Online learning at the edge

Paul Youssef

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

École doctorale : MSTII - Mathématiques, Sciences et technologies de l'information, Informatique
Spécialité : Informatique
Unité de recherche : Laboratoire d'Informatique de Grenoble

Apprentissage en ligne pour le edge computing

Online learning at the edge

Présentée par :

Paul YOUSSEF

Direction de thèse :

Denis TRYSTRAM Professeur des Universités, Grenoble INP	Directeur de thèse
Kim Thang NGUYEN Professeur des Universités, Grenoble INP	Co-directeur de thèse

Rapporteurs :

OLIVIER BEAUMONT
Directeur de recherche, INRIA CENTRE BORDEAUX SUD-OUEST
GIOVANNI NEGLIA
Directeur de recherche, INRIA -SOPHIA ANTIPOLIS-MEDITERRANEE

Thèse soutenue publiquement le **13 mars 2023**, devant le jury composé de :

DENIS TRYSTRAM Professeur des Universités, GRENOBLE INP	Directeur de thèse
OLIVIER BEAUMONT Directeur de recherche, INRIA CENTRE BORDEAUX SUD-OUEST	Rapporteur
GIOVANNI NEGLIA Directeur de recherche, INRIA -SOPHIA ANTIPOLIS-MEDITERRANEE	Rapporteur
FLORENCE FORBES Directeur de recherche, INRIA CENTRE GRENOBLE-RHONE-ALPES	Présidente
FRANCIS BACH Directeur de recherche, INRIA CENTRE DE PARIS	Examinateur
DIDIER DONSEZ Professeur des Universités, UNIVERSITE GRENOBLE ALPES	Examinateur

Invités :

KIM THANG NGYEN
Professeur des Universités, GRENOBLE INP

Online Learning At The Edge

Paul Youssef

June 9, 2023

University Grenoble Alpes

Laboratoire d'Informatique de Grenoble (LIG)
Multidisiplinary Institut of Artificial Intelligence (MIAI)
DATAMOVE team

Decentralized Online Learning

Online Learning At The Edge

Paul Youssef

1. Reviewer	Olivier Beaumont Bordeaux 1 University LaBRI - Laboratoire Bordelais de Recherche en Informatique
2. Reviewer	Giovanni Neglia Côte d'Azur University INRIA - National Institute for Research in Digital Science and Technology
Supervisors	Nguyen Kim Thang and Denis Trystram

June 9, 2023

Paul Youssef
Online Learning At The Edge
Decentralized Online Learning, June 9, 2023
Reviewers: Olivier Beaumont and Giovanni Neglia
Supervisors: Nguyen Kim Thang and Denis Trystram

University Grenoble Alpes
DATAMOVE team
Multidisiplinary Institut of Artificial Intelligence (MIAI)
Laboratoire d'Informatique de Grenoble (LIG)
700 Av. Centrale
38400 and Saint-Martin-d'Hères

Supported by the Multidisciplinary Institute in Artificial Intelligence, Univ.Grenoble Alpes, France (ANR-19-P3IA-0003).

Abstract

Due to the proliferation of IoT devices, the volume of data produced is becoming challenging to transfer, store, and process in remote centralized architectures. Hence, machine learning at the edge has recently attracted considerable interest. The goal is to process the data as close to its source. We typically have multiple agents collaborating via peer-to-peer communication to optimize an aggregate of loss functions with components spread over a connected network. Many decentralized optimization algorithms, both projection and projection-free algorithms with theoretical guarantees have been proposed in the literature, focusing mainly on offline settings. However, for most real-world machine learning problems, the data is often revealed online, for example, in the case of recommender systems. Therefore, we study decentralized optimization within online settings.

We propose online decentralized Frank-Wolfe algorithms that use stochastic gradient estimates, which achieve an asymptotically tight regret guarantee of $\mathcal{O}\left(\sqrt{T}\right)$ where T is a given time horizon. The algorithms differ in their communication protocol, one has synchronized communication and the second has a random walk approach. Furthermore, we compare these algorithms' performance for optimizing the online multiclass logistic regression model on real-world standard image datasets (MNIST, CIFAR10) with centralized online algorithms and the previously best-known decentralized online algorithms. We show proof of concept by deploying the above experiment on a cluster of devices with limited resources, namely Raspberry Pi. Finally, by optimizing a long short-term memory neural network model, we discuss their usefulness in typical edge applications: temperature forecasting in smart buildings.

Abstract (french)

En raison de la prolifération des dispositifs IoT, le volume de données produites devient difficile à transférer, à stocker et à traiter dans des architectures centralisées à distance. Par conséquent, l'apprentissage automatique dans le Edge a récemment suscité un intérêt considérable. L'objectif est de traiter les données au plus près de leur source. En général, plusieurs agents collaborent via une communication de pair à pair pour optimiser un agrégat de fonctions de perte dont les composants sont répartis sur un réseau connecté. De nombreux algorithmes d'optimisation décentralisés, à la fois des algorithmes avec et sans projection avec des garanties théoriques, ont été proposés dans la littérature, en se concentrant principalement sur des approches offline. Cependant, pour la plupart des problèmes d'apprentissage automatique du monde réel, les données sont souvent révélées de manière online, par exemple, dans le cas des systèmes de recommandation. Par conséquent, nous étudions l'optimisation décentralisée dans des contextes online. Nous proposons des algorithmes de Frank-Wolfe décentralisés en ligne qui utilisent des estimations de gradient stochastiques et qui garantissent un regret asymptotiquement de $\mathcal{O}\left(\sqrt{T}\right)$, où T est un horizon temporel donné.

Acknowledgement

First and foremost, I extend my deepest gratitude to my co-supervisors, Denis Trystram and Nguyen Kim Thang. Their guidance, mentorship, and unwavering support have been instrumental in my personal and academic growth.

Denis Trystram, your consistent support and teachings have shaped me into not just a better researcher, but a more conscientious and understanding individual. Your insights into the ethical issues surrounding our ecological situation have enriched my worldview and instilled a greater sense of responsibility within me. I offer my heartfelt thanks for your exceptional teachings and dedication.

Nguyen Kim Thang, your exceptional availability and willingness to explain complex technical matters, despite a packed schedule, is a testament to your generosity and profound expertise. Your youthful mindset and effective teaching methods have made the learning process both insightful and engaging. I am deeply honored to have been under your guidance and highly appreciate your unwavering support.

I wish to express my sincere thanks to the esteemed jury members and reviewers: Giovanni Neglia, Olivier Beaumont, Francis Bach, Florence Forbes, and Didier Donsez. Your insights, meticulous feedback, and diligent contributions have significantly enriched this thesis.

Special recognition goes to Giovanni Neglia and Olivier Beaumont, who also served as reviewers. Your critical assessment and constructive suggestions have significantly contributed to refining and enhancing this work.

Giovanni Neglia, in particular, deserves special mention for his exceptional contributions in providing constructive feedback that greatly enhanced the understanding of my manuscript. His insightful suggestions and attention to detail were invaluable in improving the clarity and coherence of my work. Additionally, Giovanni's meticulous proofreading ensured the accuracy and correctness of the final document. I am deeply grateful for his expertise and dedication, which significantly contributed to refining and enhancing this thesis.

I am grateful for the funding provided by the Multidisciplinary Institute of Artificial Intelligence (MIAI) for this research. Their support has been essential in advancing my studies and contributing to the field of artificial intelligence. I appreciate their belief in my potential and their investment in my academic journey.

My collaborators, Abhinav Srivastav, Tuan Ahn Nguyen, and Angan Mitra, deserve a special mention.

Tuan Ahn Nguyen, your consistent support, generosity, and light-hearted demeanor have been truly admirable. Your readiness to assist in all circumstances made you a much-valued collaborator and friend. Your jovial nature and easy-going personality always brought an aura of warmth and reassurance, transforming any work-related stress into moments of joy and learning. You made the journey significantly more enjoyable and nurturing, and for this, I am profoundly grateful.

Abhinav Srivastav, your nurturing guidance and altruistic nature have been a source of comfort and encouragement throughout this journey. Although we haven't had the opportunity to meet face-to-face, I look forward to doing so soon.

Angan Mitra, your energetic personality and ability to bring joy into any situation have provided much-needed relief during our most challenging times. I extend my wishes for your success and fulfillment in pursuing your wild aspirations.

Enikő Kevi, you have been a beacon of support, both academically and personally. Your unwavering willingness to assist in any way possible, as well as your companionship, have left a lasting impact on my life. Sharing an office, a desk, and countless memorable moments with you, transformed our professional association into a cherished friendship. You've enriched my journey in more ways than words can express. As you embark on your own path, I hope all the best for you and am confident that you will illuminate any endeavor you undertake with your brilliance.

I would like to extend my heartfelt thanks to Etienne Dublé for his indispensable assistance with the WalT platform. Etienne's guidance, expertise, and provision of Raspberry Pi devices were instrumental in the success of my research. He generously dedicated his time and knowledge to help me overcome networking issues, ensuring the smooth operation of the platform. I am truly grateful for his unwavering support and his significant contributions to my work.

I thoroughly enjoyed working with David Emukpere and Jizong Zhan during their time as interns. Their dedication, enthusiasm, and willingness to learn were evident in their work. They both

provided assistance that was instrumental to the success of my project. I am grateful for their efforts and the positive energy they brought. I sincerely hope you achieve the careers you aspire to and find fulfillment and success in your future endeavors.

I would like to express my deep appreciation for Annie Simon, whose unique ability to resolve administrative problems has been priceless. Our philosophical discussions have added depth and engagement to our interactions.

I express my heartfelt gratitude to my colleagues at Datamove and the Polaris team for cultivating an engaging and encouraging atmosphere that has significantly enhanced my experience throughout my doctoral pursuit. Additionally, I am grateful for the camaraderie that developed with my office mates, who have become cherished friends along this journey.

To the coinche players, particularly Vincent Fagnon, thank you for the delightful games that have provided much-needed relaxation amidst research rigor.

I would also like to thank Tuan Ahn Nguyen, Enikő Kevi, Mathilde Jay, and all others who helped organize my defense and the ensuing celebration (pot). I am also immensely grateful to all those who participated in my farewell gift. Your thoughtfulness truly touched me, and I feel fortunate to have had the opportunity to meet each one of you. I sincerely wish you all the best in your future endeavors.

I extend my sincere thanks to my friends and family who have stood by me through this journey. My brother, Pierre, deserves special mention for traveling to Grenoble to help with the organization of my thesis celebration. His support and assistance during this crucial time have been invaluable.

To my parents, Joseph Youssef and Mona Chedid, and to the entirety of my family - your unconditional love, understanding, and patience throughout this arduous journey have been my pillars of strength. Each one of you, in your unique way, has contributed to the person I am today and the achievement we are celebrating. You have instilled in me a love for learning and an unwavering faith in my abilities. This achievement is as much yours as it is mine.

Despite my sincere desire to express gratitude to all those who have supported me, the extensive list of individuals deserving recognition necessitates a difficult decision. Countless names come to mind, each deserving of heartfelt appreciation. However, in the interest of brevity and the limits of this acknowledgment section, I must reluctantly bring this list to a close. Please know that even if your name isn't mentioned explicitly, your contributions have not gone unnoticed,

and I am truly grateful for the impact you have made on my life and this thesis. After all, if I continue, this thesis might turn into an encyclopedic masterpiece of acknowledgments!

Contents

1 Introduction

1.1 Edge computing paradigm

In many modern applications, devices rely on data processing centers to provide services beyond their technical capabilities. Typically, data is collected near the generating devices, such as IoT sensors, but due to the low computation and storage capacity of these devices, the data must be transferred over a network to be processed in a remote data processing center. The results of these computations are then transmitted back through the network to the original devices. However, this paradigm has several significant drawbacks. Firstly, it incurs high communication costs and energy usage. For example, data center traffic accounted for over three quarters of global internet traffic, with a significant portion of this traffic being attributed to data transfers between data centers and devices. Additionally, data transfers can require a large amount of energy, with estimates ranging from hundreds of kilojoules per gigabyte for short-range transfers to thousands of kilojoules per gigabyte for longer-range transfers. Secondly, this paradigm is limited in its ability to handle real-time data, as frequent data transfers can easily overwhelm the system, leading to delays and increased latency. Finally, IoT devices become useless if they lose their connection to the data processing centers, which can be a major issue for applications that rely on real-time data or require continuous operation.

To address these issues, the distributed paradigm of *edge computing* has gained significant attention in recent years. Edge computing aims to bring computation as close as possible to the location where data is generated, thus eliminating the aforementioned drawbacks. Edge applications rely on dispersed data on *edge devices*, which makes data processing more secure and resilient. One of the main challenges of edge computing is the limited *storage capacity* and *computing power* of edge devices. For example, many IoT devices have storage capacities in the range of tens to hundreds of megabytes and processing capabilities limited to simple operations. To enable the implementation of edge applications, it is important to follow the following policies:

- **Design low complex algorithms:** Edge devices have limited *storage capacity* and *computing power*. Using algorithms with high complexity can quickly exhaust the available resources, leading to poor performance or even failure.

