effect of complexity and sampling schemes. In a simplified setting with $N = 5$ nodes, the fit and average cost are compared among the BanSaP variants with M-point feedback under different sampling schemes in Figs. 6.3 and 6.4. Clearly, for both sampling schemes, the cost and fit of BanSaP solvers decrease as the amount of bandit feedback increases. However, such performance gain varnishes when feedback increases; e.g., $M \geq 4$. Regarding the sampling schemes, Fig. 6.3 demonstrates that when all the BanSaP variants have low dynamic fit, the uniform sampling-based

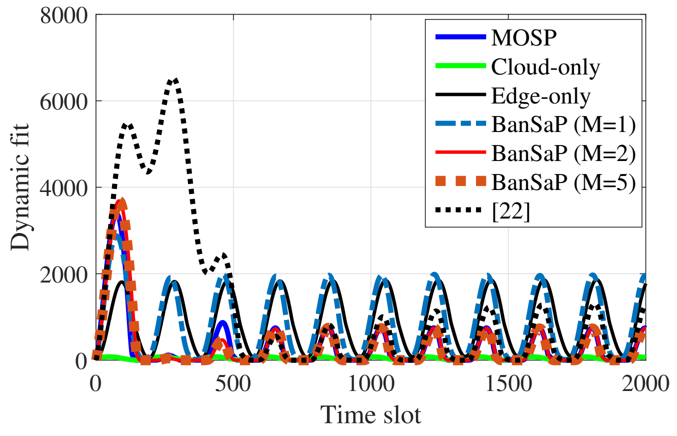

Figure 6.5: Comparison based on dynamic fit.

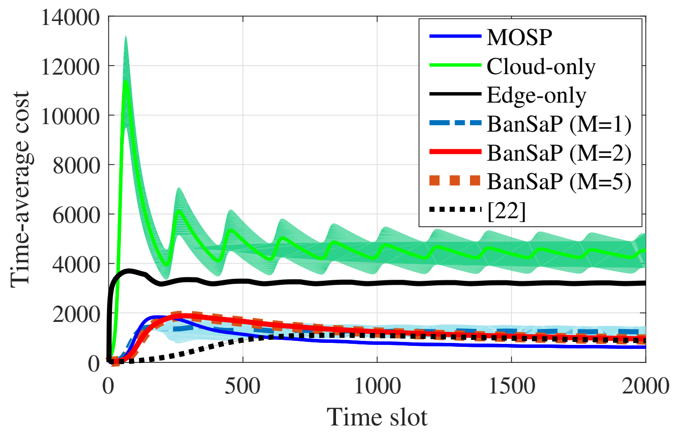

Figure 6.6: Comparison of average costs. The shaded region represents the cost distribution of each scheme within one standard deviation of the mean.

BanSaP with one-point feedback has large initial fit; and Fig. 6.4 confirms that for $M = 1$, the coordinate sampling-based BanSaP outperforms that with uniform sampling; and, for $M \geq 2$, the BanSaP solvers with uniform sampling incur lower cost. Therefore, to optimize empirical performance in the subsequent tests, coordinate sampling is adopted by BanSaP with $M = 1$, while uniform sampling is used in BanSaP with $M \geq 2$.

Optimality and feasibility. With optimized sampling schemes for BanSaP solvers, the dynamic fit and average cost are then compared among three BanSaP variants, MOSP, the perturbed primal-dual method in [106], and two heuristic schemes in Figs. 6.5 and 6.6. Without queueing at the fog side, the cloud-only scheme has much lower dynamic fit since all user demands are offloaded to the remote cloud. However, it incurs a much higher average cost (service latency) as the network

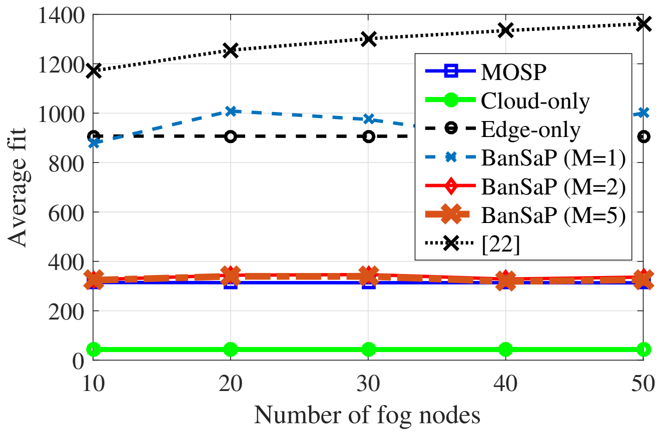

Figure 6.7: Impact of network size on dynamic fit per fog node.

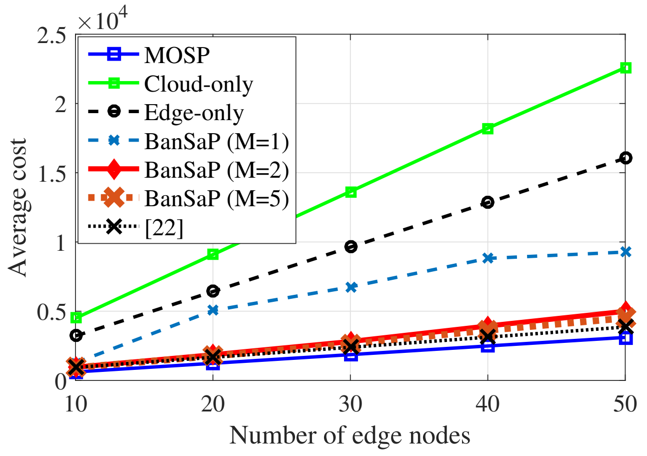

Figure 6.8: Impact of network size on average network cost.

latency between fog and cloud becomes high due to the large offloading amount. By increasing the amount of feedback, the BanSaP solver tends to have a lower fit and a lower average cost, both of which are comparable to those of MOSP when $M \geq 2$. On the other hand, the BanSaP with only one-point bandit feedback still has a similar fit relative to the fog-only scheme, but enjoys much lower cost. Interestingly enough, when the variance (cf. the shaded area in Fig. 6.6) of the one-point BanSaP's cost is high, it markedly varnishes when multiple function values become available, which corroborates our claims in Theorems 11-12. For the perturbed method in [106], while its average cost is similar or slightly better than that of BanSaP, its dynamic fit is much higher than all BanSaP variants, which is aligned with its $\mathcal{O}(T^{\frac{3}{4}})$ fit (cf. $\mathcal{O}(T^{\frac{1}{2}})$ in our case).

Effect of network size. The third test evaluates the performance of all schemes under different number of fog nodes (i.e., network size). For each algorithm, the fit averaged over all fog nodes and time is presented in Fig. 6.7, and the cost averaged over the time is shown in Fig. 6.8. Clearly, the one-point BanSaP has lower average fit than the fog-only approach in most scenarios, and also incurs less average cost in all tested settings. Similar to those in Figs. 6.5 and 6.6, the average fit and cost of BanSaP with multiple function evaluations is still comparable to that of the full-information MOSP as the network size grows. On the other hand, the method in [106] enjoys slightly lower average cost as the network size grows, but its dynamic fit is again much higher than all BanSaP variants. An interesting observation here is that as the number of fog nodes increases, the performance gain of the BanSaP solver with a large M becomes more evident; see e.g., Fig. 6.8. This implies that for a larger network, BanSaP benefits from more bandit information to learn and track the network dynamics.

6.6 Proofs of lemmas and theorems

6.6.1 Bias and variance of gradient estimators

Before proving the main theory, we first establish several lemmas related to the quality of gradient estimators, which later will serve to certify our primal and dual updates. The following lemma establishes unbiasedness of one- and two-point estimations [52, 3].

Lemma 20 (Unbiasedness of gradient estimators)**.** *With* $\mathbf{u}$ *drawn uniformly from the surface of the unit ball* $\mathbb{S} := \{\mathbf{u} : \|\mathbf{u}\| = 1\} \subseteq \mathbb{R}^d$*, we have for given a constant* $\delta > 0$ *that*

$$\mathbb{E}_{\mathbf{u}}\left[\frac{d}{\delta} f_t(\mathbf{x} + \delta\mathbf{u})\mathbf{u}\right] = \nabla \check{f}_t(\mathbf{x}) \tag{6.35}$$

where $\nabla \check{f}_t(\mathbf{x})$ *is the gradient of the smoothed function* $\check{f}_t(\mathbf{x}) := \mathbb{E}_{\mathbf{v}}[f_t(\mathbf{x} + \delta\mathbf{v})]$ *with* $\mathbf{v}$ *drawn from a unit ball* $\mathbb{B}$*, and* d *is the dimension of* $\mathbf{x}$*. Likewise, for the two-point case, we have that*

$$\mathbb{E}_{\mathbf{u}}\left[\frac{d}{2\delta}\Big(f_t(\mathbf{x} + \delta\mathbf{u}) - f_t(\mathbf{x} - \delta\mathbf{u})\Big)\mathbf{u}\right] = \nabla \check{f}_t(\mathbf{x}). \tag{6.36}$$

Lemma 20 provides valuable insights for performing gradient-based algorithms in bandit setting. Namely, $\hat{\nabla}^1 f_t(\hat{\mathbf{x}}_t)$ and $\hat{\nabla}^2 f_t(\hat{\mathbf{x}}_t)$ are the unbiased gradient estimators of the *smoothed* function $\check{f}_t(\mathbf{x})$, which is an approximation of $f_t(\mathbf{x})$.

The following lemma establishes the norm (or variance) of one- and two-point gradient estimations [52, 3].

Lemma 21 (Norm of gradient estimators)**.** *For the gradient $\hat{\nabla}^1 f_t(\hat{\mathbf{x}}_t)$ in* (6.12)*, we have that*

$$\|\hat{\nabla}^1 f_t(\hat{\mathbf{x}}_t)\| \leq \frac{d}{\delta} F \tag{6.37}$$

where F is an upper-bound of the function. For the gradient estimator $\hat{\nabla}^2 f_t(\hat{\mathbf{x}}_t)$ in (6.15)*, we have that*

$$\|\hat{\nabla}^2 f_t(\hat{\mathbf{x}}_t)\| \leq dG \tag{6.38}$$

where G is the Lipschitz constant of the loss function.

Having bounded the norm of stochastic gradients, the next lemma is useful to ensure feasibility of actual online actions.

Lemma 22 ([52, Observation 2])**.** *Consider a constant $r > 0$ so that $r\mathbb{B} \subseteq \mathcal{X}$, where $\mathbb{B} := \{\mathbf{v} : \|\mathbf{v}\| \leq 1\} \subseteq \mathbb{R}^d$ is the unit ball. If we choose $\gamma = \delta/r$, and the iterate satisfies $\hat{\mathbf{x}}_t \in (1-\gamma)\mathcal{X}$, then $\hat{\mathbf{x}}_t + \delta \mathbf{u}_t \in \mathcal{X}$, where $\mathbf{u}_t$ is drawn uniformly from the unit sphere $\mathbb{S} := \{\mathbf{u} : \|\mathbf{u}\| = 1\} \subseteq \mathbb{R}^d$.*

Specifically, Lemma 22 asserts that if the perturbation constant γ is sufficiently small, the actually perturbed IoT actions $\{\mathbf{x}_t\}$ generated by BanSaP are feasible w.r.t. $\mathcal{X}$.

6.6.2 Relating regret and fit to primal and dual drift

The next lemma is crucial to establish the dynamic fit [117].

Lemma 23 (Approximate constraint violation)**.** *Considering the BanSaP recursion, we have the following bound for the constraint violation*

$$\sum_{t=1}^{T} \mathbf{g}_t(\hat{\mathbf{x}}_t) \leq \frac{\boldsymbol{\lambda}_{T+1}}{\mu} + \frac{G^2 T \mathbf{1}}{2\beta} + \frac{\beta}{2} \sum_{t=1}^{T} \|\hat{\mathbf{x}}_{t+1} - \hat{\mathbf{x}}_t\|^2 \mathbf{1} \tag{6.39}$$

where $\mu > 0$ is the stepsize of the dual iteration (6.8)*, and $\beta > 0$ is a pre-defined constant.*

Complex as it may appear, the implication of Lemma 23 is that the dynamic fit in (6.21) will depend on the norm of dual variables as well as the variation of consecutive primal iterates.

Analogous to Lemma 23, the next lemma serves as an intermediate step to establish the dynamic regret.