- **Limit communications:** *Communications* should be minimized in size, distance, and number. Reducing the amount of data transmitted and the distance it must travel can help to minimize energy usage and improve real-time performance.
- **Ensure privacy by design:** Edge devices should retain their *personal data* (i.e. edge devices should not send their personal data to any other entity). Ensuring that personal data remains on the edge device can help to protect privacy and reduce the risk of data breaches.

In the following sections, we will introduce various machine learning settings and algorithms to find the best fitting settings for the edge computing paradigm.

1.2 Classical machine learning

1.2.1 Settings

Based on the assumption that successful inferences from the past are likely to stay true in the future, *machine learning algorithms* build a *model* using samples, known as *training data*. The models capture the general patterns within the training data, exploiting the past to suggest *predictions* or *decisions*. Machine learning problem statements are usually described as *optimization* problems. Hence, the field of machine learning has a large intersection with the field of *optimization*. You can read about the history of numerical optimization in the context of machine learning in [BCN18]. A simple example is fitting the model's *parameters* using a maximum likelihood estimator. Given a set of data points, one needs to estimate the model's parameters through risk minimization [Vap98].

Formally, a *model* $\mathcal{M}_{\boldsymbol{x}}$ is a multidimensional function with *parameters* $\boldsymbol{x}$. The unknown future that we want to predict is an oracle function Φ. Given an input e, the output of $\mathcal{M}_{\boldsymbol{x}}(e)$ should be as close as possible to the ground truth $\Phi(e)$. We measure the closeness of the model's outputs to the ground truth with a *criteria function* δ. The *training data* $\mathcal{B}_{train}$ consists of past events, therefore we know the outputs of $\Phi(e)$ for all $e \in \mathcal{B}_{train}$. During the training process, we want to set the model's parameters to closely estimate Φ. To measure the model's closeness, we define a *loss/cost function* f dependant on the training data and the criteria function δ as follows

$$f(\boldsymbol{x}) = \frac{1}{|\mathcal{B}_{train}|} \sum_{e \in \mathcal{B}_{train}} \delta\left(\mathcal{M}_{\boldsymbol{x}}(e), \Phi(e)\right). \tag{1.1}$$

Adding constraints on a model's parameters can be an effective method for improving the model's performance and prediction accuracy in machine learning. These constraints may arise naturally from the problem being solved, or they may be chosen to promote certain desirable properties in the model. Selecting and applying constraints carefully leads to the creation of more effective and reliable machine learning models.

In other words, the goal of the learning algorithm is to find the best parameters $\boldsymbol{x}^*$ within a constraint set $\mathcal{K}$, by *minimizing* the cost function f. This is expressed mathematically as:

$$\boldsymbol{x}^* \in \arg\min_{x \in \mathcal{K}} f(\boldsymbol{x}). \tag{1.2}$$

Additionally, it is important to note that the choice of criteria function δ and cost function f can significantly impact the performance of the machine learning model. These functions should be carefully chosen based on the specific goals and characteristics of the problem. For example, if the goal is to minimize the average error between the model's output and the ground truth, the mean squared error function could be used as the costfunction. On the other hand, if the goal is to maximize the probability of correct predictions, the negative log likelihood function could be used as the cost function.

It is also important to thoroughly test the trained model on a separate set of *testing data* to ensure that it is accurate enough for deployment. This helps to prevent overfitting, or the tendency of the model to perform well on the training data but poorly on new data.

In conclusion, classical machine learning involves choosing and optimizing a model to make predictions or decisions based on past data, using a variety of algorithms and carefully chosen criteria and cost functions. Proper testing and evaluation of the trained model is crucial for ensuring its accuracy and effectiveness.

1.2.2 Offline learning algorithms

This section introduces two pillars of learning algorithms :

1. **Gradient Descent (GD):** Gradient descent is the most popular algorithm in the context of machine learning. Augustin-Louis Cauchy first conceived the algorithm in 1847 to solve *unconstrained* optimization problems. Intuitively, GD iteratively updates the model's parameters according to the objective cost function's gradient, following the slope of the

objective function f. Formally, let $\boldsymbol{x}_\ell$ be the parameters at iteration ℓ, GD updates $\boldsymbol{x}_\ell$ as follows,

$$\boldsymbol{x}_{\ell+1} \leftarrow \boldsymbol{x}_\ell - \eta_\ell \nabla f(\boldsymbol{x}_\ell) \tag{1.3}$$

where $\nabla f(\boldsymbol{x}_\ell)$ is the *gradient* of f at $\boldsymbol{x}_\ell$ and η_ℓ is the *step size* of the update.

Algorithms must ensure the feasibility of all iterates $\boldsymbol{x}_\ell$ when we consider constrained optimization. A natural extension of GD is the *Projected Gradient Descent* algorithm (PGD) in which each *iterate* is *projected* onto the feasible set $\mathcal{K}$ using the *projection* denoted by $\mathcal{P}_\mathcal{K}$ as follows :

$$\boldsymbol{x}_{\ell+1} \leftarrow \mathcal{P}_\mathcal{K}\left[\boldsymbol{x}_\ell - \eta_\ell \nabla f(\boldsymbol{x}_\ell)\right] \tag{1.4}$$

Note that the choice of projection $\mathcal{P}_\mathcal{K}$ is crucial to the performance of the algorithm both in terms of computation and quality of convergence.

Recall that computing the gradient ∇f of the function f requires the execution of a sum scaled by the size of the training set $\mathcal{B}_{train}$ (see equation 1.1). However, in large-scale machine learning problems in which $\mathcal{B}_{train}$ is very large, evaluating the gradient ∇f becomes expensive. To improve the computational cost of each iteration, an extension to GD called Stochastic Gradient Descent (SGD) was proposed by [RM51]. Instead of naively computing the gradient, SGD samples a subset $\widetilde{\mathcal{B}}$ of $\mathcal{B}_{train}$ at each iteration, then computes a *stochastic gradient estimate* $\widetilde{\nabla} f$ with the sample $\widetilde{\mathcal{B}}$. Many other variants have been proposed and analyzed with varying contexts (see overview [BCN18]).

2. **Frank-Wolfe (FW):** The Frank-Wolfe algorithm, also known as *conditional gradient method*, is one of the simplest and earliest proposed algorithms to tackle constrained optimization problems, first introduced in [FW56] and more recently revisited by [Jag13]. Like the Projected-SGD algorithm, FW is an approximation iterative process as well. Given a current iterate $\boldsymbol{x}_\ell$, the algorithm considers the *linearization* of the objective function f (equation 1.5), then updates the iterates following the direction of a *minimizer* $\boldsymbol{v}_\ell$ of that linear function using a *convex combination* (equation 1.6):

$$\boldsymbol{v}_\ell \leftarrow \arg\min_{\boldsymbol{v}\in\mathcal{K}} \langle \boldsymbol{v}, \nabla f(\boldsymbol{x}_\ell)\rangle \tag{1.5}$$

$$\boldsymbol{x}_{\ell+1} \leftarrow (1-\eta_\ell)\boldsymbol{x}_\ell + \eta_\ell \boldsymbol{v}_\ell \tag{1.6}$$

Its success comes from its good scalability and its property to maintain its iterates $\boldsymbol{x}_k$ in the feasible set $\mathcal{K}$ by using *linear optimizers* and convex combinations instead of projections. (see [Cla10; Jag11; Bra+22] for an overview).

1.3 Online machine learning

Context.
In the offline setting, we assume that the chosen model with the training and testing data is a reliable estimate of the ground truth for the training and test procedure to be practically efficient. This assumption is based on the idea that the training data is representative of the underlying distribution of the data, and that the model will generalize well to new, unseen data. However, this assumption may not always hold true in practice, especially when the data distribution is constantly changing or when the data is generated by an adversary.

For instance, people are generally consistent with their handwriting over a long period of time, and thus collecting a sufficient number of diverse handwritten digit images in a large enough sample can yield training data in which offline learning approaches have proven effective in recognizing handwritten digits. However, other learning applications may be more susceptible to changes in the data distribution.

One example is spam filtering, where the learner's objective is to classify emails as spam or valid. Because spam emails are often designed to deceive, the data is generated by an adversary who may attempt to disrupt the learner's decision-making. In this case, the offline learning strategy may not be able to maintain its predictive accuracy, as the data distribution is constantly changing and the model is not able to adapt to these changes. In such scenarios, *online machine learning* may be more suitable, as it allows the model to continuously update itself based on new data and adapt to the uncertainty of the future. This is because online learning allows the model to continuously learn and adapt to changes in the data distribution, allowing it to maintain its predictive accuracy over time.

1.3.1 Online settings

In online learning, one needs to design an algorithm that trains a model under the uncertainty of the future. It can be seen as a process in which a learner tries to make decisions in an evolving environment. Analogously, the learner plays the role of the online learning algorithm that iteratively updates the learning model. After each model update, the environment generates data samples that challenge the model's accuracy. The *online decision making* proceeds in repeated rounds and is described as follows.

Adversarial online settings.
At each *round* t, the learner has to make an *irrevocable* decision $\boldsymbol{x}^t$ from a given set $\mathcal{K}$ while unaware of its corresponding outcome. Once the learner is committed to his decision, an

adversary (the environment) generates a batch of data $\mathcal{B}^t$ which reveals a cost function f^t, causing the decision maker to suffer the loss $f^t(\boldsymbol{x}^t)$. We must restrict the choice of loss functions. The losses chosen by the adversary must lie in some bounded region. Otherwise, the adversary could scale the function to induce losses that are not recoverable for the learner. Moreover, the learner's decisions must lie in a bounded and structured set $\mathcal{K}$.

Regret with hindsight.

Intuitively, an online optimizer aims to improve its decision-making by learning from its past performance. This translates into a notion of *regret* that empirically measure the learner's performance. In particular, we consider the *regret with hindsight*, i.e. the difference between the total cost that the learner has accumulated and that of the best-fixed decision in hindsight. Formally, given $\{\boldsymbol{x}^t\}_{t\in 1,\dots,T}$ the learner's sequence of T decisions, and $\{f^t\}_{t\in 1,\dots,T}$ the cost functions chosen by the adversary, we define the regret with hindsight of the decision maker as:

$$\mathcal{R}^T = \sum_{t=1}^{T} f^t(\boldsymbol{x}^t) - \min_{\boldsymbol{x}^*\in\mathcal{K}} \sum_{t=1}^{T} f^t(\boldsymbol{x}^*) \tag{1.7}$$

We say an online learning algorithm performs well when its regret is sublinear as a function of T, i.e. $\mathcal{R}^T \in o(T)$.

1.3.2 Online algorithms

The online decision-making process has emerged as an attractive paradigm in machine learning for tackling an extensive array of optimization problems to maintain robust solutions under the uncertainty of the future. This captures many real-world scenarios, including machine learning. Many machine learning papers studied online optimization (see [Haz16] as a starting point). In this section, we will introduce the two most famous algorithms :

1. **Online Gradient Descent (OGD):** Online gradient descent is a straightforward extension of gradient descent from section 1.2.2 to the general online settings. The algorithm was first introduced and analyzed in [Zin03]. Intuitively, the online gradient descent algorithm updates its next decision by following the slope of the previously observed cost function. Similar to the projected gradient descent, if this decision lies outside the feasible set, the algorithm projects it back. Formally, given the decision $\boldsymbol{x}^t$ at time t and its cost $f^t(\boldsymbol{x}^t)$, OGD computes its next decision $\boldsymbol{x}^{t+1}$ with the following :

$$\boldsymbol{x}^{t+1} \leftarrow \mathcal{P}_{\mathcal{K}}\left[\boldsymbol{x}^t - \eta_t \nabla f^t(\boldsymbol{x}^t)\right] \tag{1.8}$$

   This simple approach yields sublinear regret. More precisely, the regret is bounded by $\mathcal{O}\left(\sqrt{T}\right)$ where T is the time horizon.

2. **Online Frank Wolfe :** Online Frank Wolfe is inspired by the Frank Wolfe algorithm from the offline settings described in 1.2.2. However, unlike the extension of gradient descent to the online settings, a straightforward extension of the Frank Wolfe algorithm is impossible. The online version of FW was proposed by [HK12]. At each step t, the idea is to apply the FW step described by equation 1.5 while replacing the function f with a regularized aggregate sum of all previous cost functions as follows :

$$\boldsymbol{v}^{t+1} \leftarrow \arg\min_{\boldsymbol{v}\in\mathcal{K}} \langle \boldsymbol{v}, \sum_{\tau=1}^{t} \nabla f^{\tau}(\boldsymbol{x}^t) + \nabla\psi(\boldsymbol{x}^t)\rangle \tag{1.9}$$

where ψ is a regularization function.

1.4 Decentralized machine learning

Context.

Decentralized machine learning focuses on designing and analyzing optimization algorithms that utilize local computation and communication among interconnected computing nodes without central coordination. It plays a vital role in a wide variety of applications that arise in the domain of machine learning [DMP16; WAP18; TLR12], control theory [LV10; Cao+13], signal processing [Rei+19] and operations research [EB12]. The main motivation behind distributed computing is when an optimization algorithm running on a single machine does not meet the required performance. One solution is using multiple machines, decomposing the problem, and running a decentralized algorithm. This approach is often the only choice for solving extremely large optimization problems. A simple example in machine learning is the model fitting using a maximum likelihood estimator. Given a set of data points, one needs to estimate the model's parameters through risk minimization [Vap98]. Here, an average of convex local loss functions defines the global objective, such as square loss or logistic loss associated with each data point in the set. Due to the large dataset volume, these optimization tasks cannot perform on a single computing node. Instead, one must choose decentralized solutions that efficiently exploit computational resources distributed in a connected network. Additionally, local computations should be light to perform on a single node. This differs from centralized distributed optimization, where the information from different nodes needs to be sent to a central processing unit.

1.4.1 Decentralized optimization settings

In decentralized settings, we consider n learners, also known as agents. The learners are connected over a network represented by a *graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is a finite set of learners and $\mathcal{E}$ is the set of edges. Each agent $i \in \mathcal{V}$ is given a machine learning model $\mathcal{M}$ and its *personal*

batch of dataset $\mathcal{B}_i$. However, instead of each agent optimizing its model on its local data, they all share a common objective which is to collaboratively optimize their models as if the union of all batches $\mathcal{B} = \bigcup_{i \in \mathcal{V}} \mathcal{B}_i$ is stored in one place. Formally, each agent i have only access to a local function f_i defined using its local dataset $\mathcal{B}_i$ as follows.

$$f_i(x) = \sum_{e \in \mathcal{B}_i} \delta(\mathcal{M}_{\boldsymbol{x}}(e), \Phi(e)) \tag{1.10}$$

Hence, when all the batches are stored in one place, we would consider F defined by the union of the batches $\{\mathcal{B}_i\}_{i \in \mathcal{V}}$ where F is normalized by the number of agents n as follows.

$$F(\boldsymbol{x}) = \frac{1}{n} \sum_{e \in \mathcal{B}} \delta(\mathcal{M}_{\boldsymbol{x}}, \Phi(e)) = \frac{1}{n} \sum_{i \in \mathcal{V}} f_i(\boldsymbol{x}) \tag{1.11}$$

It is natural to restrict agents' ability to share sensitive information with others, such as personal data. Exchanges, on the other hand, are necessary for the optimization procedure. For example, model updates and function gradients can be exchanged. It should be noted that this does not ensure complete privacy.

Decentralized objectives.

- **Optimize F :** In decentralized optimization, agents independently update their local model with local communication and their local personal data. However, they have the same objective which is to approach the best model parameter $\boldsymbol{x}_i^*$ that minimizes the objective function F defined in equation 1.11. Note that the objective function F is the same for all the agents.

$$\boldsymbol{x}_i^* \in \arg\min_{\boldsymbol{x} \in \mathcal{K}} F(\boldsymbol{x}) \tag{1.12}$$

- **Minimize communications :** Decentralized training is more efficient than centralized one when operating on networks with low bandwidth, or high latency [Lia+17; HBJ18]. Therefore, it is crucial when designing a decentralized algorithm to minimize the number and the range of its communications. One natural restriction is only to exchange information with closeness criteria; for instance, one can restrict the range by only allowing exchanges between direct neighbors.

1.5 Decentralized online machine learning at the edge

Recently several studies have embedded machine learning algorithms into low-complex edge computing devices. Such computing components exchange the parameters of their local models, and they update their models locally [Che+17; Jin+20]. Training at the edge is also along the line of *federated learning* [McM+21; Li+20]. In the latter, offline centralized training, a star network where a central server is connected to several devices, is the currently dominant paradigm.

The goal of this thesis is to design robust *online learning* algorithms directly on edge devices using a collaborative decentralized approach, with an emphasis on data and communication proximity and edge devices' limited resources. We intend to develop optimizers for a decentralized online learning model. More precisely, we investigate issues at the crossroads of two rapidly expanding fields, namely, *decentralized optimization* and *online learning*.

It is crucial to choose the right machine learning settings to effectively fit the edge computing paradigm in order to achieve the desired results :

- **Edge devices are often connected over a network with high latency and low bandwidth**
  ⇒ With such constraints, **decentralized** (section 1.4) training is more favorable than centralized ones. (see [Lia+17; HBJ18])
  ⇒ It is natural to restrict the devices to **communicate locally** with the smallest range possible.
- **Edge devices have limited resources**
  ⇒ **Online learning** (section 1.3) and **decentralized learning** (section 1.4) are both reliable approaches that scale the computation with limited storage capacity and computing power.
- **Edge applications often deal with dynamic environments**
  ⇒ Dynamic evolving environments heavily motivates the **online learning** settings.
- **Privacy preservation is desirable in edge applications**
  ⇒ **Decentralized learning** settings do not allow agents to communicate their raw data.

These settings, including decentralized and online learning, are designed to address the unique constraints and requirements of edge devices, such as limited resources, high latency and low bandwidth networks, dynamic environments, and the need for privacy preservation. By carefully selecting the appropriate machine learning settings, edge computing systems can be optimized for performance, efficiency, and effectiveness.

1.6 Problem statement and preliminaries

This section formalizes the problem statement while introducing useful definitions, notations, and assumptions.

1.6.1 Distributed environment

We are given a set of n computing units $\mathcal{V} = \{1, \dots, n\} = [n]$, referred as *agents* (vertices/edge devices) connected over a network represented by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. We assume that the graph $\mathcal{G}$ is *connected* (i.e. for any $i, j \in \mathcal{V}^2$, there exists a path between i and j) and *undirected* (i.e. if $(i, j) \in \mathcal{E}$ then $(j, i) \in \mathcal{E}$).

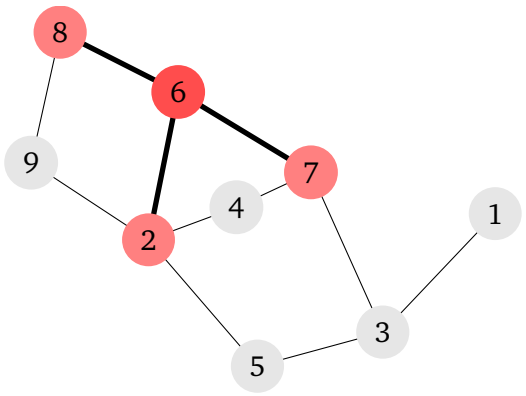


Fig. 1.1: In the graph $\mathcal{G}$ with 9 agents, the agent 6 can only send messages to its neighbors ($\mathcal{N}(6) = \{2, 7, 8\}$) colored in red.

Agents can **only communicate** to their **neighbors**, as illustrated in figure 1.1.
We say that j is a *neighbor* of i when $(i, j) \in \mathcal{E}$. We denote $\mathcal{N}(i)$ the set of neighbors of node i in the graph where $i \notin \mathcal{N}(i)$, and the degree of a node i is $\deg(i) = |\mathcal{N}(i)|$.

1.6.2 Online decision-making in distributed environment

The online decision-making process is divided into T *rounds*, where T is the *time horizon*. See figure 1.2 that illustrates this T-round process for each agent.

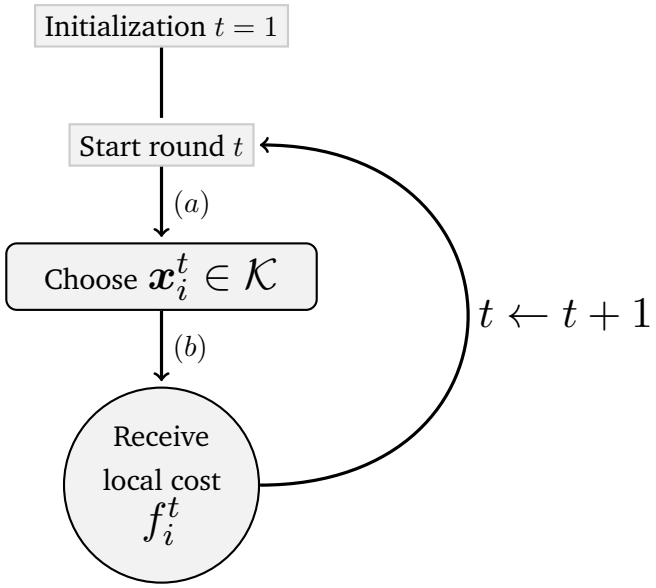


Fig. 1.2: Diagram describing an online decision-making process for each agent $i \in \mathcal{V}$.

Details on the online decision-making process (Figure 1.2).

A model iterate of size d is described by a vector $\boldsymbol{x} \in \mathbb{R}^d$ in dimension d. We use boldface letters (e.g. $\boldsymbol{x}$) to represent vectors. We denote $\boldsymbol{x}_i^t$ as the decision vector of agent i at time step t. We are given a set $\mathcal{K} \subseteq \mathbb{R}^d$ that represents the *constraint set* of the optimization. We assume that the set $\mathcal{K}$ is

- **convex :** A set $\mathcal{K} \subseteq \mathbb{R}^d$ is *convex* if, for all $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{K}^2$, the *line segment* between $\boldsymbol{x}$ and $\boldsymbol{y}$ is in $\mathcal{K}$. In other words, let $z = \alpha\boldsymbol{x} + (1-\alpha)\boldsymbol{y}$ where $\alpha \in [0,1]$, if $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{K}^2$ then $\boldsymbol{z} \in \mathcal{K}$.

and

- **compact :** The constraint set $\mathcal{K}$ is a bounded and closed convex set. We use the Euclidean norm $\|\cdot\|$ and the diameter of the convex domain $\mathcal{K}$ is defined as $D = \sup_{\boldsymbol{x},\boldsymbol{y}\in\mathcal{K}} \|\boldsymbol{x} - \boldsymbol{y}\|$.

[(a) in Figure 1.2] At the start of round t, the agent i already observed $f_i^1, \ldots, f_i^{t-1}$. With the help of past observations and by communicating with its neighbors, the agent i computes a decision $\boldsymbol{x}_i^t$.

[(b) in Figure 1.2] Once the agent i *commits to its decision*, a **private convex function** $f_i^t : \mathcal{K} \to \mathbb{R}$ is revealed *exclusively* to agent i. Consequently, the agent can evaluate its decision's local cost $f_i^t(\boldsymbol{x}_i^t)$. He can also compute the corresponding gradient $\nabla f_i^t(\cdot)$, and in the case of stochastic gradient estimates, we denote it as $\tilde{\nabla} f_i^t(\cdot)$.

The global objective of agents *vs* their partial knowledge.

Although each agent i observes only function f_i^t, all the agents are interested in optimizing the

same objective function at time t, which is the normalized aggregated sum of all local functions defined as $F^t(\cdot) = \frac{1}{n}\sum_{i=1}^{n} f_i^t(\cdot)$. Typically, f_i^t are functions defined on an agent's local batch of data received at time t, and F^t is the function defined on the union of all the batches. It is important to note that each agent can only observe its own local function and does not fully observe or get access to its objective function in the optimization process. We still require the decision vectors to be in the same constraint set denoted as $\mathcal{K}$.

The objective is to minimize the total cost via local communication where each agent $i \in \mathcal{V}$ can only communicate with its immediate neighbors, i.e., adjacent agents in $\mathcal{G}$. We say an algorithm is $\mathcal{R}(T)$-*regret* if

$$\sum_{t=1}^{T} F^t(\boldsymbol{x}_i^t) - \min_{\boldsymbol{x}\in\mathcal{K}} \sum_{t=1}^{T} F^t(\boldsymbol{x}) \leq \mathcal{R}(T) \quad \forall 1 \leq i \leq n.$$

Intuitively, we look for an algorithm where every agent, with only local information exchange, has a regret $\mathcal{R}(T) = o(T)$ compared to the best solution in hindsight, which has the full information overall functions f_i^t for $1 \leq i \leq n$ at every time t.

Assumptions on the feedback functions.

For the theoretical analysis, we assume that the objective functions are **convex**. Additionally, our analysis requires *lipschitzness* and *smoothness*.

Lipschitzness : A function f is *G-Lipschitz* with respect to the ℓ_2-norm that is, $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{K}, |f(\boldsymbol{x}) - f(\boldsymbol{y})| \leq G\|\boldsymbol{x} - \boldsymbol{y}\|$.