Lemma 24 (Approximate per-slot regret)**.** *Consider the BanSaP algorithm with a generic gradient $\hat{\nabla} f_t(\hat{\mathbf{x}}_t)$, which is estimated from one- or multi-point feedback. With $\Delta(\boldsymbol{\lambda}_t) := \frac{1}{2}(\|\boldsymbol{\lambda}_{t+1}\|^2 - \|\boldsymbol{\lambda}_t\|^2)$, it holds for $\forall \mathbf{x} \in (1-\gamma)\mathcal{X}$ that*

$$\mathbb{E}[\check{f}_t(\hat{\mathbf{x}}_t)] - \check{f}_t(\mathbf{x}) \leq -\frac{1}{\mu}\mathbb{E}[\Delta(\boldsymbol{\lambda}_t)] + \mathbb{E}[\boldsymbol{\lambda}_t^\top \mathbf{g}_t(\mathbf{x})] + 2\mu G^2 R^2 \\ + \frac{1}{2\alpha}\mathbb{E}[\|\mathbf{x} - \hat{\mathbf{x}}_t\|^2] - \frac{1}{2\alpha}\mathbb{E}[\|\mathbf{x} - \hat{\mathbf{x}}_{t+1}\|^2] + \alpha\|\hat{\nabla} f_t(\hat{\mathbf{x}}_t)\|^2 \tag{6.40}$$

where the constants G, R and F are as in (as2) and (as3).

Proof: Taking the norm square in (6.13), we have

$$\|\boldsymbol{\lambda}_{t+1}\|^2 \leq \|\boldsymbol{\lambda}_t\|^2 + 2\mu\boldsymbol{\lambda}_t^\top(\mathbf{g}_t(\hat{\mathbf{x}}_t) + \nabla^\top \mathbf{g}_t(\hat{\mathbf{x}}_t)(\hat{\mathbf{x}}_{t+1} - \hat{\mathbf{x}}_t)) \\ + 2\mu^2\|\mathbf{g}_t(\hat{\mathbf{x}}_t)\|^2 + 2\mu^2\|\nabla^\top \mathbf{g}_t(\hat{\mathbf{x}}_t)(\hat{\mathbf{x}}_{t+1} - \hat{\mathbf{x}}_t)\|^2. \tag{6.41}$$

With $\Delta(\boldsymbol{\lambda}_t) := \frac{1}{2}(\|\boldsymbol{\lambda}_{t+1}\|^2 - \|\boldsymbol{\lambda}_t\|^2)$, (6.41) implies that

$$\frac{1}{\mu}\Delta(\boldsymbol{\lambda}_t) \leq \boldsymbol{\lambda}_t^\top(\mathbf{g}_t(\hat{\mathbf{x}}_t) + \nabla^\top \mathbf{g}_t(\hat{\mathbf{x}}_t)(\hat{\mathbf{x}}_{t+1} - \hat{\mathbf{x}}_t)) + 2\mu G^2 R^2. \tag{6.42}$$

On the other hand, recall that the primal iterate $\hat{\mathbf{x}}_{t+1}$ is the optimal solution to the following optimization problem

$$\hat{\mathbf{x}}_{t+1} = \arg\min_{\mathbf{x}\in(1-\gamma)\mathcal{X}} \hat{\nabla}_{\mathbf{x}}^\top \mathcal{L}_t(\hat{\mathbf{x}}_t, \boldsymbol{\lambda}_t)(\mathbf{x} - \hat{\mathbf{x}}_t) + \frac{1}{2\alpha}\|\mathbf{x} - \hat{\mathbf{x}}_t\|^2. \tag{6.43}$$

Recalling the definition of $\hat{\nabla}_{\mathbf{x}} \mathcal{L}_t(\hat{\mathbf{x}}_t, \boldsymbol{\lambda}_t)$, we thus have that

$$\hat{\mathbf{x}}_{t+1} = \arg\min_{\mathbf{x}\in(1-\gamma)\mathcal{X}} \hat{\nabla}^\top f_t(\hat{\mathbf{x}}_t)(\mathbf{x} - \hat{\mathbf{x}}_t) + \boldsymbol{\lambda}_t^\top(\mathbf{g}_t(\hat{\mathbf{x}}_t) + \nabla^\top \mathbf{g}_t(\hat{\mathbf{x}}_t)(\mathbf{x} - \hat{\mathbf{x}}_t)) + \frac{1}{2\alpha}\|\mathbf{x} - \hat{\mathbf{x}}_t\|^2 \tag{6.44}$$

where we add $\boldsymbol{\lambda}_t^\top \mathbf{g}_t(\mathbf{x}_t)$ to the RHS of (6.43). the minimizer of (6.43) does not change, since the added term is constant.

To connect (6.42) with (6.44), adding $\hat{\nabla}^\top f_t(\hat{\mathbf{x}}_t)(\hat{\mathbf{x}}_{t+1} - \hat{\mathbf{x}}_t) + \frac{1}{2\alpha}\|\hat{\mathbf{x}}_{t+1} - \hat{\mathbf{x}}_t\|^2$ to the RHS of (6.42), we have that

$$\frac{1}{\mu}\Delta(\boldsymbol{\lambda}_t) + \hat{\nabla}^\top f_t(\hat{\mathbf{x}}_t)(\hat{\mathbf{x}}_{t+1} - \hat{\mathbf{x}}_t) + \frac{1}{2\alpha}\|\hat{\mathbf{x}}_{t+1} - \hat{\mathbf{x}}_t\|^2$$

$$\le \boldsymbol{\lambda}_t^\top \left(\mathbf{g}_t(\hat{\mathbf{x}}_t)+\nabla^\top \mathbf{g}_t(\hat{\mathbf{x}}_t)(\hat{\mathbf{x}}_{t+1}-\hat{\mathbf{x}}_t)\right)+\frac{1}{2\alpha}\left\|\hat{\mathbf{x}}_{t+1}-\hat{\mathbf{x}}_t\right\|^2$$
$$+\hat{\nabla}^\top f_t(\hat{\mathbf{x}}_t)\left(\hat{\mathbf{x}}_{t+1}-\hat{\mathbf{x}}_t\right)+2\mu G^2 R^2. \tag{6.45}$$

Note that $\hat{\mathbf{x}}_{t+1}$ is the minimizer of (6.44), where the objective on the RHS of (6.44) is strongly-convex, thus we have that

$$\frac{1}{\mu}\Delta(\boldsymbol{\lambda}_t)+\hat{\nabla}^\top f_t(\hat{\mathbf{x}}_t)(\hat{\mathbf{x}}_{t+1}-\hat{\mathbf{x}}_t)+\frac{1}{2\alpha}\left\|\hat{\mathbf{x}}_{t+1}-\hat{\mathbf{x}}_t\right\|^2$$
$$\le \boldsymbol{\lambda}_t^\top \left(\mathbf{g}_t(\hat{\mathbf{x}}_t)+\nabla^\top \mathbf{g}_t(\hat{\mathbf{x}}_t)(\mathbf{x}-\hat{\mathbf{x}}_t)\right)+\frac{1}{2\alpha}\left\|\mathbf{x}-\hat{\mathbf{x}}_t\right\|^2+2\mu G^2 R^2$$
$$+\hat{\nabla}^\top f_t(\hat{\mathbf{x}}_t)(\mathbf{x}-\hat{\mathbf{x}}_t)-\frac{1}{2\alpha}\left\|\mathbf{x}-\hat{\mathbf{x}}_{t+1}\right\|^2$$
$$\le \boldsymbol{\lambda}_t^\top \mathbf{g}_t(\mathbf{x})+\hat{\nabla}^\top f_t(\hat{\mathbf{x}}_t)(\mathbf{x}-\hat{\mathbf{x}}_t)+2\mu G^2 R^2$$
$$+\frac{1}{2\alpha}\left\|\mathbf{x}-\hat{\mathbf{x}}_t\right\|^2-\frac{1}{2\alpha}\left\|\mathbf{x}-\hat{\mathbf{x}}_{t+1}\right\|^2,\ \forall \mathbf{x}\in(1-\gamma)\mathcal{X} \tag{6.46}$$

where we used the non-negativity that $\boldsymbol{\lambda}_t \ge \mathbf{0}$, and the convexity such that $\mathbf{g}_t(\hat{\mathbf{x}}_t)+\nabla^\top \mathbf{g}_t(\hat{\mathbf{x}}_t)(\mathbf{x}-\hat{\mathbf{x}}_t)\le \mathbf{g}_t(\mathbf{x})$.

Using the Cauchy-Schwarz inequality, we have that

$$-\hat{\nabla}^\top f_t(\hat{\mathbf{x}}_t)(\hat{\mathbf{x}}_{t+1}-\hat{\mathbf{x}}_t)\le \alpha\|\hat{\nabla} f_t(\hat{\mathbf{x}}_t)\|^2+\frac{\|\hat{\mathbf{x}}_{t+1}-\hat{\mathbf{x}}_t\|^2}{4\alpha}. \tag{6.47}$$

Plugging (6.47) into (6.46), for $\forall \mathbf{x}\in(1-\gamma)\mathcal{X}$, we have that

$$\frac{1}{\mu}\Delta(\boldsymbol{\lambda}_t)+\frac{1}{4\alpha}\left\|\hat{\mathbf{x}}_{t+1}-\hat{\mathbf{x}}_t\right\|^2\le \boldsymbol{\lambda}_t^\top \mathbf{g}_t(\mathbf{x})+\hat{\nabla}^\top f_t(\hat{\mathbf{x}}_t)\left(\mathbf{x}-\hat{\mathbf{x}}_t\right)$$
$$+2\mu G^2 R^2+\frac{1}{2\alpha}\left\|\mathbf{x}-\hat{\mathbf{x}}_t\right\|^2-\frac{1}{2\alpha}\left\|\mathbf{x}-\hat{\mathbf{x}}_{t+1}\right\|^2+\alpha\|\hat{\nabla} f_t(\hat{\mathbf{x}}_t)\|^2. \tag{6.48}$$

Taking expectation over $\mathbf{u}_t$ on both side of (6.48) conditioning on $\hat{\mathbf{x}}_t$, it follows that

$$\frac{1}{\mu}\mathbb{E}[\Delta(\boldsymbol{\lambda}_t)]+\frac{1}{4\alpha}\mathbb{E}[\|\hat{\mathbf{x}}_{t+1}-\hat{\mathbf{x}}_t\|^2] \tag{6.49}$$
$$\le \boldsymbol{\lambda}_t^\top \mathbf{g}_t(\mathbf{x})+\mathbb{E}\left[\hat{\nabla}^\top f_t(\hat{\mathbf{x}}_t)\left(\mathbf{x}-\hat{\mathbf{x}}_t\right)\right]+2\mu G^2 R^2$$
$$+\frac{1}{2\alpha}\left\|\mathbf{x}-\hat{\mathbf{x}}_t\right\|^2-\frac{1}{2\alpha}\mathbb{E}[\|\mathbf{x}-\hat{\mathbf{x}}_{t+1}\|^2]+\alpha\|\hat{\nabla} f_t(\hat{\mathbf{x}}_t)\|^2$$
$$\overset{(a)}{=} \boldsymbol{\lambda}_t^\top \mathbf{g}_t(\mathbf{x})+\nabla^\top \check{f}_t(\hat{\mathbf{x}}_t)\left(\mathbf{x}-\hat{\mathbf{x}}_t\right)+2\mu G^2 R^2$$
$$+\frac{1}{2\alpha}\left\|\mathbf{x}-\hat{\mathbf{x}}_t\right\|^2-\frac{1}{2\alpha}\mathbb{E}[\|\mathbf{x}-\hat{\mathbf{x}}_{t+1}\|^2]+\alpha\|\hat{\nabla} f_t(\hat{\mathbf{x}}_t)\|^2$$

where (a) holds since the randomness $\mathbf{u}_t$ in $\hat{\nabla} f_t(\hat{\mathbf{x}}_t)$ is independent of $\hat{\mathbf{x}}_t$, and $\hat{\nabla} f_t(\hat{\mathbf{x}}_t)$ is an unbiased estimator of $\nabla \check{f}_t(\hat{\mathbf{x}}_t)$.

The convexity of $f_t(\mathbf{x})$ implies that $\check{f}_t(\mathbf{x})$ is also convex, and thus $\nabla^\top \check{f}_t(\hat{\mathbf{x}}_t)\,(\mathbf{x}-\hat{\mathbf{x}}_t) \leq \check{f}_t(\mathbf{x}) - \check{f}_t(\hat{\mathbf{x}}_t)$. Plugging into (6.49) and taking expectation over all possible $\hat{\mathbf{x}}_t$, it follows that

$$\begin{aligned}&\frac{1}{\mu}\mathbb{E}[\Delta(\boldsymbol{\lambda}_t)]+\frac{1}{4\alpha}\mathbb{E}[\|\hat{\mathbf{x}}_{t+1}-\hat{\mathbf{x}}_t\|^2] \\ &\leq \check{f}_t(\mathbf{x}) - \mathbb{E}[\check{f}_t(\hat{\mathbf{x}}_t)] + \mathbb{E}[\boldsymbol{\lambda}_t^\top \mathbf{g}_t(\mathbf{x})] + 2\mu G^2 R^2 \\ &+\frac{1}{2\alpha}\mathbb{E}[\|\mathbf{x}-\hat{\mathbf{x}}_t\|^2]-\frac{1}{2\alpha}\mathbb{E}[\|\mathbf{x}-\hat{\mathbf{x}}_{t+1}\|^2]+\alpha\mathbb{E}[\|\hat{\nabla} f_t(\hat{\mathbf{x}}_t)\|^2]\end{aligned} \tag{6.50}$$

which completes the proof by dropping $\mathbb{E}[\|\hat{\mathbf{x}}_{t+1}-\hat{\mathbf{x}}_t\|^2]$.

If one plugs $\mathbf{x} = \mathbf{x}_t^*$ into (6.40), Lemma 24 asserts that the approximate per-slot regret depends on the norm of the primal-dual gradients as well as the drift of the primal-dual updates.

6.6.3 Proof of Theorem 11

With $\gamma = \delta/r$, the feasibility of actions $\{\mathbf{x}_{1,t}\}$ readily follows from Lemma 22, i.e., $\mathbf{x}_{1,t} \in \mathcal{X}$, $\forall t$. As the dynamic fit eventually depends on the norm of the dual variable (cf. Lemma 23), the following result is needed.

Lemma 25 (Bound on dual variables)**.** *For the BanSaP recursion, if $\alpha = \mu = \mathcal{O}(T^{-\frac{3}{4}})$ and $\delta = \mathcal{O}(T^{-\frac{1}{4}})$, the dual iterates are bounded by $\|\boldsymbol{\lambda}_t\| \leq C = \mathcal{O}(1)$, with constant given by*

$$C := \max\left\{2GR,\ \Big(\frac{1}{\eta}+1\Big)GR+\frac{2G^2R^2\mu}{\eta}+\frac{d^2F^2\alpha}{\eta\delta^2}+\frac{\mu R^2}{2\alpha\eta}\right\} \tag{6.51}$$

where the constants G, R, and η are as in (as2)-(as4).

Proof: Plugging the bounded norm of the one-point gradient estimator (6.37) into (6.40), it holds that

$$\begin{aligned}\frac{1}{\mu}\mathbb{E}[\Delta(\boldsymbol{\lambda}_t)] \leq GR + \mathbb{E}[\boldsymbol{\lambda}_t^\top \mathbf{g}_t(\mathbf{x})] + 2\mu G^2R^2 + \frac{d^2F^2\alpha}{\delta^2} \\ +\frac{1}{2\alpha}\mathbb{E}[\|\mathbf{x}-\hat{\mathbf{x}}_t\|^2]-\frac{1}{2\alpha}\mathbb{E}[\|\mathbf{x}-\hat{\mathbf{x}}_{t+1}\|^2]\end{aligned} \tag{6.52}$$

where we used the Lipschitz condition on (6.40); i.e.,

$$\mathbb{E}[\check{f}_t(\mathbf{x}) - \check{f}_t(\hat{\mathbf{x}}_t)] \leq GR. \tag{6.53}$$

Selecting the interior point $\mathbf{x} = \tilde{\mathbf{x}} \in (1-\gamma)\mathcal{X}$ so that $\mathbf{g}_t(\tilde{\mathbf{x}}) \le -\eta\mathbf{1}$, it follows from (6.52) that

$$\frac{1}{\mu}\mathbb{E}[\Delta(\boldsymbol{\lambda}_t)] \le GR - \eta\mathbb{E}[\boldsymbol{\lambda}_t^\top \mathbf{1}] + 2\mu G^2R^2 + \frac{d^2F^2\alpha}{\delta^2} + \frac{1}{2\alpha}\mathbb{E}[\|\tilde{\mathbf{x}} - \hat{\mathbf{x}}_t\|^2] - \frac{1}{2\alpha}\mathbb{E}[\|\tilde{\mathbf{x}} - \hat{\mathbf{x}}_{t+1}\|^2]. \tag{6.54}$$

Using $-\eta\boldsymbol{\lambda}_t^\top\mathbf{1} = -\eta\|\boldsymbol{\lambda}_t\|_1 \le -\eta\|\boldsymbol{\lambda}_t\|$, we arrive at

$$\frac{1}{\mu}\mathbb{E}[\Delta(\boldsymbol{\lambda}_t)] \le GR - \eta\mathbb{E}[\|\boldsymbol{\lambda}_t\|] + 2\mu G^2R^2 + \frac{d^2F^2\alpha}{\delta^2} + \frac{1}{2\alpha}\mathbb{E}[\|\tilde{\mathbf{x}} - \hat{\mathbf{x}}_t\|^2] - \frac{1}{2\alpha}\mathbb{E}[\|\tilde{\mathbf{x}} - \hat{\mathbf{x}}_{t+1}\|^2]. \tag{6.55}$$

Now we are ready to show that the norm of the dual variable is uniformly bounded by a constant C that is independent of time; that is, $\|\boldsymbol{\lambda}_t\| \le C,\ \forall t$.

For $1 \le t \le \frac{1}{\mu}$, it follows readily that

$$\begin{aligned}\|\boldsymbol{\lambda}_t\| &\le \|\boldsymbol{\lambda}_{t-1}\| + \mu\|\mathbf{g}_t(\hat{\mathbf{x}}_t) + \nabla^\top\mathbf{g}_t(\hat{\mathbf{x}}_t)(\hat{\mathbf{x}}_{t+1} - \hat{\mathbf{x}}_t)\| \\ &\le \|\boldsymbol{\lambda}_{t-1}\| + 2\mu GR \le \|\boldsymbol{\lambda}_1\| + 2\mu tGR \le C\end{aligned} \tag{6.56}$$

where the last inequality follows from $\boldsymbol{\lambda}_1 = \mathbf{0}$, $t \le 1/\mu$, and the definition of C in (6.51).

For $\frac{1}{\mu} \le t \le T$, we will prove the claim by contradiction. Assume T_0 is the first slot for which $\|\boldsymbol{\lambda}_{T_0}\| > C$. Therefore, we have $\|\boldsymbol{\lambda}_{T_0}\| > C \ge \|\boldsymbol{\lambda}_{T_0-\frac{1}{\mu}}\|$, which after recalling (6.55) and the definition of $\Delta(\boldsymbol{\lambda}_t)$, yields

$$\frac{1}{\mu}\sum_{t=T_0-\frac{1}{\mu}}^{T_0-1}\mathbb{E}[\Delta(\boldsymbol{\lambda}_t)] = \frac{1}{2\mu}\left(\mathbb{E}\left[\|\boldsymbol{\lambda}_{T_0}\|^2 - \|\boldsymbol{\lambda}_{T_0-\frac{1}{\mu}}\|^2\right]\right) > 0. \tag{6.57}$$

On the other hand however, summing up (6.55), we obtain

$$\frac{1}{\mu}\sum_{t=T_0-\frac{1}{\mu}}^{T_0-1}\mathbb{E}[\Delta(\boldsymbol{\lambda}_t)] \le \frac{GR}{\mu} - \eta\sum_{t=T_0-\frac{1}{\mu}}^{T_0-1}\mathbb{E}[\|\boldsymbol{\lambda}_t\|] + 2G^2R^2 + \frac{d^2F^2\alpha}{\mu\delta^2} + \frac{1}{2\alpha}\mathbb{E}[\|\tilde{\mathbf{x}} - \hat{\mathbf{x}}_{T_0-\frac{1}{\mu}}\|^2] - \frac{1}{2\alpha}\mathbb{E}[\|\tilde{\mathbf{x}} - \hat{\mathbf{x}}_{T_0}\|^2]$$

$$\overset{(a)}{\leq}\frac{GR}{\mu}-\eta\sum_{t=T_0-\frac{1}{\mu}}^{T_0-1}\mathbb{E}[\|\boldsymbol{\lambda}_t\|]+2G^2R^2+\frac{d^2F^2\alpha}{\mu\delta^2}+\frac{R^2}{2\alpha} \tag{6.58}$$

where (a) uses again the bound $\|\tilde{\mathbf{x}}-\hat{\mathbf{x}}_{T_0-\frac{1}{\mu}}\|\leq R$.

Note that since $\|\boldsymbol{\lambda}_{T_0}\|>C$ and $\|\boldsymbol{\lambda}_{T_0}\|-\|\boldsymbol{\lambda}_{T_0-1}\|\leq 2\mu GR$, we have that

$$\|\boldsymbol{\lambda}_{T_0-\tau}\|>C-2\tau\mu GR. \tag{6.59}$$

Combining (6.58) with (6.59), we deduce

$$\frac{1}{\mu}\sum_{t=T_0-\frac{1}{\mu}}^{T_0-1}\mathbb{E}[\Delta(\boldsymbol{\lambda}_t)]\leq\frac{GR}{\mu}-\frac{C\eta}{\mu}+\frac{\eta GR}{\mu}+2G^2R^2+\frac{d^2F^2\alpha}{\mu\delta^2}+\frac{R^2}{2\alpha}. \tag{6.60}$$

Together with (6.57), recursion (6.60) implies that

$$C<\frac{GR}{\eta}+GR+\frac{2G^2R^2\mu}{\eta}+\frac{d^2F^2\alpha}{\eta\delta^2}+\frac{\mu R^2}{2\alpha\eta} \tag{6.61}$$

which contradicts the definition of C in (6.51). Hence, there is no T_0 satisfying $\|\boldsymbol{\lambda}_t\|\leq C$, which implies that $\|\boldsymbol{\lambda}_t\|\leq C,\ \forall t$.

By choosing the stepsizes $\alpha=\mu=\mathcal{O}(T^{-\frac{3}{4}})$, and the parameter $\delta=\mathcal{O}(T^{-\frac{1}{4}})$, it follows that

$$C=\mathcal{O}\left(\frac{GR}{\eta}+GR+\frac{2G^2R^2}{\eta T^{\frac{3}{4}}}+\frac{d^2F^2}{\eta T^{\frac{1}{4}}}+\frac{R^2}{2\eta}\right)=\mathcal{O}(1) \tag{6.62}$$

which completes the proof of the lemma.

Dynamic regret in Theorem 1: The dynamic regret follows from Lemma 25. Recall that $\mathbf{x}_t^*$ is the minimizer of the time-varying problem (6.20), and $(1-\gamma)\mathbf{x}_t^*\in(1-\gamma)\mathcal{X}$. Hence, plugging $(1-\gamma)\mathbf{x}_t^*$ into (6.40), we have

$$\begin{aligned}&\frac{1}{\mu}\mathbb{E}[\Delta(\boldsymbol{\lambda}_t)]\leq\check{f}_t((1-\gamma)\mathbf{x}_t^*)-\mathbb{E}[\check{f}_t(\hat{\mathbf{x}}_t)]\\&+\frac{1}{2\alpha}\mathbb{E}[\|(1-\gamma)\mathbf{x}_t^*-\hat{\mathbf{x}}_t\|^2]-\frac{1}{2\alpha}\mathbb{E}[\|(1-\gamma)\mathbf{x}_t^*-\hat{\mathbf{x}}_{t+1}\|^2]\\&+\mathbb{E}[\boldsymbol{\lambda}_t^\top\mathbf{g}_t((1-\gamma)\mathbf{x}_t^*)]+\frac{\alpha}{\delta^2}d^2F^2+2\mu G^2R^2.\end{aligned} \tag{6.63}$$

From the Lipschitz condition, we can bound the inner product in (6.63) by

$$\mathbb{E}[\boldsymbol{\lambda}_t^\top \mathbf{g}_t((1-\gamma)\mathbf{x}_t^*)] \leq \mathbb{E}[\boldsymbol{\lambda}_t^\top(\mathbf{g}_t(\mathbf{x}_t^*)+\gamma GR\cdot\mathbf{1})] \overset{(a)}{\leq} \gamma GR\mathbb{E}[\|\boldsymbol{\lambda}_t\|] \overset{(b)}{\leq} \gamma GR\|\bar{\boldsymbol{\lambda}}\| \tag{6.64}$$

where (a) follows from $\boldsymbol{\lambda}_t^\top \mathbf{g}_t(\mathbf{x}_t^*) \leq 0$ since $\mathbf{g}_t(\mathbf{x}_t^*) \leq \mathbf{0}$, and $\boldsymbol{\lambda}_t \geq \mathbf{0}$; and (b) uses the upper bound of $\|\bar{\boldsymbol{\lambda}}\| := \max_t \|\boldsymbol{\lambda}_t\|$. The two distance terms in (6.63) can be bounded by

$$\begin{aligned}
&\|(1-\gamma)\mathbf{x}_t^* - \hat{\mathbf{x}}_t\|^2 - \|(1-\gamma)\mathbf{x}_t^* - \hat{\mathbf{x}}_{t+1}\|^2 \\
&= \|(1-\gamma)\mathbf{x}_t^* - \hat{\mathbf{x}}_t\|^2 - \|(1-\gamma)\mathbf{x}_{t-1}^* - \hat{\mathbf{x}}_t\|^2 \\
&\qquad + \|(1-\gamma)\mathbf{x}_{t-1}^* - \hat{\mathbf{x}}_t\|^2 - \|(1-\gamma)\mathbf{x}_t^* - \hat{\mathbf{x}}_{t+1}\|^2 \\
&= (1-\gamma)\|\mathbf{x}_t^* - \mathbf{x}_{t-1}^*\|\|(1-\gamma)(\mathbf{x}_t^* + \mathbf{x}_{t-1}^*) - 2\hat{\mathbf{x}}_t\| \\
&\qquad + \|(1-\gamma)\mathbf{x}_{t-1}^* - \hat{\mathbf{x}}_t\|^2 - \|(1-\gamma)\mathbf{x}_t^* - \hat{\mathbf{x}}_{t+1}\|^2.
\end{aligned} \tag{6.65}$$