Smoothness : A function f is β-*smooth* with respect to the ℓ_2-norm that is $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{K}, f(\boldsymbol{y}) \leq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x}\rangle + (\beta/2)\|\boldsymbol{y} - \boldsymbol{x}\|^2$ or equivalently $\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\| \leq \beta\|\boldsymbol{x} - \boldsymbol{y}\|$.

Indexing in Decentralized Online Optimization.

In decentralized online optimization algorithms, there are three levels of indexing used to keep track of different components. The first level of indexing is related to the decentralization aspect of the problem, where i is used as an index to represent the model of agent i with $\boldsymbol{x}_i$ and the function of agent i with f_i. The second level of indexing is related to the online nature of the problem, where t is used as an index to represent different rounds of the algorithm, where agent i makes a decision $\boldsymbol{x}_i^t$ and receives the loss function f_i^t, running from 1 to T. Finally, the third level of indexing is related to the iterative nature of the algorithm, where ℓ is used as an index to represent different iterations within each round, where agent i updates its decision $\boldsymbol{x}_{i,\ell}^t$, running from 1 to L.

Tab. 1.1: Summary of notations

$\boldsymbol{x} \in \mathbb{R}^d$	we use lowercase **bold** font to indicate vectors
$\mathbf{A} \in \mathbb{R}^{p \times q}$	we use UPPERCASE **bold** font to indicate matrices
$\mathcal{K} \subseteq \mathbb{R}^d$	the optimization constraint set
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	the graph with the set of nodes/agents $\mathcal{V} = [n] = \{1, \ldots, n\}$ and edges $\mathcal{E}$
$\mathcal{N}(i)$	the set of neighbors of agent i in the graph $\mathcal{G}$
$\deg(i)$	the number of neighbors of agent i in the graph $\mathcal{G}$
n	the number of nodes/agents
d	the dimension/size of the model
T	the time horizon, meaning the number of online rounds
f_i^t	the local function of agent i at round t
F^t	the aggregated normalized sum of all local functions : $F^t = \frac{1}{n}\sum_{i=1}^n f_i^t$
$\mathcal{R}^T$ or $\mathcal{R}(T)$	the regret with hindsight of an online algorithm running for T rounds
$\nabla f(\boldsymbol{x})$	the gradient of the function f at $\boldsymbol{x}$
$\widetilde{\nabla} f(\boldsymbol{x})$	a stochastic estimate of the gradient of the function f at $\boldsymbol{x}$
G	the Lipschitz constant
β	the smoothness constant
$\mathcal{M}_{\boldsymbol{x}}(e)$	the output of the model when given input e
$\Phi(e)$	the ground truth output for input e.

1.7 Contributions

This thesis aims to propose decentralized online convex optimization algorithms with tight theoretical regret bounds and conducts experimental studies within the desired constraints of the edge computing paradigm.

1.7.1 A decentralized online algorithm with tight regret guarantee (DMFW)

We first present in chapter 2 an online algorithm for decentralized online convex optimization that uses stochastic gradient estimates and achieves an asymptotically tight regret bound. Our algorithm is inspired by the Meta Frank-Wolfe (MFW) algorithm in the (centralized) online setting and the Decentralized Frank-Wolfe (DFW) algorithm in the decentralized (offline) setting.

Meta Frank Wolfe (Centralized Online Settings) [CHK18].

The MFW algorithm, introduced by Chen et al. [CHK18] in the context of online convex and DR-submodular (Diminishing Returns-submodular) optimization, generalizes the classic Frank-Wolfe algorithm to an online version where offline linear minimization is replaced by online ones. MFW uses the concept of *meta-actions* first introduced in [SG08]. Meta-actions allow the conversion of offline algorithms into online ones. In MFW, we use meta-actions to mimic the process of the (offline) FW algorithm in an online setting. Recall that in online settings, f^t remains unknown until the algorithm commits to a decision $\boldsymbol{x}^t$ at round t. As a thought experiment, suppose that we are given the sequence of functions $\{f^t\}_{t\in[T]}$ in advance to optimize. Therefore, we could construct the normalized aggregate function $f = \frac{1}{T}\sum_{t=1}^{T} f^t$ as an offline objective function, and use the FW algorithm to optimize f. FW requires to compute L linear optimizations (see equation 1.5) where L is the number of iterations:

$$\boldsymbol{v}_1 \leftarrow \arg\min_{\boldsymbol{v}\in\mathcal{K}} \langle \boldsymbol{v}, \nabla f(\boldsymbol{x}_1)\rangle \quad , \quad \cdots \quad , \quad \boldsymbol{v}_L \leftarrow \arg\min_{\boldsymbol{v}\in\mathcal{K}} \langle \boldsymbol{v}, \nabla f(\boldsymbol{x}_L)\rangle \tag{1.13}$$

Nevertheless, it is impossible to execute the above sequence 1.13 in online settings because f^t is unknown. Thus, MFW mimics the sequence 1.13 by using L online linear optimizers $\{\mathcal{O}_\ell\}_{\ell\in[L]}$, called *oracles*. The oracles aim to learn in an online fashion to produce $\boldsymbol{v}_\ell^t$ according to the previously observed sequence of gradients $\nabla f^1(\boldsymbol{x}_\ell^1), \ldots, \nabla f^t(\boldsymbol{x}_\ell^{t-1})$. Thus, we obtain the following sequence at each online round t :

$$\boldsymbol{v}_1^t \leftarrow \text{output of } \mathcal{O}_1 \text{ at round } t \quad , \quad \cdots \quad , \quad \boldsymbol{v}_L^t \leftarrow \text{output of } \mathcal{O}_L \text{ at round } t \tag{1.14}$$

Using the sequence $\boldsymbol{v}_1^t, \ldots, \boldsymbol{v}_L^t$, we compute the sequence of iterates $\boldsymbol{x}_1^t, \ldots, \boldsymbol{x}_L^t$ with the FW update described in equation 1.6. Once the MFW algorithm commits to its decision $\boldsymbol{x}_L^t$ at round t, the function f^t is observed, and we can compute the sequence of gradients $\{\nabla f^t(\boldsymbol{x}_\ell^t)\}_{\ell\in[L]}$, and feedback the gradients respectively to each L online oracles as linear coefficients:

$$\mathcal{O}_1 \xleftarrow{\text{feedback}} \nabla f^t(\boldsymbol{x}_1^t) \quad , \quad \cdots \quad , \quad \mathcal{O}_L \xleftarrow{\text{feedback}} \nabla f^t(\boldsymbol{x}_L^t) \tag{1.15}$$

Formally, each oracle $\mathcal{O}_\ell$ yields a sequence of decisions $\boldsymbol{v}_\ell^1, \ldots, \boldsymbol{v}_\ell^T$ in an online fashion while minimizing the regret $\mathcal{R}_{\mathcal{O}_\ell}^T$ defined as follows :

$$\mathcal{R}_{\mathcal{O}_\ell}^T = \sum_{t=1}^{T} \langle \boldsymbol{v}_\ell^t, \nabla f^t(\boldsymbol{x}_\ell^t)\rangle - \min_{\boldsymbol{v}_\ell^*\in\mathcal{K}} \sum_{t=1}^{T} \langle \boldsymbol{v}_\ell^*, \nabla f^t(\boldsymbol{x}_\ell^t)\rangle \tag{1.16}$$

Decentralized Frank Wolfe (Decentralized Offline Settings) [Wai+17].

The DFW algorithm, given by [Wai+17], solves (offline) decentralized convex and non-convex problems. Recall that in decentralized optimization, the aim is to optimize a function F which is

spread through a network into n smaller component $\{f_i\}_{i\in[n]}$. In [Wai+17], it is assumed that the objective function F is of the form

$$F(\cdot) = \frac{1}{n}\sum_{i=1}^{n} f_i(\cdot) \tag{1.17}$$

DFW extends the classic Frank-Wolfe algorithm in such a way that at every step, each agent computes the update direction using local information aggregating from its neighbors (similar to [Joh+08; SJ16]). Let the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the topology of the network with n agents, DFW builds a matrix $\mathbf{W} \in \mathbb{R}^{n\times n}$ such that

- $\mathbf{W}$ is a *doubly stochastic matrix*:

$$\sum_{i=1}^{n} \mathbf{W}_{ij} = \sum_{j=1}^{n} \mathbf{W}_{ij} = 1 \text{ with } 0 \leq \mathbf{W}_{ij} \leq 1 \text{ for all } i, j \in [n]$$

- and $\mathbf{W}$ is a *weighted adjacency matrix* of $\mathcal{G}$:

$$\begin{cases} \mathbf{W}_{ij} = 0 & \text{if } i \text{ and } j \text{ are not neighbors} \\ 0 < \mathbf{W}_{ij} \leq 1 & \text{otherwise.} \end{cases}$$

Each agent i uses this matrix $\mathbf{W}$ to reach a local *consensus* with its neighborhood $\mathcal{N}(i)$ by computing in a *synchronized* fashion a local network average $\boldsymbol{y}_{i,\ell}$ of their local models $\boldsymbol{x}_{i,\ell}$. Recall that in a decentralized setting, access to the global objective function F is not possible, therefore DFW uses a *gradient-tracking* technique (see [Wai+17; QL18; NOS17; DS16]) that computes $\boldsymbol{g}_{i,\ell}$ to estimate the gradient $\nabla F(\boldsymbol{x}_{i,\ell})$. More specifically, each agent i decentralizes the FW iterations as follows:

FW $\qquad\qquad$ **DFW**

$$\boldsymbol{v}_\ell \leftarrow \arg\min_{v\in\mathcal{K}} \langle \boldsymbol{v}, \nabla F(\boldsymbol{x}_\ell)\rangle \quad \xrightarrow{\text{gradient-tracking}} \quad \begin{cases} \boldsymbol{g}_{i,1} \leftarrow \nabla f_i(\boldsymbol{y}_{i,1}) \\ \boldsymbol{d}_{i,\ell} \leftarrow \sum_{j\in\mathcal{N}(i)} \mathbf{W}_{ij} \nabla \boldsymbol{g}_{j,\ell} \\ \boldsymbol{g}_{i,\ell+1} \leftarrow \nabla f_i(\boldsymbol{y}_{i,\ell+1}) - \nabla f_i(\boldsymbol{y}_{i,\ell}) + \boldsymbol{d}_{i,\ell} \\ \boldsymbol{v}_{i,\ell} \leftarrow \arg\min_{\boldsymbol{v}\in\mathcal{K}} \langle \boldsymbol{v}, \boldsymbol{g}_{i,\ell}\rangle \end{cases} \tag{1.18}$$

where $\boldsymbol{d}_{i,\ell}$ is the local network average that estimates $\nabla F(\boldsymbol{x}_\ell)$.

$$\begin{array}{cc} \textbf{FW} & \textbf{DFW} \\ \boldsymbol{x}_{\ell+1} \leftarrow (1-\eta_k)\boldsymbol{x}_\ell + \eta_\ell \boldsymbol{v}_\ell \quad \xrightarrow{\text{model- consensus}} & \begin{cases} \boldsymbol{y}_{i,\ell} \leftarrow \sum_{j\in\mathcal{N}(i)} \mathbf{W}_{ij}\boldsymbol{x}_{j,\ell} \\ \boldsymbol{x}_{i,\ell+1} \leftarrow (1-\eta_\ell)\boldsymbol{y}_{i,\ell} + \eta_\ell \boldsymbol{v}_{i,\ell} \end{cases} \end{array} \tag{1.19}$$

where $\boldsymbol{y}_{i,\ell}$ is the local network average that estimates $\boldsymbol{x}_\ell$.

Decentralized Meta Frank Wolfe (Decentralized Online Settings).