Using the triangle inequality, it follows that

$$\|(1-\gamma)(\mathbf{x}_t^* + \mathbf{x}_{t-1}^*) - 2\hat{\mathbf{x}}_t\| \leq \|(1-\gamma)\mathbf{x}_t^* - \hat{\mathbf{x}}_t\| + \|(1-\gamma)\mathbf{x}_{t-1}^* - \hat{\mathbf{x}}_t\| \leq 2R \tag{6.66}$$

which together with (6.65), implies that

$$\begin{aligned}
&\|(1-\gamma)\mathbf{x}_t^* - \hat{\mathbf{x}}_t\|^2 - \|(1-\gamma)\mathbf{x}_t^* - \hat{\mathbf{x}}_{t+1}\|^2 \\
&\leq 2(1-\gamma)R\|\mathbf{x}_t^* - \mathbf{x}_{t-1}^*\| + \|(1-\gamma)\mathbf{x}_{t-1}^* - \hat{\mathbf{x}}_t\|^2 - \|(1-\gamma)\mathbf{x}_t^* - \hat{\mathbf{x}}_{t+1}\|^2.
\end{aligned} \tag{6.67}$$

Plugging (6.64) and (6.67) into (6.63), and summing up over $t = 1, \ldots, T$, we find

$$\begin{aligned}
&\frac{1}{2\mu}\left(\mathbb{E}[\|\boldsymbol{\lambda}_{T+1}\|^2 - \|\boldsymbol{\lambda}_1\|^2]\right) + \sum_{t=1}^{T}\left(\mathbb{E}[\check{f}_t(\hat{\mathbf{x}}_t)] - \check{f}_t((1-\gamma)\mathbf{x}_t^*)\right) \\
&\leq \gamma GR\|\bar{\boldsymbol{\lambda}}\|T + \sum_{t=1}^{T}\frac{(1-\gamma)R}{\alpha}\|\mathbf{x}_t^* - \mathbf{x}_{t-1}^*\| + 2\mu G^2R^2T + \frac{\alpha d^2F^2T}{\delta^2} \\
&\quad + \frac{1}{2\alpha}\Big(\mathbb{E}\big[\|(1-\gamma)\mathbf{x}_0^* - \hat{\mathbf{x}}_1\|^2\big] - \mathbb{E}\big[\|(1-\gamma)\mathbf{x}_T^* - \hat{\mathbf{x}}_{T+1}\|^2\big]\Big) \\
&\overset{(c)}{\leq} \gamma GR\|\bar{\boldsymbol{\lambda}}\|T + \frac{R}{\alpha}V(\mathbf{x}_{1:T}^*) + 2\mu G^2R^2T + \frac{R^2}{2\alpha} + \frac{\alpha d^2F^2T}{\delta^2}
\end{aligned} \tag{6.68}$$

where (c) uses $\|(1-\gamma)\mathbf{x}_0^* - \hat{\mathbf{x}}_1\| \leq \|\mathbf{x}_0^* - \hat{\mathbf{x}}_1\| \leq R$, and $V(\mathbf{x}_{1:T}^*) := \sum_{t=1}^{T}\|\mathbf{x}_t^* - \mathbf{x}_{t-1}^*\|$.

Since $\mathbb{E}[\|\boldsymbol{\lambda}_{T+1}\|^2] \geq 0$, initializing the dual variable with $\boldsymbol{\lambda}_1 = \mathbf{0}$, and rearranging (6.68), we have that

$$\sum_{t=1}^{T}\left(\mathbb{E}[\check{f}_t(\hat{\mathbf{x}}_t)]-\check{f}_t((1-\gamma)\mathbf{x}_t^*)\right)$$
$$\leq\gamma GR\|\bar{\boldsymbol{\lambda}}\|T+\frac{R}{\alpha}V(\mathbf{x}_{1:T}^*)+2\mu G^2R^2T+\frac{R^2}{2\alpha}+\frac{\alpha d^2F^2T}{\delta^2}. \tag{6.69}$$

The iterates $\{\hat{\mathbf{x}}_t\}$ in this bound are not the actual decisions taken by the learner. To obtain the regret bound, our next step is to decompose the regret as

$$\sum_{t=1}^{T}\Big(\mathbb{E}[f_t(\mathbf{x}_{1,t})]-f_t(\mathbf{x}_t^*)\Big)=\sum_{t=1}^{T}\Big(\underbrace{\mathbb{E}[f_t(\mathbf{x}_{1,t})]-\mathbb{E}[\check{f}_t(\mathbf{x}_{1,t})]}_{U_1}$$
$$+\underbrace{\mathbb{E}[\check{f}_t(\mathbf{x}_{1,t})]-\mathbb{E}[\check{f}_t(\hat{\mathbf{x}}_t)]}_{U_2}+\underbrace{\mathbb{E}[\check{f}_t(\hat{\mathbf{x}}_t)]-\check{f}_t((1-\gamma)\mathbf{x}_t^*))}_{U_3}$$
$$+\underbrace{\check{f}_t((1-\gamma)\mathbf{x}_t^*))-\check{f}_t(\mathbf{x}_t^*)}_{U_4}+\underbrace{\check{f}_t(\mathbf{x}_t^*)-f_t(\mathbf{x}_t^*)}_{U_5}\Big). \tag{6.70}$$

We next bound each under-braced, starting with

$$U_1=\mathbb{E}\left[f_t(\mathbf{x}_{1,t})-\mathbb{E}_{\mathbf{v}}[f_t(\mathbf{x}_{1,t}+\delta\mathbf{v}_t)]\right]$$
$$\overset{(d)}{\leq}\mathbb{E}\left[f_t(\mathbf{x}_{1,t})-f_t(\mathbb{E}_{\mathbf{v}}[\mathbf{x}_{1,t}+\delta\mathbf{v}_t])\right]\overset{(e)}{=}0 \tag{6.71}$$

where (d) uses Jensen's inequality, and (e) follows from $\mathbb{E}_{\mathbf{v}}[\delta\mathbf{v}_t]=\mathbf{0}$ since $\mathbf{v}_t$ is drawn from $\mathbb{B}:=\{\mathbf{v}:\|\mathbf{v}\|\leq 1\}$.

Regarding the second term, it follows that

$$U_2=\mathbb{E}[\check{f}_t(\hat{\mathbf{x}}_t+\delta\mathbf{u}_t)-\check{f}_t(\hat{\mathbf{x}}_t)]\overset{(f)}{\leq}\mathbb{E}[G\|\delta\mathbf{u}_t\|]=\delta G \tag{6.72}$$

where (f) uses the Lipschitz condition of $\check{f}_t(\mathbf{x})$. The third term U_3 has been already bounded as in (6.69).

Using the Lipschitz condition of $\check{f}_t(\mathbf{x})$, we can further bound

$$U_4=\check{f}_t((1-\gamma)\mathbf{x}_t^*))-\check{f}_t(\mathbf{x}_t^*)\leq\gamma GR \tag{6.73}$$

and likewise for the last term for which

$$U_5 = \mathbb{E}_{\mathbf{v}}[f_t(\mathbf{x}_t^* + \delta\mathbf{v}_t)] - f_t(\mathbf{x}_t^*) \leq \mathbb{E}_{\mathbf{v}}\left[G\|\delta\mathbf{v}_t\|\right] \leq \delta G. \tag{6.74}$$

Plugging (6.69) and (6.71)-(6.74) into (6.70), we arrive that

$$\sum_{t=1}^{T}\Big(\mathbb{E}[f_t(\mathbf{x}_{1,t})] - f_t(\mathbf{x}_t^*)\Big) \leq \frac{R}{\alpha}V(\mathbf{x}_{1:T}^*) + \frac{R^2}{2\alpha} + \frac{d^2G^2R^2\alpha T}{\delta^2}$$
$$+\gamma GRT(1+\|\bar{\boldsymbol{\lambda}}\|) + 2\mu G^2R^2T + 2G\delta T. \tag{6.75}$$

Upon choosing $\alpha = \mu = \mathcal{O}(T^{-\frac{3}{4}})$, and $\delta = \mathcal{O}(T^{-\frac{1}{4}})$ along with $\gamma = \delta/r$, it follows that (cf. Lemma 25)

$$\mathrm{Reg}_T^{\mathrm{d}} = \mathcal{O}\Big(RV(\mathbf{x}_{1:T}^*)T^{\frac{3}{4}} + GRCT^{\frac{3}{4}} + 2G^2R^2T^{\frac{1}{4}} + d^2G^2R^2T^{\frac{3}{4}}\Big)$$

from which the proof is complete.

Dynamic fit in Theorem 1: To bound the dynamic fit, recall that the constraint violations in Lemma 23 depend on the norm of the dual variable and the difference of two consecutive primal iterates. While the dual variable has been bounded in Lemma 25, the distance between $\hat{\mathbf{x}}_t$ and $\hat{\mathbf{x}}_{t+1}$ is bounded by

$$\|\hat{\mathbf{x}}_{t+1} - \hat{\mathbf{x}}_t\| \overset{(a)}{\leq} \left\|\alpha\hat{\nabla}_{\mathbf{x}}^1\mathcal{L}_t(\hat{\mathbf{x}}_t, \boldsymbol{\lambda}_t)\right\|$$
$$\overset{(b)}{\leq} \frac{d}{\delta}|f_t(\hat{\mathbf{x}}_t + \delta\mathbf{u}_t)| + \|\nabla\mathbf{g}_t(\hat{\mathbf{x}}_t)\|\|\boldsymbol{\lambda}_t\| \overset{(c)}{\leq} \frac{\alpha dF}{\delta} + \alpha G\|\boldsymbol{\lambda}_t\| \tag{6.76}$$

where (a) uses the non-expansive property of the projection operator, (b) relies on (6.12) and the Cauchy-Schwarz's inequality; and (c) uses the bounds in (as2).

On the other hand, using the Lipschitz continuity of $\mathbf{g}_t(\mathbf{x})$ and (6.39), it follows that

$$\sum_{t=1}^{T}\mathbf{g}_t(\mathbf{x}_{1,t}) \leq \sum_{t=1}^{T}\mathbf{g}_t(\hat{\mathbf{x}}_t) + \delta GT\mathbf{1} \tag{6.77}$$
$$\leq \frac{\boldsymbol{\lambda}_{T+1}}{\mu} + \frac{G^2T\mathbf{1}}{2\beta} + \frac{\beta}{2}\sum_{t=1}^{T}\|\hat{\mathbf{x}}_{t+1} - \hat{\mathbf{x}}_t\|^2\mathbf{1} + \delta GT\mathbf{1}$$
$$\overset{(d)}{\leq} \frac{\boldsymbol{\lambda}_{T+1}}{\mu} + \frac{G^2T\mathbf{1}}{2\beta} + \beta T\Big(\frac{\alpha^2d^2F^2}{\delta^2} + \alpha^2G^2\|\bar{\boldsymbol{\lambda}}\|^2\Big)\mathbf{1} + \delta GT\mathbf{1}$$

where (d) uses (6.76), and the fact that $(a+b)^2 \leq 2(a^2+b^2)$. Taking $[\cdot]^+$ and $\|\cdot\|$ on both sides of (6.77), we have (cf. (6.21))

$$\begin{aligned}\mathrm{Fit}_T^{\mathrm{d}} \leq & \frac{\|\bar{\boldsymbol{\lambda}}\|}{\mu} + \frac{G^2\sqrt{N}T}{2\beta} + \delta G\sqrt{N}T \\ & + \beta\sqrt{N}T\left(\alpha^2 d^2 F^2/\delta^2 + \alpha^2 G^2\|\bar{\boldsymbol{\lambda}}\|^2\right) \end{aligned} \tag{6.78}$$

which establishes (6.24). Upon selecting $\alpha = \mathcal{O}(T^{-\frac{3}{4}})$, and $\delta = \mathcal{O}(T^{-\frac{1}{4}})$, we find from Lemma 25 that $\|\bar{\boldsymbol{\lambda}}\| \leq C = \mathcal{O}(1)$. Together with $\mu = \mathcal{O}(T^{-\frac{3}{4}})$ and $\beta = \mathcal{O}(T^{\frac{1}{4}})$, it holds that

$$\begin{aligned}\mathrm{Fit}_T^{\mathrm{d}} \leq & CT^{\frac{3}{4}} + \frac{G^2\sqrt{N}T^{\frac{3}{4}}}{2} + G\sqrt{N}T^{\frac{3}{4}} \\ & + \sqrt{N}T^{\frac{5}{4}}\left(d^2F^2T^{-1} + T^{-\frac{3}{2}}G^2C^2\right) = \mathcal{O}(T^{\frac{3}{4}}) \end{aligned} \tag{6.79}$$

which completes the proof of (6.25).

6.6.4 Proof of Theorem 12

The proof is similar to that of Theorem 11, and it is included here for completeness. The feasibility of actions $\{\mathbf{x}_{1,t}, \mathbf{x}_{2,t}\}$ readily follows from Lemma 22. To prove the dynamic regret and fit bounds in this setup, the following result is needed.

Lemma 26 (Bound on dual variables)**.** *For the BanSaP recursion, selecting* $\alpha = \mu = \mathcal{O}(T^{-\frac{1}{2}})$ *ensures that the dual iterates are bounded by* $\|\boldsymbol{\lambda}_t\| \leq C = \mathcal{O}(1)$*, with the constant given by*

$$C := \max\left\{2GR, \left(\frac{1}{\eta}+1\right)GR + \frac{2G^2R^2\mu}{\eta} + \frac{d^2G^2\alpha}{\eta} + \frac{\mu R^2}{2\alpha\eta}\right\} \tag{6.80}$$

where the constants G, R, and η *are as in (as2)-(as4).*

Proof: It follows the same steps as those used to prove Lemma 25.

Lemma 26 asserts that the dual variable in BanSaP with two-point bandit feedback is also uniformly bounded from above. Now, we are ready to prove the regret bound in Theorem 12.

Dynamic regret in Theorem 2: To obtain the regret bound in the two-point case, our first step is to connect the regret with the loss induced by the virtual iterates $\{\hat{\mathbf{x}}_t\}$, given by

$$\begin{aligned} & \frac{1}{2}\Big(\mathbb{E}[f_t(\mathbf{x}_{1,t})] + \mathbb{E}[f_t(\mathbf{x}_{2,t})]\Big) - f_t(\mathbf{x}_t^*) \\ \leq & \frac{1}{2}\Big(\mathbb{E}[f_t(\hat{\mathbf{x}}_t)] + \delta G + \mathbb{E}[f_t(\hat{\mathbf{x}}_t)] + \delta G\Big) - f_t(\mathbf{x}_t^*) \end{aligned}$$

$$=\Big(\mathbb{E}[f_t(\hat{\mathbf{x}}_t)] - f_t(\mathbf{x}_t^*)\Big) + \delta G \tag{6.81}$$

where the inequality follows from the Lipschitz condition.

The LHS of (6.81) can be further decomposed as

$$\underbrace{\mathbb{E}[f_t(\hat{\mathbf{x}}_t)] - \mathbb{E}[\check{f}_t(\hat{\mathbf{x}}_t)]}_{U_1} + \underbrace{\mathbb{E}[\check{f}_t(\hat{\mathbf{x}}_t)] - \check{f}_t((1-\gamma)\mathbf{x}_t^*))}_{U_2}$$
$$+ \underbrace{\check{f}_t((1-\gamma)\mathbf{x}_t^*)) - \check{f}_t(\mathbf{x}_t^*)}_{U_3} + \underbrace{\check{f}_t(\mathbf{x}_t^*) - f_t(\mathbf{x}_t^*)}_{U_4} + \delta G. \tag{6.82}$$

For the first term, following the steps in (6.71), we have that

$$U_1 \le \mathbb{E}\left[f_t(\hat{\mathbf{x}}_t) - f_t(\mathbb{E}_{\mathbf{v}}[\hat{\mathbf{x}} + \delta \mathbf{v}_t])\right] \le 0. \tag{6.83}$$

Similar to (6.69), we have for the case of two-point feedback

$$\sum_{t=1}^{T} U_2 \le \gamma G R \|\bar{\boldsymbol{\lambda}}\| T + \frac{R}{\alpha} V(\mathbf{x}_{1:T}^*) + 2\mu G^2 R^2 T + \frac{R^2}{2\alpha} + \alpha d^2 G^2 T.$$

Using the Lipschitz condition of $\check{f}_t(\mathbf{x})$, U_3 is bounded by

$$U_3 = \check{f}_t((1-\gamma)\mathbf{x}_t^*)) - \check{f}_t(\mathbf{x}_t^*) \le \gamma G R \tag{6.84}$$

and for U_4, it follows from the Lipschitz condition that

$$U_4 = \mathbb{E}_{\mathbf{v}}[f_t(\mathbf{x}_t^* + \delta \mathbf{v}_t)] - f_t(\mathbf{x}_t^*) \le \delta G. \tag{6.85}$$

Summing up (6.81) over t, and plugging (6.83)-(6.85), we have

$$\frac{1}{2}\sum_{t=1}^{T}\Big(\mathbb{E}[f_t(\mathbf{x}_{1,t})] + \mathbb{E}[f_t(\mathbf{x}_{2,t})]\Big) - \sum_{t=1}^{T} f_t(\mathbf{x}_t^*) \le \frac{R}{\alpha} V(\mathbf{x}_{1:T}^*)$$
$$+ \frac{R^2}{2\alpha} + 2\mu G^2 R^2 T + \alpha d^2 G^2 T + \gamma G R T (1 + \|\bar{\boldsymbol{\lambda}}\|) + 2\delta G T. \tag{6.86}$$

Upon choosing $\alpha = \mu = \mathcal{O}(T^{-\frac{1}{2}})$, and $\delta = \mathcal{O}(T^{-1})$ along with $\gamma = \delta / r$, it follows that

(ignoring constant terms)

$$\mathrm{Reg}_T^{\mathrm{d}} = \mathcal{O}\Big(RV(\mathbf{x}_{1:T}^*)T^{\frac{1}{2}} + \frac{1}{2}R^2T^{\frac{1}{2}} + 2G^2R^2T^{\frac{1}{2}} + d^2G^2T^{\frac{1}{2}}\Big)$$

where we used Lemma 26. This completes the proof of (6.26).

Dynamic fit in Theorem 2: Regarding the dynamic fit, recall that the constraint violations in (6.39) depend on the norm of dual variables and the difference of consecutive primal iterates. While the dual variable has been bounded in Lemma 26, the distance between iterates $\mathbf{x}_t$ and $\hat{\mathbf{x}}_{t+1}$ can be bounded by

$$\begin{aligned} \|\hat{\mathbf{x}}_{t+1} - \hat{\mathbf{x}}_t\| \leq \big\|\alpha\hat{\nabla}_{\mathbf{x}}^2\mathcal{L}_t(\hat{\mathbf{x}}_t, \boldsymbol{\lambda}_t)\big\| \\ \leq \|\hat{\nabla}^2 f_t(\hat{\mathbf{x}}_t)\| + \|\nabla\mathbf{g}_t(\hat{\mathbf{x}}_t)\|\|\boldsymbol{\lambda}_t\| \leq \alpha dG + \alpha G\|\bar{\boldsymbol{\lambda}}\|. \end{aligned} \tag{6.87}$$

In addition, similar to the step in (6.77), we arrive at

$$\begin{aligned} &\frac{1}{2}\sum_{t=1}^{T}(\mathbf{g}_t(\mathbf{x}_{1,t}) + \mathbf{g}_t(\mathbf{x}_{2,t})) \qquad (6.88)\\ &\overset{(a)}{\leq} \frac{\boldsymbol{\lambda}_{T+1}}{\mu} + \frac{G^2T\mathbf{1}}{2\beta} + \beta T\Big(\alpha^2d^2G^2 + \alpha^2G^2\|\bar{\boldsymbol{\lambda}}\|^2\Big)\mathbf{1} + \delta GT\mathbf{1} \end{aligned}$$

where (a) uses (6.87) instead of (6.76) used in (6.77).

If we take $[\cdot]^+$ and $\|\cdot\|$ on both sides of (6.88), and choose $\alpha = \mu = \mathcal{O}(T^{-\frac{1}{2}})$, $\delta = T^{-1}$, and $\beta = \mathcal{O}(T^{\frac{1}{2}})$, we arrive at

$$\begin{aligned} \mathrm{Fit}_T^{\mathrm{d}} \leq & \frac{\|\boldsymbol{\lambda}_{T+1}\|}{\mu} + \frac{G^2N^{\frac{1}{2}}T}{2\beta} + \beta N^{\frac{1}{2}}T\Big(\alpha^2d^2G^2 + \alpha^2G^2\|\bar{\boldsymbol{\lambda}}\|^2\Big) \\ = & CT^{\frac{1}{2}} + N^{\frac{1}{2}}T^{\frac{1}{2}}G^2\left(\frac{1}{2} + d^2 + C^2\right) = \mathcal{O}\left(T^{\frac{1}{2}}\right) \end{aligned} \tag{6.89}$$

where we used the bound on dual variables in Lemma 26. This completes also the proof of (6.28), and also that of Theorem 12.

Chapter 7

Summary and future directions

In this final chapter, we provide a summary of the main results discussed in this thesis, and also point out a few promising directions for future research.

7.1 Thesis summary

This thesis presented a set of contributions at the intersection of optimization, machine learning and networked systems such as IoT. The focus was on building fundamental connections between methodologies from machine learning, optimization, and networking communities, and developing inter-disciplinary approaches for IoT.

In the first part of the thesis, which contains Chapters 2 and 3, the aim was to develop communication-efficient distributed learning methods amenable to efficient implementation in the IoT paradigm with ubiquitous devices. The novel methods are simple and general, thus facilitating application to (un-/semi-)supervised learning and reinforcement learning tasks.

Chapter 2 dealt with the federated learning problem emerging in IoT, and developed a promising communication-cognizant method for distributed machine learning that we term Lazily Aggregated Gradient (LAG) approach. LAG can achieve the same convergence rates as batch gradient descent (GD) in smooth strongly-convex, convex, and nonconvex cases, and requires fewer communication rounds than GD given that the datasets at different workers are heterogeneous. Confirmed by the impressive empirical performance on both synthetic and real datasets, LAG is expected to bring valuable insights to the future algorithm design.

Chapter 3 studied the distributed reinforcement learning (DRL) problem involving a central controller and a group of heterogeneous learners, which includes the popular *multi-agent* *collaborative* RL and the *parallel* RL settings. Targeting DRL applications in communication-constrained environments, the goal was to learn a DRL policy minimizing the loss aggregated over all learners, using as few communication rounds as possible. We developed a promising communication-cognizant method for DRL that we term Lazily Aggregated Policy Gradient (LAPG) approach. LAPG can achieve the same convergence rates as PG, and requires fewer communication rounds given that the learners in DRL are heterogeneous. Promising empirical performance on the multi-agent cooperative navigation task corroborated our theoretical findings.

The second part of the thesis, which includes Chapters 4-6, introduced a class of online resource management approaches, which are adaptive to different stationarity assumptions of the IoT operation, and different levels of available information in the complex environment.

Leveraging recent advances in statistical learning and optimization, a novel online approach termed LA-SDG was developed in Chapter 4. LA-SDG learns the network state statistics through an additional sample recourse procedure. The associated novel iteration can be nicely interpreted as a modified heavy-ball recursion with an extra correction step to mitigate steady-state oscillations. It was analytically established that LA-SDG achieves a near-optimal cost-delay tradeoff $[\mu, \log^2(\mu)/\sqrt{\mu}]$, which is better than $[\mu, 1/\mu]$ of stochastic dual gradient (SDG), at the cost of only one extra gradient evaluation per new datum. A future research agenda can include novel approaches to further hedge against non-stationarity, and improved learning schemes to uncover other valuable statistical patterns from historical data.

Chapter 5 tackled the network resource management problem from the perspective of online convex optimization (OCO) with both adversarial costs and constraints. Different from existing works, the focus is on a setting where some of the constraints are revealed after taking actions, they are tolerable to instantaneous violations, but must be satisfied on average. Performance of the novel OCO algorithm is measured by: i) the difference of its objective relative to the best dynamic solution with one-slot-ahead information of the cost and the constraint (dynamic regret); and, ii) its accumulated amount of constraint violations (dynamic fit). It has been shown that the proposed MOSP algorithm adapts to the considered OCO setting with adversarial constraints. Under standard assumptions, MOSP simultaneously yields sub-linear dynamic regret and fit, if the accumulated variations of the per-slot minimizers and adversarial constraints are sub-linearly increasing with time. Algorithm design and performance analysis in this novel OCO setting, under adversarial constraints and with a dynamic benchmark, broaden the applicability of OCO to a wider application regime, which includes dynamic network resource allocation and online demand response in smart grids. Numerical tests demonstrated that the proposed algorithm outperforms state-of-the-art alternatives under different scenarios.

Chapter 6 studied the network resource management problem from the vantage point of bandit convex optimization (BCO). Different from existing works in bandit settings, the focus was on a broader setting where part of the constraints are revealed after taking actions, and are also tolerable to instantaneous violations but have to be satisfied on average. The novel BCO setting fits well the emerging fog computing tasks in IoT. A class of online bandit saddle-point (BanSaP) approaches were proposed, and their online performance was rigorously analyzed. It was shown that the resultant regret bounds match those attained in BCO setups without long-term constraints. Furthermore, the BanSaP solvers can simultaneously yield sub-linear dynamic regret and fit, if the dynamic solutions vary slowly over time.

7.2 Future research directions

Moving forward, I will continue my research on developing scalable learning approaches for intelligent systems including IoT. I will develop a research program that poses problems of practical interest and addresses their theoretical challenges. Below is an outline of thrusts I aim to pursue.