We combine the ideas of MFW and DFW to design our algorithm. At a high level, we maintain several oracles that solve online linear optimization problems (for example, online gradient descent, follow-the-perturbed-leader algorithms [Haz16, chapter 5],etc). Each agent i is equipped with L oracles $\{\mathcal{O}_{i,\ell}\}_{\ell\in[L]}$. At every round t, each agent i performs L update steps as in the Frank-Wolfe algorithm where every update vector is constructed by combining the outputs of online oracles $\left\{\boldsymbol{v}_{i,\ell}^t\right\}_{\ell\in[L]}$ and the current iterates of its neighbors $\left\{\boldsymbol{x}_{j,\ell}^t\right\}_{j\in\mathcal{N}(i)}$:

$$\begin{aligned} &\boldsymbol{v}_{i,\ell}^t \leftarrow \text{output of } \mathcal{O}_{i,\ell} \text{ at round } t && \text{(similar to \textbf{MFW}, see (1.14))} \\ &\begin{cases} \boldsymbol{y}_{i,\ell}^t \leftarrow \sum_{j\in\mathcal{N}(i)} \mathbf{W}_{ij}\boldsymbol{x}_{j,\ell}^t \\ \boldsymbol{x}_{i,\ell+1}^t \leftarrow (1-\eta_\ell)\boldsymbol{y}_{i,\ell}^t + \eta_\ell \boldsymbol{v}_{i,\ell}^t \end{cases} && \text{(model-consensus from \textbf{DFW}, see (1.19))} \end{aligned}$$

After aggregating information from neighbors, each agent subtly computes a sequence of feedback linear functions with coefficients $\left\{\boldsymbol{d}_{i,\ell}^t\right\}_{\ell\in[L]}$ using the *gradient-tracking* technique to its associated oracle for the subsequent time:

$$\begin{aligned} &\begin{cases} \boldsymbol{g}_{i,1}^t \leftarrow \nabla f_i^t(\boldsymbol{x}_{i,1}^t) \\ \boldsymbol{d}_{i,\ell}^t \leftarrow \sum_{j\in\mathcal{N}(i)} \mathbf{W}_{ij}\boldsymbol{g}_{i,\ell}^t \\ \boldsymbol{g}_{i,\ell+1}^t \leftarrow \nabla f_i^t(\boldsymbol{x}_{i,\ell+1}^t) - \nabla f_i^t(\boldsymbol{x}_{i,\ell}^t) + \boldsymbol{d}_{i,\ell}^t \quad (\ell > 1) \end{cases} && \text{(gradient-tracking from \textbf{DFW}, see (1.18))} \\ &\mathcal{O}_{i,\ell} \xleftarrow{\text{feedback}} \boldsymbol{d}_{i,\ell}^t && \text{(similar to \textbf{MFW})} \end{aligned}$$

The main features of our algorithms are the following.

- They achieve asymptotically tight regret bounds of $\mathcal{O}\left(\sqrt{T}\right)$ matching to the optimal regret bound for online convex optimization even in the centralized setting.
- They have the flexibility to be converted into projection-free. Using projection-free oracles (provided, for instance, by the follow-the-perturbed-leader oracle), the algorithm is

projection-free $\mathcal{O}\left(\sqrt{T}\right)$-regret which improves upon the best-known $\mathcal{O}\left(T^{3/4}\right)$ projection-free the online algorithm in the decentralized setting [Zha+17].

- They are robust to stochastic gradient estimates (note that the one in [Zha+17] requires exact gradients). To achieve this result, we provide a distributed variance reduction version, which might be of independent interest to designing algorithms in decentralized online settings. Moreover, by reducing variance in our algorithms, we can reduce the effect of data heterogeneity[1] closely related to the variance.

Synchronization issues of DMFW.

The DMFW algorithm executes L iterations in each round. Each agent makes L *synchronized* exchanges with its direct neighbors in a round. Thus, the number of communications for each agent i is of order $\mathcal{O}\left(\deg\left(i\right)L\right)$ where $\deg\left(i\right)$ is the degree of i, and the overall number of communications is of order $\mathcal{O}\left(\left|\mathcal{E}\right|L\right)$ where $\left|\mathcal{E}\right|$ is the number of edges of the given graph. Moreover, the regret analysis of DMFW depends on the number of iterations. Hence it depends on the number of synchronized communication in each round. In other words, there is a trade-off between the quality of prediction and communication complexity.

1.7.2 A random walk walk approach (RWMFW)

Synchronization is a strong property that in some applications we wish to avoid. In chapter 3, we propose an alternative algorithm, Random Walk Meta Frank Wolfe (RWMFW), keeping most of the desired features.

Random Walk Meta Frank Wolfe (Decentralized Online Settings).

Inspired by the Random Walk Gradient Descent [SSY18; AR21] in the offline decentralized settings, instead of exchanging with all neighbors, only one agent communicates with only one of its neighbors chosen with a uniform probability distribution. Intuitively, instead of the consensus and gradient-tracking technique of DMFW, RWMFW moves the model in the graph via a random walk. At each step k of the walk, the visited agent i_k updates the model $\boldsymbol{x}_k$ with the help of the corresponding online oracle $\mathcal{O}_k$, and then feedbacks the local gradient $\nabla f_i^t(\boldsymbol{x}_k)$ after observing its local cost function f_i^t.

DMFW *vs* RWMFW.

To summarize, RWMFW maintains the following features :

[1] Heterogeneity in stochastic gradient estimates refers to the diversity or non-uniformity in the data being analyzed, which can lead to high variance in the gradient estimates and negatively impact the optimization process. Variance reduction techniques help address this issue by reducing the variance of the gradient estimates, resulting in a more stable and reliable optimization process.

- It achieves asymptotically tight regret bounds of $\mathcal{O}\left(\sqrt{T}\right)$, when the length of the walk is of order $\mathcal{O}(T)$.
- It still has the flexibility to be converted into projection-free.

RWMFW brings the following advantages compared to DMFW:

- It has an asynchronous communication protocol.
- Only one agent is active at a given iteration of the algorithm.

However, the convergence of the RWMFW depends on the number of steps of the uniform random walk for the distribution to reach the uniform stationary distribution, known as the *mixing-time* of the *Markov chain*. Therefore, the advantages and disadvantages of RWMFW over DMFW depend on the network's topology. To further illustrate this dependence, consider the contrast between the mixing time of the complete graph topology and the cycle graph:

- Mixing time of a complete graph of n nodes is 1.
- Mixing time of a cycle graph with n nodes is of order $\mathcal{O}(n^2)$.

1.7.3 Experimental contributions

Validation of the theoretical results.

We demonstrate the performance of both DMFW and RWMFW for optimizing the online multiclass logistic regression model on real-world standard image datasets (MNIST,CIFAR$_{10}$) on a simulated decentralized environment by comparing with centralized online projection-free algorithms([Xie+20]). We outperform the best-known decentralized constrained online algorithms (Decentralized Regularized Online Frank Wolfe (DROFW) [Zha+17]) in terms of regret bounds.

A proof of concept.

In addition, we implemented DMFW on the multiclass logistic regression problem on a physical network of computing units with limited memory and processing power, specifically Raspberry Pi 3b+. We demonstrate the algorithms' scalability and their potential to edge computing applications.

A smart-building application (Temperature forecasting).

Finally, we conducted an experiment using DMFW to an actual smart-building application by optimizing non-convex models, more precisely a Long Short-Term Memory (LSTM) artificial neural network.

1.7.4 Positioning of the proposed algorithms (**DMFW**, **SDMFW** and **RWMFW**)

This section serves to summarize and position the contribution of this thesis. It is first important to establish that any algorithm for the online convex optimization settings incurs $\Omega\left(\sqrt{T}\right)$ regret in the worst case([Haz16]). In other words, if an algorithm incurs $\mathcal{O}\left(\sqrt{T}\right)$ regret then its regret bound is tight. A summary of our results and previous ones is provided in Table 1.2.

	Regret	**Per round cost**	**Decentralized**		**Proj-free**	**Stochastic Gradients**
			Per round communications	**Sync**		
OGD [Zin03]	$\mathcal{O}\left(\sqrt{T}\right)$	$\mathcal{O}\left(1\right)$	_	_	No	Yes
OFW [Haz16]	$\mathcal{O}\left(T^{3/4}\right)$	$\mathcal{O}\left(1\right)$	_	_	Yes	No
MFW [Che+18]	$\mathcal{O}\left(\sqrt{T}\right)$	$\mathcal{O}\left(T^{3/2}\right)$	_	_	Yes	Yes
DROFW [Zha+17]	$\mathcal{O}\left(T^{3/4}\right)$	$\mathcal{O}\left(1\right)$ (per node)	$\mathcal{O}\left(1\right)$	Yes	Yes	No
DMFW Chapter 2	$\mathcal{O}\left(\sqrt{T}\right)$	$\mathcal{O}\left(\sqrt{T}\right)$ (per node)	$\mathcal{O}\left(\lvert\mathcal{E}\rvert\sqrt{T}\right)$	Yes	Yes	No
SDMFW Chapter 2	$\mathcal{O}\left(\sqrt{T}\right)$	$\mathcal{O}\left(T^{3/2}\right)$ (per node)	$\mathcal{O}\left(\lvert\mathcal{E}\rvert T^{3/2}\right)$	Yes	Yes	Yes
RWMFW Chapter 3	$\mathcal{O}\left(\sqrt{T}\right)$	$\mathcal{O}\left(T\right)$ (overall nodes)	$\mathcal{O}\left(T\right)$	No	Yes	No

Tab. 1.2: Comparison with previous work on both *decentralized* and *centralized online* algorithms, and our proposed algorithms (in bold). Stochastic Decentralized Meta Frank-Wolfe (**SDMFW**) is an extension of **DMFW** in which a variance reduction is used to handle stochastic gradient estimates. The first column shows the theoretical regret bound. The second column shows the computation cost per round to achieve such a regret bound. Note that for decentralized algorithms, the *per round cost* is distinguished by two types. DROFW, **DMFW** and **SDMFW**'s cost are counted for each node. However, **RWMFW** is of different nature: the cost of nodes is not equally balanced and depends on the random walk. Therefore, we present the overall combined cost per round of all the nodes is $\mathcal{O}\left(T\right)$. For decentralized algorithms, the column "*per round communications*" shows the number of exchanges done in a round. The column "Sync" shows whether or not *synchronization* is required. The column "*Proj-free*" shows whether or not the algorithm is projection-free. Finally, the last column shows whether or not the algorithm is robust to stochastic gradient estimates.

Many decentralized algorithms have been proposed in both offline and online settings based on gradient methods. To further understand the positioning of this work, we will present related works in both *offline* and *online decentralized optimization*

Decentralized Offline Optimization.

Many algorithms have been proposed for decentralized optimization. Decentralized gradient and sub-gradient descent methods are the most popular algorithms that combine local gradient descent steps with a surrogate gradient averaging using consensus algorithm [KQW16; Ned+09]. Later, building on the idea of penalty functions in the centralized Nesterov method, Jakovetic et al. [JXM14] proposed distributed Nesterov gradient with consensus iterations that achieve faster convergence for smooth convex functions. However, the algorithm in [JXM14] places a large communication burden and needs extra coordination among agents. Thus, Qu and Li [QL20] presented an accelerated distributed Nesterov gradient descent that uses only communication step per gradient evaluation and achieves a fast convergence rate. Mokhtari et al. [AQA17] and Bajovic et al. [Baj+17] proposed Newton-like methods for distributed optimization whereas Eisen et al. [EMR17] presented quasi-Newton algorithms. Apart from these primal methods, numerous dual methods have been proposed in the literature for decentralized optimization [IAG08; RNB05; JMX15] and methods that use dual variables to coordinate solutions to local subproblems include ADMM [BPC11] and CoCoA [Jag+14; Ma+15; Ma+17]. Accelerated decentralized methods have been recently studied [HBM19a; HBM19b]. Very recently [Sca+19] provided optimal convergence rates and corresponding optimal algorithms for distributed convex problems under a variety of regularity assumptions and communication schemes. The most closely related to this thesis's work is the DFW algorithm, given by Wai et al. [Wai+17] to study (offline) convex and non-convex problems, extends the classic Frank-Wolfe algorithm by using local information aggregating from its neighbors.

Decentralized Online Optimization.

Decentralized optimization has also been explored in the online setting. Yan et al. [Yan+13] introduced distributed online projected subgradient descent and showed vanishing regret for convex and strongly convex functions. In contrast, Hosseini et al. [HCM13] extended distributed dual averaging technique to online setting using a general regularized projection for both unconstrained and constrained optimization. Furthermore, Shahrampour and Jadbabaie [SJ18] proposed a decentralized variation of the celebrated mirror descent algorithm. Decentralized online unconstrained optimization problems have been investigated in [DJ14] and [AGL17], where an online descent algorithm and a distributed online sub-gradient push sum algorithm are presented, respectively. Many other algorithms for global/local constrained decentralized optimization have been proposed [KJR15; NLR15; HCM16; LNR17]. Despite their success, these algorithms have limited applicability to many real-world problems as they require expensive projection operators. To circumvent these costs, a distributed variant of online conditional gradient [Haz16] was designed and analyzed in [Zha+17] that requires linear minimizers and uses exact gradients. Their algorithm is inspired by the original online conditional gradient algorithm for centralized settings that requires the computation of exact gradients [Haz16].

However, computing exact gradients may be prohibitively expensive for moderately sized data and intractable when a closed form does not exist.

2 A Stochastic Conditional Gradient Algorithm for Decentralized Online Convex Optimization

2.1 Introduction

We study in this chapter several problems at the intersection of *decentralized optimization* and *online learning*. Decentralized optimization plays a vital role in machine learning and has recently garnered much attention due to its inherent advantage in handling edge computations. Many decentralized optimization algorithms, both projection and projection-free algorithms with theoretical guarantees, have been proposed in the literature, focusing mainly on offline settings. However, for most real-world machine learning problems, the data is often revealed online, for example, in the case of recommender systems, image/video processing, and stock portfolio management. Therefore, in this work, we study decentralized optimization within the framework of online settings with constraints imposed on the optimization solutions (e.g., sparsity or low rank of matrices). More specifically, we consider the problem of optimizing an aggregate of convex loss functions that arrive over time such that their components are distributed over a connected network. We present a consensus-based online decentralized Frank-Wolfe algorithm that uses stochastic gradient estimates, which achieves an asymptotically tight regret guarantee of $\mathcal{O}\left(\sqrt{T}\right)$ where T is a given time horizon.

2.1.1 Contributions

In this chapter, the main contribution is to propose new online decentralized convex optimization algorithms that achieve asymptotically tight regret bounds. We highlight the main features of our algorithms.

- They achieve asymptotically tight regret bounds of $\mathcal{O}\left(\sqrt{T}\right)$[1] matching to the optimal regret bound for online convex optimization even in the centralized setting.

[1]The dependence of different constant parameters is explicitly specified in corresponding theorems.

- They have the flexibility to be converted into projection-free. Using projection-free oracles (provided, for instance, by the follow-the-perturbed-leader oracle), our algorithm is projection-free $\mathcal{O}\left(\sqrt{T}\right)$-regret which improves upon the best-known $\mathcal{O}\left(T^{3/4}\right)$ projection-free online algorithm in the decentralized setting [Zha+17].
- They are robust to stochastic gradient estimates (note that the one in [Zha+17] requires exact gradients). To achieve this result, we provide a distributed version of variance reduction (Lemma 4), which might be of independent interest for designing algorithms in decentralized online settings. Moreover, by reducing variance in our algorithms, we can reduce the effect of data heterogeneity closely related to their variance.
- Additionally, the experimental results demonstrate their advantages over the other existing methods, and they validate that the empirical regret is similar to the one of the centralized setting.

Federated Learning.

Our work is along the line of federated learning [McM+21; Li+20]. In the latter, offline centralized training (a start network where a central server is connected to several devices) is the currently dominant paradigm. However, decentralized training is more efficient than centralized one when operating on networks with low bandwidth or high latency [Lia+17; HBJ18]. In this chapter, we consider a further step by studying arbitrary communication networks without a central coordinator, and the local data (so local cost functions) evolve over time.

2.2 Conditional Gradient based Algorithms for Decentralized Online Convex Optimization

In this section, we present first an online algorithm in the setting where the *exact* gradients of functions can be computed efficiently and the functions are smooth. Subsequently, we show that the smoothness assumption on functions can be avoided using standard techniques. Finally, we extend the algorithm to the setting with *stochastic* gradients estimates. In those algorithms, for clarity, we assume the knowledge of the time horizon T. This assumption can be removed by the standard doubling trick (algorithm 4).

2.2.1 An Algorithm with Exact Gradients.

Algorithm description.

Our algorithm is inspired by two recent algorithms, namely, the centralized Meta Frank-Wolfe (MFW) algorithm in the online setting and the Decentralized Frank-Wolfe (DFW) algorithm in the

decentralized (offline) setting. The MFW algorithm was introduced by [CHK18] in the context of online convex and DR-submodular optimization, for generalizing the online Frank-Wolfe algorithm where offline linear optimization functions are replaced by online linear minimization oracles. The DFW algorithm was proposed by [Wai+17] to study (offline) convex and non-convex problems. Essentially, it extends the classical Frank-Wolfe algorithm in such a way that at every step, each agent computes the update direction using local information aggregating from its neighbors (similar to [Joh+08; SJ16]).

At the beginning of the algorithm, we maintain several gradient-based optimizers that solve online linear optimization problems (for example, online gradient descent, follow-the-perturbed-leader algorithms, etc.). We call them *oracles*. At every time t, each agent i executes L steps of the Frank-Wolfe algorithm where every update vector (for iterations $1 \leq \ell \leq L$) is constructed by combining the outputs of linear optimization oracles $\mathcal{O}_{j,\ell}$ and the current vectors of its neighbors $j \in \mathcal{N}(i)$. The solution $\boldsymbol{x}_i^t$ for each agent/node $1 \leq i \leq n$ is produced at the end of the L-th step. Subsequently, after aggregating informations related to functions f_j^t for $j \in \mathcal{N}(i)$, the algorithm subtly computes a vector $\boldsymbol{d}_{i,\ell}^t$ and feedbacks $\langle \cdot, \boldsymbol{d}_{i,\ell}^t \rangle$ as the reward function at time t to the oracle $\mathcal{O}_{i,\ell}$ for $1 \leq \ell \leq L$.

Metropolis-Hastings matrix.

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, fix a doubly stochastic, non-negative symmetric matrix W such that $\mathbf{W}_{ij} > 0$ iff $(i,j) \in E$; and $\sum_{j=1}^n \mathbf{W}_{ij} = 1$ for all i; and $\sum_{i=1}^n \mathbf{W}_{ij} = 1$ for all j. We use this matrix to compute the aggregation between neighbours. A desired mixing matrix can be constructed using *Metropolis-Hastings* weights [Has70]:

$$\mathbf{W}_{ij} = \begin{cases} 1/(1+\max\{d_i, d_j\}) & \text{if } (i,j) \in E, \\ 0 & \text{if } (i,j) \notin E \text{ and } i \neq j, \\ 1 - \sum_{j \in N(i)} \mathbf{W}_{ij} & \text{if } i = j, \end{cases}$$

The formal description is given in Algorithm 1.

Algorithm 1 Decentralized online algorithm with exact gradients

Input: A convex set $\mathcal{K}$, a time horizon T, a parameter L, online linear optimization oracles $\mathcal{O}_{i,1}, \dots, \mathcal{O}_{i,L}$ for each player $1 \leq i \leq n$, step sizes $\eta_\ell \in (0,1)$ for all $1 \leq \ell \leq L$
Output: Each agent i, plays in an online fashion $\boldsymbol{x}_i^1, \dots, \boldsymbol{x}_i^T$

Initialize linear optimizing oracle $\mathcal{O}_{i,\ell}$ for all $1 \leq \ell \leq L$
for $t = 1$ to T **do**
  for every agent $1 \leq i \leq n$ **do**
    Initialize arbitrarily $\boldsymbol{x}_{i,1}^t \in \mathcal{K}$
    for $1 \leq \ell \leq L$ **do**
      Let $\boldsymbol{v}_{i,\ell}^t$ be the output of oracle $\mathcal{O}_{i,\ell}$ at time step t.
      Send $\boldsymbol{x}_{i,\ell}^t$ to all neighbors $\mathcal{N}(i)$
      Once receiving $\boldsymbol{x}_{j,\ell}^t$ from all neighbors $j \in \mathcal{N}(i)$, set $\boldsymbol{y}_{i,\ell}^t \leftarrow \sum_j \mathbf{W}_{ij} \boldsymbol{x}_{j,\ell}^t$.
      Compute $\boldsymbol{x}_{i,\ell+1}^t \leftarrow (1-\eta_\ell)\boldsymbol{y}_{i,\ell}^t + \eta_\ell \boldsymbol{v}_{i,\ell}^t$.
    end for
    Set $\boldsymbol{x}_i^t \leftarrow \boldsymbol{x}_{i,L+1}^t$
    Decision at time t: play $\boldsymbol{x}_i^t$
    Receive function f_i^t
    Set $\boldsymbol{g}_{i,1}^t \leftarrow \nabla f_i^t(\boldsymbol{y}_{i,1}^t)$
    for $1 \leq \ell \leq L$ **do**
      Send $\boldsymbol{g}_{i,\ell}^t$ to all neighbors $\mathcal{N}(i)$.
      After receiving $\boldsymbol{g}_{j,\ell}^t$ from all neighbors $j \in \mathcal{N}(i)$, compute $\boldsymbol{d}_{i,\ell}^t \leftarrow \sum_{j \in \mathcal{N}(i)} \mathbf{W}_{ij} \boldsymbol{g}_{j,\ell}^t$ and $\boldsymbol{g}_{i,\ell+1}^t \leftarrow (\nabla f_i^t(\boldsymbol{y}_{i,\ell+1}^t) - \nabla f_i^t(\boldsymbol{y}_{i,\ell}^t)) + \boldsymbol{d}_{i,\ell}^t$.
      Feedback function $\langle \boldsymbol{d}_{i,\ell}^t, \cdot \rangle$ to oracles $\mathcal{O}_{i,\ell}$. (The cost of the oracle $\mathcal{O}_{i,\ell}$ at time t is $\langle \boldsymbol{d}_{i,\ell}^t, \boldsymbol{v}_{i,\ell}^t \rangle$.)
    end for
  end for
end for

Theorem 1. *Let $\mathcal{K}$ be a convex set with diameter D. Assume that functions F^t are β-smooth and $\|\nabla F^t\| \leq G$ for every t.*
Then, with the step-sizes $\eta_\ell = \min\{1, A/\ell\}$ where $A = \max\left\{\frac{G}{\beta D}, \frac{(C_d+\beta C_p)}{2\beta D}, 3\right\}$ for all $1 \leq \ell \leq L$, $C_p = \ell_0 \sqrt{n} AD$ and $C_d = \ell_0 \sqrt{n} \max\left\{|\lambda_2(\mathbf{W})|\left(\frac{\beta D}{1-|\lambda_2(\mathbf{W})|} + G\right), (6C_p + 10AD)\beta\right\} + 3\beta(C_p + 2AD)$, Algorithm 1 guarantees that for every $1 \leq i \leq n$,

$$\sum_{t=1}^{T} F^t(\boldsymbol{x}_i^t) \leq \min_{\boldsymbol{x} \in \mathcal{K}} \sum_{t=1}^{T} F^t(\boldsymbol{x}) + \frac{2\beta A D^2 T}{L} + 3(A+1) \cdot \mathcal{R}^T + \frac{G C_p T}{L}$$

where $\mathcal{R}^T$ is the regret of online linear minimization oracles. Consequently, choosing $L = \mathcal{O}\left(\sqrt{T}\right)$ and oracles as gradient descent (or follow-the-perturbed leader if aiming for projection-free algorithm) with regret $\mathcal{R}^T = \mathcal{O}\left(GD\sqrt{T}\right)$, for every $1 \leq i \leq n$, we have

$$\sum_{t=1}^{T} F^t(\boldsymbol{x}_i^t) \leq \min_{\boldsymbol{x}\in\mathcal{K}} \sum_{t=1}^{T} F^t(\boldsymbol{x}) + \mathcal{O}\left((G + C_d + \beta C_p + 3\beta)GD\sqrt{T}\right).$$

You will find the analysis in Section 2.3.

Remark 1.

At each time step, each agent performs $\mathcal{O}(L)$ computations and $\mathcal{O}(L)$ communications. In terms of the complexity, for a precision of ϵ, the algorithm performs $L = \mathcal{O}\left(\sqrt{T}\right) = \mathcal{O}(1/\epsilon)$ computations and $\mathcal{O}(1/\epsilon)$ of communications for each time step. Moreover, each message is of size d which is typically much smaller than the size of the data batch for each time step.

Remark 2.

Projection-free methods for online convex optimization with smooth functions have been widely studied. Several algorithms have been given with regret bound depending on the number of linear optimization computations per time step; some of them make only a single computation per time step whereas others perform more. Recently, [EE20] proposed a refined metric to compare different methods: let $\mathcal{A}$ be an online optimization algorithm, and define $T_\epsilon(\mathcal{A})$ to be the overall number of gradient oracle calls as well as linear optimization calls made until the average regret becomes at most ϵ. Subsequently, they gave an algorithm with $T_\epsilon = \mathcal{O}(d/\epsilon^3)$ for smooth functions (in the centralized online setting). Inspecting the guarantee bound of Theorem 2, we obtain $T_\epsilon = \mathcal{O}(dn^2/\epsilon^3)$ in the decentralized online setting; restricting to the centralized setting (i.e., $n = 1$), our bound matches to that in [EE20] up to a constant factor.

2.2.2 Extension to Non-Smooth Functions

Given a function f, we smooth the function by the standard convolution technique. Specifically, define

$$\hat{f}_\delta(\boldsymbol{x}) := \int_{\mathbb{R}^n} f(\boldsymbol{x} - \boldsymbol{r})\mu_\delta(\boldsymbol{r})d\boldsymbol{r} = \int_{\boldsymbol{r}\sim\mathbb{B}^n} f(\boldsymbol{x} - \delta\boldsymbol{r})d\boldsymbol{r}$$

where μ_δ is density of the uniform distribution on a ball of radius δ.

Algorithm 2.

At every time step t, apply Algorithm 1 in which every function f_i^t is replaced by its smooth counterpart $(\hat{f}_i^t)_\delta$ where $\delta = 1/\sqrt{T}$ for all $1 \leq i \leq n$ and $1 \leq t \leq T$. The parameters are $\eta_\ell = \min\{1, A/\ell\}$ where $A = \max\{\frac{G}{\beta D}, \frac{(C_d+\beta C_p)}{2\beta D}, 3\}$ for all $1 \leq \ell \leq L$ and the parameter $L = T$.