7.2.1 Risk-averse learning and computing

While IoT promises major benefits drawn from the seamless integration of machine learning and AI, the critical concerns on their safe deployment cannot be understated. However, a number of devices in IoT may be highly unreliable or even easily compromised by hackers. In this scenario, the current edge computing paradigm lacks secure training ability, which renders it vulnerable to failures, not mentioning adversarial attacks. For example, stochastic gradient descent, the workhorse of large-scale learning, is vulnerable to even one malicious device. Such levels of risk provide ample drive for fundamental research efforts to fulfill the desiderata of safe AI for IoT. To this end, my research agenda seeks both communication-efficient and risk-averse approaches for the entire gamut of federated learning problems, from supervised to unsurprised learning, as well as reinforcement learning that is essential for pushing forward the autonomous driving techniques. Inspired by robust estimation of statistical signal processing, I will investigate robust information aggregation from heterogeneous devices, and put forth a class of resilient learning approaches with provable performance guarantees.

7.2.2 Communication and machine learning co-design

The overarching goal of edge computing is to fast extract intelligence from the large volume of data distributed at IoT devices. This critically depends on machine learning approaches that run on edge servers, as well as efficient communication between edge servers and devices. Unfortunately, the traditional design principle of communication systems, namely low packet loss and high data rate, does not account for the need of running iterative learning approaches at the edge. On the other hand, the pursuit of machine learning research, namely high model expressibility and low learning accuracy, does not optimize for the existing communication network infrastructure. While research efforts in wireless communication and machine learning have so far evolved separately, my strong belief is that they will eventually converge in the forthcoming IoT paradigm. With the co-design of communication and machine learning, communication can exploit the insights gained from learning algorithms, while learning can become more cost-effective.

References

[1] "Midcontinent independent system operator (MISO) locational marginal price," Nov. 2016. [Online]. Available: http://www.misoenergy.org/MarketsOperations/Prices

[2] "8 Sensors to Help You Create A Smart Home," Dec. 2017. [Online]. Available: www.ibm.com/blogs/internet-of-things/sensors-smart-home/

[3] A. Agarwal, O. Dekel, and L. Xiao, "Optimal algorithms for online convex optimization with multi-point bandit feedback." in *Proc. Annual Conf. on Learning Theory*, Haifa, Israel, 2010, pp. 28–40.

[4] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[5] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "Qsgd: Communication-efficient sgd via gradient quantization and encoding," in *Proc. Advances in Neural Info. Process. Syst.*, Long Beach, CA, Dec. 2017, pp. 1709–1720.

[6] L. L. Andrew, S. Barman, K. Ligett, M. Lin, A. Meyerson, A. Roytman, and A. Wierman, "A tale of two metrics: Simultaneous bounds on competitiveness and regret," in *Proc. Annual Conf. on Learning Theory*, Princeton, NJ, Jun. 2013.

[7] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[8] B. Awerbuch and R. D. Kleinberg, "Adaptive routing with end-to-end feedback: Distributed learning and geometric approaches," in *Proc. ACM Symp. on Theory of Computing*, Chicago, IL, Jun. 2004, pp. 45–53.

[9] S. Barbarossa, S. Sardellitti, E. Ceci, and M. Merluzzi, "The edge cloud: A holistic view of communication, computation and caching," in *Cooperative and Graph Signal Processing*, P. Djuric and C. Richard, Eds. Springer, 2018. [Online]. Available: arXivpreprint:1802.00700

[10] S. Barbarossa, S. Sardellitti, and P. Di Lorenzo, "Communicating while computing: Distributed mobile cloud computing over 5G heterogeneous networks," *IEEE Sig. Proc. Mag.*, vol. 31, no. 6, pp. 45–55, 2014.

[11] J. Baxter and P. L. Bartlett, "Infinite-horizon policy-gradient estimation," *J. Artificial Intelligence Res.*, vol. 15, pp. 319–350, 2001.

[12] A. Beck, A. Nedic, A. Ozdaglar, and M. Teboulle, "An $\mathcal{O}(1/k)$ gradient method for network resource allocation problems," *IEEE Trans. Control Netw. Syst.*, vol. 1, no. 1, pp. 64–73, Mar. 2014.

[13] D. P. Bertsekas, *Nonlinear Programming*. Belmont, MA: Athena scientific, 1999.

[14] D. P. Bertsekas, A. Nedic, and A. Ozdaglar, *Convex Analysis and Optimization*. Belmont, MA: Athena Scientific, 2003.

[15] O. Besbes, Y. Gur, and A. Zeevi, "Non-stationary stochastic optimization," *Operations Research*, vol. 63, no. 5, pp. 1227–1244, Sep. 2015.

[16] D. Blatt, A. O. Hero, and H. Gauchman, "A convergent incremental gradient method with a constant step size," *SIAM J. Optimization*, vol. 18, no. 1, pp. 29–51, Feb. 2007.

[17] V. S. Borkar, "Convex analytic methods in markov decision processes," in *Handbook of Markov decision processes*. Springer, 2002, pp. 347–375.

[18] L. Bottou, "Large-Scale Machine Learning with Stochastic Gradient Descent," in *Proceedings of COMPSTAT'2010*, Y. Lechevallier and G. Saporta, Eds. Heidelberg: Physica-Verlag HD, 2010, pp. 177–186.

[19] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *arXiv preprint:1606.04838*, Jun. 2016.

[20] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," in *Proc. Advances in Neural Info. Process. Syst.*, Denver, CO, Nov. 1994, pp. 671–678.

[21] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “OpenAI Gym,” *arXiv:1606.01540*, 2016. [Online]. Available: https://github.com/openai/gym

[22] S. Bubeck, N. Cesa-Bianchi *et al.*, “Regret analysis of stochastic and nonstochastic multi-armed bandit problems,” *Found. and Trends in Mach. Learn.*, vol. 5, no. 1, pp. 1–122, 2012.

[23] L. Cannelli, F. Facchinei, V. Kungurtsev, and G. Scutari, “Asynchronous parallel algorithms for nonconvex big-data optimization: Model and convergence,” *arXiv preprint:1607.04818*, Jul. 2016.

[24] N. Chen, J. Comden, Z. Liu, A. Gandhi, and A. Wierman, “Using predictions in online optimization: Looking forward with an eye on the past,” in *Proc. ACM SIGMETRICS*, Antibes Juan-les-Pins, France, Jun. 2016, pp. 193–206.

[25] T. Chen, Q. Ling, and G. B. Giannakis, “An online convex optimization approach to proactive network resource allocation,” *IEEE Trans. Signal Process.*, Jan. 2017 (revised), Available: https://arxiv.org/abs/1701.03974.

[26] T. Chen, A. Mokhtari, X. Wang, A. Ribeiro, and G. B. Giannakis, “Stochastic averaging for constrained optimization with application to online resource allocation,” *IEEE Trans. Signal Process.*, vol. 65, no. 12, pp. 3078–3093, Jun. 2017.

[27] T. Chen, Y. Shen, Q. Ling, and G. B. Giannakis, “Online learning for “thing-adaptive” fog computing in IoT,” in *Proc. of Asilomar Conf.*, Pacific Grove, CA, Oct. 2017. [Online]. Available: www.dropbox.com/s/z4qnog6x0gzd2ko/TAOSP.pdf?dl=0

[28] T. Chen, X. Wang, and G. B. Giannakis, “Cooling-aware energy and workload management in data centers via stochastic optimization,” *IEEE J. Sel. Topics Signal Process.*, vol. 10, no. 2, pp. 402–415, Mar. 2016.

[29] T. Chen, K. Zhang, G. B. Giannakis, and T. Başar, “Random feature-based online multi-kernel learning in environments with unknown dynamics,” *Journal of Machine Learning Research*, Dec. 2018 (submitted).

[30] T. Chen, S. Barbarossa, X. Wang, G. B. Giannakis, and Z.-L. Zhang, “Learning and management for Internet-of-Things: Accounting for adaptivity and scalability,” *Proc. of the IEEE*, Nov. 2018.

[31] T. Chen, G. B. Giannakis, T. Sun, and W. Yin, “LAG: Lazily aggregated gradient for communication-efficient distributed learning,” in *Proc. Advances in Neural Info. Process. Syst.*, Montreal, Canada, Dec. 2018. [Online]. Available: arxiv.org/abs/1805.09965

[32] T. Chen, Q. Ling, and G. B. Giannakis, “Learn-and-adapt stochastic dual gradients for network resource allocation,” *IEEE Trans. Control Netw. Syst.*, revised, Jun. 2017. [Online]. Available: https://arxiv.org/pdf/1703.01673.pdf

[33] T. Chen, A. G. Marques, and G. B. Giannakis, “DGLB: Distributed stochastic geographical load balancing over cloud networks,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 7, pp. 1866–1880, Jul. 2017.

[34] T. Chen, Y. Shen, Q. Ling, and G. B. Giannakis, “Heterogeneous online learning for ‘thing-adaptive’ low-latency fog computing in IoT,” *IEEE Internet Things J.*, Oct. 2018, to appear.

[35] T. Chen, Y. Zhang, X. Wang, and G. B. Giannakis, “Robust workload and energy management for sustainable data centers,” *IEEE J. Sel. Areas Commun.*, vol. 34, no. 3, pp. 651–664, Mar. 2016.

[36] X. Chen, W. Ni, T. Chen, I. B. Collings, X. Wang, R. P. Liu, and G. B. Giannakis, “Multi-timescale online optimization of network function virtualization for service chaining,” *arXiv preprint:1804.07051*, Apr. 2018.

[37] X. Chen, L. Jiao, W. Li, and X. Fu, “Efficient multi-user computation offloading for mobile-edge cloud computing,” *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[38] M. Chiang and T. Zhang, “Fog and IoT: An overview of research opportunities,” *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, 2016.

[39] W. Chu, M. Dehghan, J. C. Lui, D. Towsley, and Z.-L. Zhang, “Joint cache resource allocation and request routing for in-network caching services,” *Computer Networks*, vol. 131, pp. 1–14, Feb. 2018.

[40] W. Chu, M. Dehghan, D. Towsley, and Z.-L. Zhang, “On allocating cache resources to content providers,” in *Proc. ACM Conf. on Info.-Centric Netw.*, Kyoto, Japan, Sep. 2016, pp. 154–159.

[41] C. Claus and C. Boutilier, "The dynamics of reinforcement learning in cooperative multiagent systems," in *Proc. of the Assoc. for the Advanc. of Artificial Intell.*, Orlando, FL, Oct. 1998, pp. 746–752.

[42] E. Dall'Anese and A. Simonetto, "Optimal power flow pursuit," *IEEE Trans. Smart Grid*, vol. 9, no. 2, pp. 942–952, Mar. 2018.

[43] D. Davis and W. Yin, "Convergence Rate Analysis of Several Splitting Schemes," in *Splitting Methods in Communication, Imaging, Science, and Engineering*. New York: Springer, 2016.

[44] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le *et al.*, "Large scale distributed deep networks," in *Proc. Advances in Neural Info. Process. Syst.*, Lake Tahoe, NV, 2012, pp. 1223–1231.

[45] A. Defazio, F. Bach, and S. Lacoste-Julien, "Saga: A fast incremental gradient method with support for non-strongly convex composite objectives," in *Proc. Advances in Neural Info. Process. Syst.*, Montreal, Canada, Dec. 2014, pp. 1646–1654.

[46] M. P. Deisenroth, *Efficient Reinforcement Learning Using Gaussian Processes*. Karlsruhe, Germany: KIT Scientific Publishing, 2010, vol. 9.

[47] L. Duan, L. Huang, C. Langbort, A. Pozdnukhov, J. Walrand, and L. Zhang, "Human-in-the-loop mobile networks: a survey of recent advancements," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 4, pp. 813–831, Apr. 2017.

[48] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Machine Learning Res.*, vol. 12, pp. 2121–2159, Jul. 2011.

[49] J. C. Duchi, A. Agarwal, M. Johansson, and M. I. Jordan, "Ergodic mirror descent," *SIAM J. Optimization*, vol. 22, no. 4, pp. 1549–1578, 2012.

[50] J. C. Duchi, M. I. Jordan, M. J. Wainwright, and A. Wibisono, "Optimal rates for zero-order convex optimization: The power of two function evaluations," *IEEE Trans. Inf. Theory*, vol. 61, no. 5, pp. 2788–2806, May 2015.

[51] A. Eryilmaz and R. Srikant, "Joint congestion control, routing, and MAC for stability and fairness in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1514–1524, Aug. 2006.

[52] A. D. Flaxman, A. T. Kalai, and H. B. McMahan, “Online convex optimization in the bandit setting: gradient descent without a gradient,” in *Proc. of ACM SODA*, Vancouver, Canada, Jan. 2005, pp. 385–394.

[53] L. Georgiadis, M. Neely, and L. Tassiulas, “Resource allocation and cross-layer control in wireless networks,” *Found. and Trends in Networking*, vol. 1, pp. 1–144, 2006.

[54] B. Gerhardt, K. Griffin, and R. Klemann, “Unlocking value in the fragmented world of big data analytics,” *Cisco Internet Business Solutions Group*, Jun. 2012.

[55] E. Ghadimi, I. Shames, and M. Johansson, “Multi-step gradient methods for networked optimization,” *IEEE Trans. Signal Process.*, vol. 61, no. 21, pp. 5417–5429, Nov. 2013.

[56] G. B. Giannakis, V. Kekatos, N. Gatsis, S.-J. Kim, H. Zhu, and B. F. Wollenberg, “Monitoring and optimization for power grids: A signal processing perspective,” *IEEE Sig. Proc. Mag.*, vol. 30, no. 5, pp. 107–128, Sep. 2013.

[57] G. B. Giannakis, Q. Ling, G. Mateos, I. D. Schizas, and H. Zhu, “Decentralized Learning for Wireless Communications and Networking,” in *Splitting Methods in Communication and Imaging, Science and Engineering*. New York: Springer, 2016.

[58] J. Gregoire, X. Qian, E. Frazzoli, A. de La Fortelle, and T. Wongpiromsarn, “Capacity-aware backpressure traffic signal control,” *IEEE Trans. Control Netw. Syst.*, vol. 2, no. 2, pp. 164–173, June 2015.

[59] J. K. Gupta, M. Egorov, and M. Kochenderfer, “Cooperative multi-agent control using deep reinforcement learning,” in *Intl. Conf. Auto. Agents and Multi-agent Systems*, 2017, pp. 66–83.

[60] M. Gurbuzbalaban, A. Ozdaglar, and P. A. Parrilo, “On the convergence rate of incremental aggregated gradient algorithms,” *SIAM J. Optimization*, vol. 27, no. 2, pp. 1035–1048, Jun. 2017.

[61] H. A. Güvenir, G. Demiröz, and N. Ilter, “Learning differential diagnosis of erythemato-squamous diseases using voting feature intervals,” *Artificial Intelligence in Medicine*, vol. 13, no. 3, pp. 147–165, 1998.

[62] E. C. Hall and R. M. Willett, “Online convex optimization in dynamic environments,” *IEEE J. Sel. Topics Signal Process.*, vol. 9, no. 4, pp. 647–662, Jun. 2015.

[63] D. Harrison Jr. and D. L. Rubinfeld, “Hedonic housing prices and the demand for clean air,” *Journal of environmental economics and management*, vol. 5, no. 1, pp. 81–102, Mar. 1978.

[64] E. Hazan, “Introduction to online convex optimization,” *Found. and Trends in Mach. Learn.*, vol. 2, no. 3-4, pp. 157–325, 2016.

[65] E. Hazan, A. Agarwal, and S. Kale, “Logarithmic regret algorithms for online convex optimization,” *Machine Learning*, vol. 69, no. 2-3, pp. 169–192, Dec. 2007.

[66] M. Hong and Z.-Q. Luo, “On the linear convergence of the alternating direction method of multipliers,” *Math. Program., Ser. A*, pp. 1–35, 2016.

[67] H. Huang, Q. Ling, W. Shi, and J. Wang, “Collaborative resource allocation over a hybrid cloud center and edge server network,” *Journal of Computational Mathematics*, 2017, to appear.

[68] L. Huang, S. Zhang, M. Chen, and X. Liu, “When backpressure meets predictive scheduling,” *IEEE/ACM Trans. Netw.*, vol. 24, no. 4, pp. 2237–2250, Aug. 2016.

[69] L. Huang, X. Liu, and X. Hao, “The power of online learning in stochastic network optimization,” in *Proc. ACM SIGMETRICS*, vol. 42, no. 1, New York, NY, Jun. 2014, pp. 153–165.

[70] L. Huang and M. J. Neely, “Delay reduction via Lagrange multipliers in stochastic network optimization,” *IEEE Trans. Autom. Contr.*, vol. 56, no. 4, pp. 842–857, Apr. 2011.

[71] A. Jadbabaie, A. Rakhlin, S. Shahrampour, and K. Sridharan, “Online optimization: Competing with dynamic comparators,” in *Intl. Conf. Artificial Intell. and Stat.*, San Diego, CA, May 2015.

[72] M. Jaggi, V. Smith, M. Takác, J. Terhorst, S. Krishnan, T. Hofmann, and M. I. Jordan, “Communication-efficient distributed dual coordinate ascent,” in *Proc. Advances in Neural Info. Process. Syst.*, Montreal, Canada, Dec. 2014, pp. 3068–3076.

[73] M. I. Jordan, J. D. Lee, and Y. Yang, “Communication-efficient distributed statistical inference,” *J. American Statistical Association*, vol. to appear, 2018.

[74] S. M. Kakade, “A natural policy gradient,” in *Proc. Advances in Neural Info. Process. Syst.*, Vancouver, Canada, Dec. 2002, pp. 1531–1538.

[75] Y.-H. Kao, K. Wright, B. Krishnamachari, and F. Bai, “Online learning for wireless distributed computing,” *arXiv preprint:1611.02830*, Nov. 2016. [Online]. Available: https://arxiv.org/abs/1611.02830

[76] S. Kar, J. M. Moura, and H. V. Poor, “QD-learning: A collaborative distributed strategy for multi-agent reinforcement learning through Consensus + Innovations,” *IEEE Trans. Sig. Proc.*, vol. 61, no. 7, pp. 1848–1862, Jul. 2013.

[77] H. Karimi, J. Nutini, and M. Schmidt, “Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition,” in *Proc. Euro. Conf. Machine Learn.*, Riva del Garda, Italy, 2016, pp. 795–811.

[78] S. J. Kim and G. Giannakis, “An online convex optimization approach to real-time energy pricing for demand response,” *IEEE Trans. Smart Grid*, to appear, 2017.

[79] R. Kohavi, “Scaling up the accuracy of Naive-Bayes classifiers: a decision-tree hybrid.” in *Proc. of KDD*, vol. 96, Portland, OR, Aug. 1996, pp. 202–207.

[80] V. R. Konda and J. N. Tsitsiklis, “Actor-critic algorithms,” in *Proc. Advances in Neural Info. Process. Syst.*, Denver, CO, Dec. 2000, pp. 1008–1014.

[81] V. Kong and X. Solo, *Adaptive Signal Processing Algorithms*. Upper Saddle River, NJ: Prentice Hall, 1995.

[82] A. Koppel, F. Y. Jakubiec, and A. Ribeiro, “A saddle point algorithm for networked online convex optimization,” *IEEE Trans. Signal Process.*, vol. 63, no. 19, pp. 5149–5164, Oct. 2015.

[83] G. Lan, S. Lee, and Y. Zhou, “Communication-efficient algorithms for decentralized and stochastic optimization,” *arXiv preprint:1701.03961*, Jan. 2017.

[84] M. Lauer and M. Riedmiller, “An algorithm for distributed reinforcement learning in cooperative multi-agent systems,” in *Proc. Intl. Conf. Machine Learn.*, Stanford, CA, Jun. 2000.

[85] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[86] J. D. Lee, Q. Lin, T. Ma, and T. Yang, "Distributed stochastic variance reduced gradient methods by sampling extra data with replacement," *J. Machine Learning Res.*, vol. 18, no. 1, pp. 4404–4446, 2017.

[87] B. Li, T. Chen, and G. B. Giannakis, "Bandit online learning with unknown delays," in *Proc. of Intl. Conf. on Artificial Intelligence and Statistics*, Naha, Japan, Apr. 2019.

[88] B. Li, T. Chen, X. Wang, and G. B. Giannakis, "Real-time energy management in microgrids with reduced battery capacity requirements," *IEEE Trans. Smart Grids*, 2018, to appear.

[89] L. Li, W. Xu, T. Chen, G. B. Giannakis, and Q. Ling, "Rsa: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets," in *AAAI*, Honolulu, Hawaii, Jan. 2019. [Online]. Available: http://arxiv.org/abs/1811.03761

[90] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling distributed machine learning with the parameter server," in *Proc. USENIX Symp. Operating Syst. Design and Implement.*, vol. 14, Broomfield, CO, Oct. 2014, pp. 583–598.

[91] M. Li, D. G. Andersen, A. J. Smola, and K. Yu, "Communication efficient distributed machine learning with the parameter server," in *Proc. Advances in Neural Info. Process. Syst.*, Montreal, Canada, Dec. 2014, pp. 19–27.

[92] Y. Li and D. Schuurmans, "Mapreduce for parallel reinforcement learning," in *European Workshop on Reinforcement Learning*. Springer, 2011, pp. 309–320.

[93] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[94] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *Proc. Intl. Conf. Learn. Representations*, San Juan, Puerto Rico, May 2016.

[95] J. D. Little, "A proof for the queuing formula: L=λw," *Operations research*, vol. 9, no. 3, pp. 383–387, Jun. 1961.

[96] J. Liu, S. Wright, C. Ré, V. Bittorf, and S. Sridhar, "An asynchronous parallel stochastic coordinate descent algorithm," *J. Machine Learning Res.*, vol. 16, no. 1, pp. 285–322, 2015.

[97] J. Liu, A. Eryilmaz, N. B. Shroff, and E. S. Bentley, "Heavy-ball: A new approach to tame delay and convergence in wireless network optimization," in *Proc. IEEE INFOCOM*, San Francisco, CA, Apr. 2016.

[98] Y. Liu, C. Nowzari, Z. Tian, and Q. Ling, "Asynchronous periodic event-triggered coordination of multi-agent systems," in *Proc. IEEE Conf. Decision Control*, Melbourne, Australia, Dec. 2017, pp. 6696–6701.

[99] S. H. Low and D. E. Lapsley, "Optimization flow control-I: Basic algorithm and convergence," *IEEE/ACM Trans. Netw.*, vol. 7, no. 6, pp. 861–874, Dec. 1999.

[100] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Advances in Neural Info. Process. Syst.*, Long beach, CA, Dec. 2017.

[101] N. Lu, N. Cheng, N. Zhang, X. Shen, and J. W. Mark, "Connected vehicles: Solutions and challenges," *IEEE Internet Things J.*, vol. 1, no. 4, pp. 289–299, Aug. 2014.

[102] D. Lymberopoulos, A. Bamis, and A. Savvides, "Extracting spatiotemporal human activity patterns in assisted living using a home sensor network," *Univ Access Info. Soc*, vol. 10, no. 2, pp. 125–138, 2011.

[103] X. Lyu, W. Ni, H. Tian, R. P. Liu, X. Wang, G. B. Giannakis, and A. Paulraj, "Optimal schedule of mobile edge computing for internet of things using partial information," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2606–2615, Nov. 2017.

[104] C. Ma, J. Konečnỳ, M. Jaggi, V. Smith, M. I. Jordan, P. Richtárik, and M. Takáč, "Distributed optimization with arbitrary local solvers," *Optimization Methods and Software*, vol. 32, no. 4, pp. 813–848, Jul. 2017.

[105] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Comm. Surveys & Tutorials*, 2017, to appear.

[106] M. Mahdavi, R. Jin, and T. Yang, "Trading regret for efficiency: Online convex optimization with long term constraints," *Journal of Machine Learning Research*, vol. 13, pp. 2503–2528, Sep 2012.

[107] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "Mobile edge computing: Survey and research outlook," *arXiv preprint:1701.01090*, Jan. 2017.

[108] A. G. Marques, L. M. Lopez-Ramos, G. B. Giannakis, J. Ramos, and A. J. Caamaño, "Optimal cross-layer resource allocation in cellular networks using channel-and queue-state information," *IEEE Trans. Veh. Technol.*, vol. 61, no. 6, pp. 2789–2807, Jul. 2012.

[109] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Intl. Conf. Artificial Intell. and Stat.*, Fort Lauderdale, FL, Apr. 2017, pp. 1273–1282.

[110] B. McMahan and D. Ramage, "Federated learning: Collaborative machine learning without centralized training data," *Google Research Blog*, Apr. 2017. [Online]. Available: https://research.googleblog.com/2017/04/federated-learning-collaborative.html

[111] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. Intl. Conf. Machine Learn.*, New York City, NY, Jun. 2016, pp. 1928–1937.

[112] A. Mokhtari, S. Shahrampour, A. Jadbabaie, and A. Ribeiro, "Online optimization in dynamic environments: Improved regret rates for strongly convex problems," in *Proc. IEEE Conf. on Decision and Control*, Las Vegas, NV, Dec. 2016.

[113] S. Munir, J. A. Stankovic, C.-J. M. Liang, and S. Lin, "Cyber physical system challenges for human-in-the-loop control." in *Proc. Feedback Computing*, San Jose, CA, Jun. 2013.

[114] A. Nair, P. Srinivasan, S. Blackwell, C. Alcicek, R. Fearon, A. De Maria, V. Panneershelvam, M. Suleyman, C. Beattie, S. Petersen *et al.*, "Massively parallel methods for deep reinforcement learning," in *Proc. Intl. Conf. Machine Learn. on Deep Learn. Workshop*, Lille, France, Jul. 2015.

[115] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. Automat. Control*, vol. 54, no. 1, pp. 48–61, Jan. 2009.

[116] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.

[117] M. J. Neely and H. Yu, "Online convex optimization with time-varying constraints," *arXiv preprint:1702.04783*, Feb. 2017.