We make use of the following lemma.

Lemma 1 ([Haz16], Lemma 2.6). *Assume that function $f : \mathcal{K}(\subseteq \mathbb{R}^d) \to \mathbb{R}$ is convex and G-Lipschitz. Then, function $\hat{f}_\delta$ is convex, β-smooth with $\beta = \frac{Gd}{\delta}$. Moreover, $\|\hat{f}_\delta(\boldsymbol{x}) - f(\boldsymbol{x})\|_\infty \leq \delta G$ for every $\boldsymbol{x} \in \mathcal{K}$.*

Theorem 2. *Let $\mathcal{K}$ be a convex set with diameter D. Assume that $\|\nabla F^t\| \leq G$ for every t. Then, Algorithm 2 achieves that for every $1 \leq i \leq n$,*

$$\begin{aligned}\sum_{t=1}^{T} F^t(\boldsymbol{x}_i^t) \leq \min_{\boldsymbol{x}\in\mathcal{K}} &\sum_{t=1}^{T} F^t(\boldsymbol{x}) \\ &+ \mathcal{O}\left((G + C_d + \beta C_p)dGD\sqrt{T}\right).\end{aligned}$$

Proof. Define $\hat{F}_\delta^t(\cdot) = \frac{1}{n}\sum_{i=1}^{n}(f_i^t)_\delta(\cdot)$ for every time $1 \leq t \leq T$. So $\hat{F}_\delta^t$ is convex and β-smooth with $\beta = Gd\sqrt{T}$. Applying Theorem 1, we get

$$\begin{aligned}\sum_{t=1}^{T} \hat{F}_\delta^t(\boldsymbol{x}^t) &\leq \min_{\boldsymbol{x}\in\mathcal{K}} \sum_{t=1}^{T} \hat{F}_\delta^t(\boldsymbol{x}) + \frac{2\beta A D^2 T}{L} + 3(A+1)\cdot\mathcal{O}\left(GD\sqrt{T}\right) + \frac{GC_pT}{L} \\ &\leq \min_{\boldsymbol{x}\in\mathcal{K}} \sum_{t=1}^{T} \hat{F}_\delta^t(\boldsymbol{x}) + \frac{2GdAD^2T\sqrt{T}}{L} + 3(A+1)\cdot\mathcal{O}\left(GD\sqrt{T}\right) + \frac{GC_pT}{L}. \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (\text{Definition of } \beta = Gd\sqrt{T})\end{aligned}$$

Besides, by the choice of $L = T$ and $\|\hat{F}(\boldsymbol{x}) - F(\boldsymbol{x})\|_\infty \leq G/\sqrt{T}$ for every $\boldsymbol{x} \in \mathcal{K}$, we deduce

$$\sum_{t=1}^{T} F^t(\boldsymbol{x}^t) \leq \min_{\boldsymbol{x}\in\mathcal{K}} \sum_{t=1}^{T} F^t(\boldsymbol{x}) + \mathcal{O}\left((G + C_d + \beta C_p)dGD\sqrt{T}\right).$$

□

2.2.3 An Algorithm with Stochastic Gradient Estimates

In this section, we extend the previous algorithm to the setting where one has access only to stochastic estimates of gradients. The new algorithm, Algorithm 3, is similar to Algorithm 1 but now instead of using exact gradient, we use stochastic gradient estimates. (Note that the stochastic variables/parameters are denoted by the same letter as its exact counterpart with an additional tilde symbol.) The only difference is the use of variance reduction (line 19) and so the oracle feedback (line 20) in Algorithm 3. Variance reduction is necessary in order to deal with stochastic gradient estimate and in the algorithm, we adopt the technique proposed by [MHK18] (which has been used in several contexts for example [Che+18; Xie+20]).

Building on the analysis of Theorem 2 and a variance reduction technique, we show that Algorithm 3 achieves an assymptotically tight regret guarantee.

Theorem 3. *Let $\mathcal{K}$ be a convex set with diameter D. Assume that for every $1 \leq t \leq T$,*

1. *functions f_i^t are β-smooth, i.e. ∇f_i^t is β-Lipschitz, (so F^t is β-smooth);*
2. *$\|\nabla f_i^t\| \leq G$ (so $\|\nabla F^t\| \leq G$);*
3. *the gradient estimates are unbiased with bounded variance σ^2, i.e., $\mathbb{E}[\widetilde{\nabla} f_i^t(\boldsymbol{x}_{i,\ell}^t)] = \nabla f_i^t(\boldsymbol{x}_{i,\ell}^t)$ and $\|\widetilde{\nabla} f_i^t(\boldsymbol{x}_{i,\ell}^t)] - \nabla f_i^t(\boldsymbol{x}_{i,\ell}^t)\| \leq \sigma$ for every $1 \leq i \leq n$ and $1 \leq \ell \leq L$;*
4. *the gradient estimates are $\widetilde{\beta}$-Lipschitz.*

Then, choosing the step-sizes $\eta_\ell = \min\{1, {}^{A}/_{\ell}\}$ where $A = \max\{\frac{G}{\beta D}, \frac{(C_d+\beta C_p)}{2\beta D}, 3\}$ for all $1 \leq \ell \leq L$, Algorithm 1 guarantees that for every $1 \leq i \leq n$,

$$\sum_{t=1}^{T} \mathbb{E}\left[F^t(\boldsymbol{x}_i^t)\right] \leq \min_{\boldsymbol{x}\in\mathcal{K}} \sum_{t=1}^{T} F^t(\boldsymbol{x}) + \frac{2\beta A D^2 T}{L} + 3(A+1)\cdot\mathcal{O}\left(\sqrt{T}\right) + \mathcal{O}\left(\frac{ATDQ^{1/2}}{L^{1/3}}\right)$$

where $Q = 6(\widetilde{\beta}^2 + \beta^2)(2C_p + AD)^2 + \sigma^2 + {}^{3B^2}/_{2}$, $B = 4C_d + 2\beta\,[2C_p + AD,]$, $C_p = \ell_0\sqrt{n}AD$ and $C_d = \ell_0\sqrt{n}\max\left\{|\lambda_2(\mathbf{W})|\left(\frac{\beta D}{1-|\lambda_2(\mathbf{W})|} + G\right), (6C_p + 10AD)\beta\right\} + 3\beta\,(C_p + 2AD)$.

Consequently, by choosing $L = T^{3/2}$, we obtain the regret of $\mathcal{O}\left(\sqrt{T}\right)$.

You will find the analysis in 2.4.

Algorithm 3 Stochastic online decentralized algorithm

Input: A convex set $\mathcal{K}$, a time horizon T, a parameter L, online linear optimization oracles $\mathcal{O}_{i,1}, \dots, \mathcal{O}_{i,L}$ for each player $1 \leq i \leq n$, step sizes $\eta_\ell \in (0,1)$ for all $1 \leq \ell \leq L$.
Output: Each agent i, plays in an online fashion $\boldsymbol{x}_i^1, \dots, \boldsymbol{x}_i^T$

Initialize linear optimizing oracle $\mathcal{O}_{i,\ell}$ for all $\ell \in \{1, \dots L\}$
Initialize arbitrarily $\boldsymbol{x}_{i,1}^t \in \mathcal{K}$ and set $\boldsymbol{a}_{i,0}^t \leftarrow \mathbf{0}$
for $t = 1$ to T **do**
  for every agent $1 \leq i \leq n$ **do**
    Let $\boldsymbol{v}_{i,\ell}^t$ be the output of oracle $\mathcal{O}_{i,\ell}$ at time step t for all $\ell \in [L]$ and $i \in [n]$
    for every $\ell \in [L]$ **do**
      Send $\boldsymbol{x}_{i,\ell}^t$ to neighbors in $\mathcal{N}(i)$.
      Once receiving $\boldsymbol{x}_{j,\ell}^t$ from all neighbors $j \in \mathcal{N}(i)$, compute $\boldsymbol{y}_{i,\ell}^t \leftarrow \sum_j \mathbf{W}_{ij} \boldsymbol{x}_{j,\ell}^t$.
      Compute $\boldsymbol{x}_{i,\ell+1}^t \leftarrow (1-\eta_\ell)\boldsymbol{y}_{i,\ell}^t + \eta_\ell \boldsymbol{v}_{i,\ell}^t$.
    end for
    Set $\boldsymbol{x}_i^t \leftarrow \boldsymbol{x}_{i,L+1}^t$
    Decision at time t: play $\boldsymbol{x}_i^t$
    Receive function $f_i^t(\cdot)$ and an unbiased gradient estimate $\widetilde{\nabla} f_i^t(\cdot)$.
    Set $\widetilde{\boldsymbol{g}}_{i,1}^t \leftarrow \widetilde{\nabla} f_i^t(\boldsymbol{x}_{i,1}^t)$.
    for $1 \leq \ell \leq L$ **do**
      Send $\widetilde{\boldsymbol{g}}_{i,\ell}^t$ to all neighbors $\mathcal{N}(i)$.
      Once receiving $\widetilde{\boldsymbol{g}}_{j,\ell}^t$ from all neighbors $j \in \mathcal{N}(i)$, set $\widetilde{\boldsymbol{d}}_{i,\ell}^t \leftarrow \sum_{j \in \mathcal{N}(i)} \mathbf{W}_{ij} \widetilde{\boldsymbol{g}}_{j,\ell}^t$.
      Set $\widetilde{\boldsymbol{g}}_{i,\ell+1}^t \leftarrow (\widetilde{\nabla} f_i^t(\boldsymbol{x}_{i,\ell+1}^t) - \widetilde{\nabla} f_i^t(\boldsymbol{x}_{i,\ell}^t)) + \widetilde{\boldsymbol{d}}_{i,\ell}^t$
      Set $\widetilde{\boldsymbol{a}}_{i,\ell}^t \leftarrow (1-\rho_\ell) \cdot \widetilde{\boldsymbol{a}}_{i,\ell-1}^t + \rho_\ell \cdot \widetilde{\boldsymbol{d}}_{i,\ell}^t$.
      Feedback $\langle \widetilde{\boldsymbol{a}}_{i,\ell}^t, \boldsymbol{v}_{i,\ell}^t \rangle$ to oracles $\mathcal{O}_{i,\ell}$. (The cost of the oracle $\mathcal{O}_{i,\ell}$ at time t is $\langle \widetilde{\boldsymbol{a}}_{i,\ell}^t, \boldsymbol{v}_{i,\ell}^t \rangle$.)
    end for
  end for
end for

Remark 3.

The effect of σ, a measure of data heterogeneity [McM+21; Li+20], on the regret can be reduced by increasing the number of iteration L. Thus, a feature of our algorithms is the ability to reduce the heterogeneity of agents' data.

2.2.4 Removing the knowledge of T

In this section, we show how to remove the assumption on the knowledge of T for Algorithm 3. The procedure for Algorithms 1 is similar. In particular, we use the standard doubling trick in which Algorithm 3 is invoked repeatedly with a doubling time horizon.