[118] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro, "Robust stochastic approximation approach to stochastic programming," *SIAM J. Optimization*, vol. 19, no. 4, pp. 1574–1609, 2009.

[119] Y. Nesterov, *Introductory Lectures on Convex Optimization: A basic course*. Berlin, Germany: Springer, 2013, vol. 87.

[120] Y. Nesterov and V. Spokoiny, "Random gradient-free minimization of convex functions," *Foundations of Computational Mathematics*, vol. 17, no. 2, pp. 527–566, Apr. 2017.

[121] D. S. Nunes, P. Zhang, and J. S. Silva, "A survey on human-in-the-loop applications towards an Internet of all," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 2, pp. 944–965, Second quarter 2015.

[122] S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian, "Deep decentralized multi-task multi-agent reinforcement learning under partial observability," in *Proc. Intl. Conf. Machine Learn.*, Sydney, Australia, Jun. 2017, pp. 2681–2690.

[123] M. Papini, D. Binaghi, G. Canonaco, M. Pirotta, and M. Restelli, "Stochastic variance-reduced policy gradient," in *Proc. Intl. Conf. Machine Learn.*, Stockholm, Sweden, Jul. 2018, pp. 4026–4035.

[124] M. Papini, M. Pirotta, and M. Restelli, "Adaptive batch size for safe policy gradients," in *Proc. Advances in Neural Info. Process. Syst.*, Long beach, CA, Dec. 2017, pp. 3591–3600.

[125] S. Paternain and A. Ribeiro, "Online learning of feasible strategies in unknown environments," *IEEE Trans. Autom. Contr.*, to appear, 2016. [Online]. Available: https://arxiv.org/pdf/1604.02137v1.pdf

[126] Z. Peng, Y. Xu, M. Yan, and W. Yin, "Arock: an algorithmic framework for asynchronous parallel coordinate updates," *SIAM J. Sci. Comp.*, vol. 38, no. 5, pp. 2851–2879, Sep. 2016.

[127] I. Pinelis, "Optimum bounds for the distributions of martingales in banach spaces," *The Annals of Probability*, vol. 22, no. 4, pp. 1679–1706, Oct. 1994.

[128] B. T. Polyak, *Introduction to Optimization*. New York, NY: Optimization Software, 1987.

[129] B. Recht, C. Re, S. Wright, and F. Niu, "Hogwild: A lock-free approach to parallelizing stochastic gradient descent," in *Proc. Advances in Neural Info. Process. Syst.*, Granada, Spain, Dec. 2011, pp. 693–701.

[130] A. Ribeiro, "Ergodic stochastic optimization algorithms for wireless communication and networking," *IEEE Trans. Signal Process.*, vol. 58, no. 12, pp. 6369–6386, Dec. 2010.

[131] H. Robbins and S. Monro, "A stochastic approximation method," *Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, Sep. 1951.

[132] N. L. Roux, M. Schmidt, and F. R. Bach, "A stochastic gradient method with an exponential convergence rate for finite training sets," in *Proc. Advances in Neural Information Processing Systems*, Lake Tahoe, NV, Dec. 2012, pp. 2663–2671.

[133] F. Samie, V. Tsoutsouras, L. Bauer, S. Xydis, D. Soudris, and J. Henkel, "Computation offloading and resource allocation for low-power IoT edge devices," in *Proc. World Forum Internet Things*, Dec. 2016, pp. 7–12.

[134] F. Samie, V. Tsoutsouras, S. Xydis, L. Bauer, D. Soudris, and J. Henkel, "Distributed QoS management for Internet of Things under resource constraints," in *Proc. Intl. Conf. on Hardware/Software Codesign and System Synthesis*, Pittsburgh, PA, Oct. 2016, pp. 1–10.

[135] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Trans. Signal Info. Process. Netw.*, vol. 1, no. 2, pp. 89–103, Jun. 2015.

[136] I. D. Schizas, A. Ribeiro, and G. B. Giannakis, "Consensus in ad hoc WSNs with noisy links – Part I: Distributed estimation of deterministic signals," *IEEE Trans. Sig. Proc.*, vol. 56, no. 1, pp. 350–364, Jan. 2008.

[137] J. Schneider, W.-K. Wong, A. Moore, and M. Riedmiller, "Distributed value functions," in *Proc. Intl. Conf. Machine Learn.*, Bled, Slovenia, Jun. 1999, pp. 371–378.

[138] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. Intl. Conf. Machine Learn.*, Lille, France, Jul. 2015, pp. 1889–1897.

[139] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint:1707.06347*, Jul. 2017.

[140] S. Shahrampour and A. Jadbabaie, "Distributed online optimization in dynamic environments using mirror descent," *arXiv preprint:1609.02845*, Sep. 2016.

[141] S. Shalev-Shwartz, "Online learning and online convex optimization," *Found. and Trends in Mach. Learn.*, vol. 4, no. 2, pp. 107–194, 2011.

[142] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe, multi-agent, reinforcement learning for autonomous driving," *arXiv preprint arXiv:1610.03295*, 2016.

[143] O. Shamir, "An optimal algorithm for bandit and zero-order convex optimization with two-point feedback," *Journal of Machine Learning Research*, vol. 18, no. 52, pp. 1–11, 2017.

[144] O. Shamir, N. Srebro, and T. Zhang, "Communication-efficient distributed optimization using an approximate newton-type method," in *Proc. Intl. Conf. Machine Learn.*, Beijing, China, Jun. 2014, pp. 1000–1008.

[145] A. Shapiro, D. Dentcheva, and A. Ruszczyński, *Lectures on Stochastic Programming: Modeling and Theory*. Philadelphia, PA: SIAM, 2009.

[146] Y. Shen, T. Chen, and G. B. Giannakis, "Online ensemble multi-kernel learning adaptive to non-stationary and adversarial environments," in *Proc. of Intl. Conf. on Artificial Intelligence and Statistics*, Lanzarote, Canary Islands, Apr. 2018.

[147] ——, "Random feature-based online multi-kernel learning in environments with unknown dynamics," *Journal of Machine Learning Research*, vol. 19, pp. 1–36, 2018.

[148] V. G. Sigillito, S. P. Wing, L. V. Hutton, and K. B. Baker, "Classification of radar returns from the ionosphere using neural networks," *Johns Hopkins APL Technical Digest*, vol. 10, no. 3, pp. 262–266, 1989.

[149] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. Intl. Conf. Machine Learn.*, Beijing, China, Jun. 2014.

[150] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Proc. Advances in Neural Info. Process. Syst.*, Long Beach, CA, Dec. 2017, pp. 4427–4437.

[151] J. A. Stankovic, "Research directions for the Internet of Things," *IEEE Internet of Things J.*, vol. 1, no. 1, pp. 3–9, Feb. 2014.

[152] S. U. Stich, "Local SGD converges fast and communicates little," *arXiv preprint:1805.09767*, May 2018.

[153] I. Stoica, D. Song, R. A. Popa, D. Patterson, M. W. Mahoney, R. Katz, A. D. Joseph, M. Jordan, J. M. Hellerstein, J. E. Gonzalez *et al.*, "A Berkeley view of systems challenges for AI," *arXiv preprint:1712.05855*, Dec. 2017.

[154] P. Stone and M. Veloso, "Multiagent systems: A survey from a machine learning perspective," *Autonomous Robots*, vol. 8, no. 3, pp. 345–383, Jun. 2000.

[155] S. Sun, M. Dong, and B. Liang, "Distributed real-time power balancing in renewable-integrated power grids with storage and flexible loads," *IEEE Trans. Smart Grid*, 2016, to appear.

[156] T. Sun, R. Hannah, and W. Yin, "Asynchronous coordinate descent under more realistic assumptions," in *Proc. Advances in Neural Info. Process. Syst.*, Long Beach, CA, Dec. 2017, pp. 6183–6191.

[157] A. T. Suresh, X. Y. Felix, S. Kumar, and H. B. McMahan, "Distributed mean estimation with limited communication," in *Proc. Intl. Conf. Machine Learn.*, Sydney, Australia, Aug. 2017, pp. 3329–3337.

[158] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction.* Cambridge, MA: MIT Press, 2018.

[159] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. Advances in Neural Info. Process. Syst.*, Denver, CO, Dec. 2000, pp. 1057–1063.

[160] J. Tadrous, A. Eryilmaz, and H. El Gamal, "Proactive resource allocation: Harnessing the diversity and multicast gains," *IEEE Trans. Inf. Theory*, vol. 59, no. 8, pp. 4833–4854, Aug. 2013.

[161] A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente, "Multiagent cooperation and competition with deep reinforcement learning," *PloS one*, vol. 12, no. 4, p. e0172395, 2017.

[162] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Autom. Contr.*, vol. 37, no. 12, pp. 1936–1948, Dec. 1992.

[163] R. Urgaonkar, B. Urgaonkar, M. Neely, and A. Sivasubramaniam, "Optimal power cost management using stored energy in data centers," in *Proc. ACM SIGMETRICS*, San Jose, CA, Jun. 2011, pp. 221–232.

[164] V. Vapnik, *The Nature of Statistical Learning Theory*. Berlin, Germany: Springer Science & Business Media, 2013.

[165] H.-T. Wai, Z. Yang, Z. Wang, and M. Hong, "Multi-agent reinforcement learning via double averaging primal-dual optimization," in *Proc. Advances in Neural Info. Process. Syst.*, Montreal, Canada, Dec. 2018.

[166] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, Feb. 2017, submitted. [Online]. Available: https://arxiv.org/abs/1702.00606

[167] ——, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 17, no. 3, pp. 1784–1797, Mar. 2018.

[168] H. Wang and A. Banerjee, "Online alternating direction method," in *Proc. Intl. Conf. on Machine Learning*, Edinburgh, Scotland, Jun. 2012.

[169] C. J. Watkins and P. Dayan, "Q-learning," *Machine Learn.*, vol. 8, no. 3-4, pp. 279–292, May 1992.

[170] E. Wei, A. Ozdaglar, and A. Jadbabaie, "A distributed Newton method for network utility maximization-I: Algorithm," *IEEE Trans. Autom. Contr.*, vol. 58, no. 9, pp. 2162–2175, Sep. 2013.

[171] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3-4, pp. 229–256, May 1992.

[172] D. H. Wolpert, K. R. Wheeler, and K. Tumer, "General principles of learning-based multi-agent systems," in *Proc. of the Annual Conf. on Autonomous Agents*, Seattle, WA, May 1999, pp. 77–83.

[173] L. Xiao, M. Johansson, and S. P. Boyd, "Simultaneous routing and resource allocation via dual decomposition," *IEEE Trans. Commun.*, vol. 52, no. 7, pp. 1136–1144, Jul. 2004.

[174] H. Yu and M. J. Neely, "A low complexity algorithm with $\mathcal{O}(\sqrt{T})$ regret and constraint violations for online convex optimization with long term constraints," *arXiv preprint:1604.02218*, Apr. 2016.

[175] K. Yuan, B. Ying, and A. H. Sayed, "On the influence of momentum acceleration on online learning," *arXiv preprint:1603.04136*, Mar. 2016.

[176] T. Zachariah, N. Klugman, B. Campbell, J. Adkins, N. Jackson, and P. Dutta, "The Internet of Things has a gateway problem," in *Proc. ACM HotMobile*, Santa Fe, NM, Feb. 2015, pp. 27–32.

[177] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for smart cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, 2014.

[178] M. Zargham, A. Ribeiro, and A. Jadbabaie, "Accelerated backpressure algorithm," *arXiv preprint:1302.1475*, Feb. 2013.

[179] K. Zhang, Z. Yang, and T. Başar, "Networked multi-agent reinforcement learning in continuous spaces," in *Proc. IEEE Conf. Decision and Control*, Miami, FL, Dec. 2018, pp. 5872–5881.

[180] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Başar, "Fully decentralized multi-agent reinforcement learning with networked agents," in *Proc. Intl. Conf. Machine Learn.*, Stockholm, Sweden, Jul. 2018, pp. 5872–5881.

[181] Y. Zhang, S.-J. Kim, and G. B. Giannakis, "Short-term wind power forecasting using nonnegative sparse coding," in *Proc. IEEE Conf. on Info. Sci. and Syst.*, Baltimore, MD, Mar. 2015.

[182] Y. Zhang, J. C. Duchi, and M. J. Wainwright, "Communication-efficient algorithms for statistical optimization." *J. Machine Learning Res.*, vol. 14, no. 11, 2013.

[183] Y. Zhang and X. Lin, "DiSCO: Distributed optimization for self-concordant empirical loss," in *Proc. Intl. Conf. Machine Learn.*, Lille, France, Jun. 2015, pp. 362–370.

[184] Q. Zhu, R. Wang, Q. Chen, Y. Liu, and W. Qin, "IoT gateway: Bridging wireless sensor networks into Internet of Things," in *Proc. Intl. Conf. Embedded Ubiquitous Comp.*, Hong Kong, China, Dec. 2010, pp. 347–352.

[185] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *Proc. Intl. Conf. on Machine Learning*, Washington D.C., Aug. 2003.