Algorithm 4 Stochastic online decentralized algorithm with Doubling Trick

Input: A convex set $\mathcal{K}$ and Algorithm 3

1: **for** $m = 0, 1, 2, \ldots$ **do**
2: $L := (2^{m+1})^{3/2}$
3: Run Algorithm 3 with horizon 2^m, from the $(2^m + 1)$-th iteration to the 2^{m+1}-th iteration.
4: Let $\boldsymbol{x}_i^t$'s for $2^m + 1 \leq t \leq 2^{m+1}$ be the solution of Algorithm 3.
5: **end for**

Theorem 4. *Given Theorem 3, the following inequality holds true for Algorithm 4:*

$$\sum_{t=1}^{T} \mathbb{E}\left[F^t(\boldsymbol{x}_i^t)\right] \leq \min_{\boldsymbol{x}\in\mathcal{K}} \sum_{t=1}^{T} F^t(\boldsymbol{x}) + \mathcal{O}\left(\sqrt{T}\right).$$

Proof. Fix an $1 \leq i \leq n$. From Theorem 3, for each m, it follows that

$$\sum_{t=2^m+1}^{2^{m+1}} \mathbb{E}\left[F^t(\boldsymbol{x}_i^t)\right] \leq \min_{\boldsymbol{x}\in\mathcal{K}} \sum_{t=2^m+1}^{2^{m+1}} F^t(\boldsymbol{x}) + \mathcal{O}\left(2^{(m+1)}/2\right)$$

Summing this quantity of $m = 0, 1, 2, \ldots, \lceil \log_2(T+1) \rceil - 1$, we have that

$$\begin{aligned}\sum_{t=1}^{T} \mathbb{E}\left[F^t(\boldsymbol{x}_i^t)\right] &\leq \min_{\boldsymbol{x}\in\mathcal{K}} \sum_{t=1}^{T} F^t(\boldsymbol{x}) + \sum_{m=0}^{\lceil \log_2(T+1) \rceil - 1} \mathcal{O}\left(2^{(m+1)}/2\right) \\ &\leq \min_{\boldsymbol{x}\in\mathcal{K}} \sum_{t=1}^{T} F^t(\boldsymbol{x}) + \mathcal{O}\left(\sqrt{T}\right).\end{aligned}$$

□

2.3 Analysis in Section 2.2

In the analysis, we denote $\overline{\boldsymbol{x}}_\ell^t := \frac{1}{n}\sum_{j=1}^n \boldsymbol{x}_{j,\ell}^t$ and $F_\ell^t := \frac{1}{n}\sum_{j=1}^n f_j^t(\boldsymbol{y}_{j,\ell}^t)$. We adapt the lemma 2 from [Wai+17]. First, let us establish some useful claims:

Claim 1 ([Wai+17], Fact 1)**.** *Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$ be as set of vectors and $\overline{\boldsymbol{x}} = \frac{1}{n}\sum_{i=1}^n \boldsymbol{x}_i$ be their average. Suppose $\mathbf{W}$ is a doubly stochastic, non-negative matrix. The output after performing one round of average consensus:*

$$\boldsymbol{y}_i = \sum_{j=1}^{n} \mathbf{W}_{ij}\boldsymbol{x}_j \tag{2.1}$$

must satisfy

$$\sqrt{\sum_{i=1}^{n} \|\boldsymbol{y}_i - \overline{\boldsymbol{x}}\|^2} \leq |\lambda_2(\mathbf{W})| \cdot \sqrt{\sum_{i=1}^{n} \|\boldsymbol{x}_i - \overline{\boldsymbol{x}}\|^2} \tag{2.2}$$

where $\lambda_2(\mathbf{W})$ *is the second largest eigenvalue of* $\mathbf{W}$.

Lemma 2 ([Wai+17], Lemmas 1 and 2)**.** *Define* ℓ_0 *as the smallest integer such that*

$$\lambda_2(\mathbf{W}) \leq \left(\frac{\ell_0}{\ell_0 + 1}\right)^2 \tag{2.3}$$

Notice that ℓ_0 *is upper bounded by:* $\ell_0 \leq \left\lceil \left(|\lambda_2(\mathbf{W}|^{-1/2} - 1 \right)^{-1} \right\rceil$ *which is finite under the assumption that the second largest (in magnitude) eigenvalue of* $\mathbf{W}$ *is strictly less than one, i. e.,* $|\lambda_2(\mathbf{W})| < 1$. *Assume that functions* f_j^t*'s are* β*-smooth,* G*-Lipschitz that is,* $\|\nabla f_j^t\| \leq G$ *for every* $1 \leq t \leq T$ *and every* $1 \leq j \leq n$ *and the diameter of* $\mathcal{K}$ *is* D. *Then, for every* $1 \leq \ell \leq L + 1$,

$$\Delta p_\ell := \max_{t=1}^{T} \max_{i=1}^{n} \|\boldsymbol{y}_{i,\ell}^t - \overline{\boldsymbol{x}}_\ell^t\| \leq \frac{C_p}{\ell} \tag{2.4}$$

$$\Delta d_\ell := \max_{t=1}^{T} \max_{i=1}^{n} \|\boldsymbol{d}_{i,\ell}^t - \nabla F_\ell^t\| \leq \frac{C_d}{\ell} \tag{2.5}$$

where $C_p = \ell_0 \sqrt{n} AD$ *and* $C_d = \ell_0 \sqrt{n} \max\left\{ |\lambda_2(\mathbf{W})| \left(\frac{\beta D}{1 - |\lambda_2(\mathbf{W})|} + G \right), (6C_p + 10AD)\beta \right\} + 3\beta\left(C_p + 2AD\right)$ *where* $\lambda_2(\mathbf{W})$ *is the second largest eigenvalue of* $\mathbf{W}$.

Proof. **Proof for** $\Delta p_\ell \leq \frac{C_p}{\ell}$**.**
We will prove this lemma by using an induction on ℓ. It suffices to show that for all $\ell \geq 1$ that for $\ell = 1$ to ℓ,

$$\sqrt{\sum_{i=1}^{n} \left\|\boldsymbol{y}_{i,\ell+1}^t - \overline{\boldsymbol{x}}_{\ell+1}^t\right\|^2} \leq \frac{C_p}{\ell} \tag{2.6}$$

For the base case ($1 \leq \ell \leq \ell_0$), the above inequality is true since $\boldsymbol{y}_{i,\ell}^t, \overline{\boldsymbol{x}}_\ell^t \in \mathcal{K}$ and the diameter of $\mathcal{K}$ is bounded by D. For the induction step, let us assume that $\sqrt{\sum_{i=1}^{n} \left\|\boldsymbol{y}_{i,\ell+1}^t - \overline{\boldsymbol{x}}_{\ell+1}^t\right\|^2} \leq \frac{C_p}{\ell}$ until some $\ell \geq \ell_0$.

Using claim 1, we observe that

$$\sum_{i=1}^{n} \left\|\boldsymbol{y}_{i,\ell+1}^t - \overline{\boldsymbol{x}}_{\ell+1}^t\right\|^2 \leq \lambda_2(\mathbf{W})^2 \sum_{i=1}^{n} \left\|\boldsymbol{x}_{i,\ell+1}^t - \overline{\boldsymbol{x}}_{\ell+1}^t\right\|^2$$

$$= \lambda_2(\mathbf{W})^2 \sum_{i=1}^{n} \left\| (1-\eta_\ell)(\boldsymbol{y}_{i,\ell}^t - \overline{\boldsymbol{x}}_\ell^t) + \eta_\ell \left(\boldsymbol{v}_{i,\ell}^t - \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{v}_{i,\ell}^t \right) \right\|^2$$

(Definition of $\boldsymbol{x}_{i,\ell}^t$ and Lemma 3)

$$\leq \lambda_2(\mathbf{W})^2 \sum_{i=1}^{n} \left[\|\boldsymbol{y}_{i,\ell}^t - \overline{\boldsymbol{x}}_\ell^t\|^2 + 2\eta_\ell D \|\boldsymbol{y}_{i,\ell}^t - \overline{\boldsymbol{x}}_\ell^t\| + \eta_\ell^2 D^2 \right]$$

(By Cauchy-Schwarz and $\left\| \boldsymbol{v}_{i,\ell}^t - \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{v}_{i,\ell}^t \right\| \leq D$ because $\mathcal{K}$ is bounded)

$$\leq \lambda_2(\mathbf{W})^2 \left[\frac{C_p^2}{\ell^2} + n\frac{A^2}{\ell^2}D^2 + \sum_{i=1}^{n} \frac{AD}{\ell} \|\boldsymbol{y}_{i,\ell}^t - \overline{\boldsymbol{x}}_\ell^t\| \right]$$

(Applying induction hypothesis)

$$\leq \lambda_2(\mathbf{W})^2 \left[\frac{C_p^2}{\ell^2} + \frac{nA^2D^2}{\ell^2} + 2\frac{A}{\ell} D\sqrt{n} \sqrt{\sum_{i=1}^{n} \|\boldsymbol{y}_{i,\ell}^t - \overline{\boldsymbol{x}}_\ell^t\|^2} \right]$$

(By Cauchy-Schwarz)

$$\leq \lambda_2(\mathbf{W})^2 \left[\frac{C_p^2}{\ell^2} + \frac{nA^2D^2}{\ell^2} + 2\frac{AD}{\ell^2}\sqrt{n}C_p \right]$$

(Applying induction hypothesis)

$$\leq \lambda_2(\mathbf{W})^2 \left[\frac{C_p^2 + nA^2D^2 + 2AD\sqrt{n}C_p}{\ell^2} \right]$$

$$\leq \lambda_2(\mathbf{W})^2 \frac{(C_p + \sqrt{n}AD)^2}{\ell^2}$$

Note that by definition of $C_p = \ell_0\sqrt{n}AD$, we observe $C_p + \sqrt{n}AD \leq C_p + \frac{C_p}{\ell_0}$. Thus, we get

$$\sum_{i=1}^{n} \left\| \boldsymbol{y}_{i,\ell+1}^t - \overline{\boldsymbol{x}}_{\ell+1}^t \right\|^2 \leq \lambda_2(\mathbf{W})^2 \left[\frac{\ell_0 + 1}{\ell_0 \cdot \ell} \right]^2 C_p^2 \tag{2.7}$$

Consequently, from equation 2.3, we observe that for all $\ell \geq \ell_0$,

$$|\lambda_2(\mathbf{W})| \frac{\ell_0 + 1}{\ell_0 \cdot \ell} \leq \frac{1}{\ell + 1} \tag{2.8}$$

Hence, we complete the induction step by the following:

$$\sum_{i=1}^{n} \left\| \boldsymbol{y}_{i,\ell+1}^t - \overline{\boldsymbol{x}}_{\ell+1}^t \right\|^2 \leq \frac{C_p}{\ell + 1} \tag{2.9}$$

Proof for $\Delta d_\ell \leq \frac{C_d}{\ell}$.

In this proof, we will use an intermediate step in which we will define $\widehat{\nabla F}_\ell^t = \frac{1}{n}\sum_{i=1}^{n} \nabla f_{i,\ell}(\boldsymbol{x}_{i,\ell})$ and bound the following:

$$\left\| \boldsymbol{d}_{i,\ell}^t - \widehat{\nabla F}_\ell^t \right\| \leq \frac{\hat{C}_d}{\ell} \tag{2.10}$$

where $\hat{C}_d = \ell_0\sqrt{n}\max\left\{|\lambda_2(\mathbf{W})|\left(\frac{\beta D}{1-|\lambda_2(\mathbf{W})|}+G)\right),(6C_p+10AD)\beta\right\}$.

Assume that we already bounded (2.10), we can prove the lemma as follows:

$$\begin{aligned}
\left\|\boldsymbol{d}_{i,\ell}^t - \nabla F_\ell^t\right\| &\leq \left\|\boldsymbol{d}_{i,\ell}^t - \widehat{\nabla F}_\ell^t\right\| + \left\|\nabla F_\ell^t - \widehat{\nabla F}_\ell^t\right\| && \text{(Triangular inequality)}\\
&\leq \frac{\hat{C}_d}{\ell} + \beta\frac{1}{n}\sum_{i=1}^n \left\|\boldsymbol{x}_{i,\ell}^t - \boldsymbol{y}_{i,\ell}^t\right\| \\
& && \text{(Assumption of equation 2.10 and } \beta\text{-smoothness of } F)\\
&\leq \frac{\hat{C}_d}{\ell} + \frac{\beta}{n}\sum_{i=1}^n \left[\left\|\boldsymbol{x}_{i,\ell+1}^t - \boldsymbol{y}_{i,\ell}^t\right\| + \left\|\boldsymbol{x}_{i,\ell+1}^t - \boldsymbol{x}_{i,\ell}^t\right\|\right] && \text{(Triangular inequality)}\\
&\leq \frac{\hat{C}_d}{\ell} + \frac{\beta}{n}\sum_{i=1}^n \eta_\ell\left\|\boldsymbol{v}_{i,\ell}^t - \boldsymbol{y}_{i,\ell}^t\right\| + \frac{(3C_p+5AD)}{\ell} && \text{(By definition of } \boldsymbol{x}_{i,\ell+1}^t)\\
&\leq \frac{\hat{C}_d}{\ell} + \beta\left[\frac{AD}{\ell} + \frac{(3C_p+5AD)}{\ell}\right] \\
& && \text{(By definition of } \eta_\ell \text{ and the constraint set is bounded by } D)
\end{aligned}$$

where

$$C_d = \ell_0\sqrt{n}\max\left\{|\lambda_2(\mathbf{W})|\left(\frac{\beta D}{1-|\lambda_2(\mathbf{W})|}+G)\right),(6C_p+10AD)\beta\right\} + 3\beta\left(C_p+2AD\right) \tag{2.11}$$

We will prove that by induction on ℓ that $\frac{1}{n}\sum_{i=1}^n \boldsymbol{g}_{i,\ell}^t = \frac{1}{n}\sum_{i=1}^n \boldsymbol{d}_{i,\ell}^t = \widehat{\nabla F}_\ell^t$.

First, observe that

$$\begin{aligned}
\frac{1}{n}\sum_{i=1}^n \boldsymbol{d}_{i,\ell}^t &= \frac{1}{n}\sum_{i=1}^n\sum_{j=1}^n \mathbf{W}_{ij}\boldsymbol{g}_{j,\ell}^t \\
&= \frac{1}{n}\sum_{j=1}^n \boldsymbol{g}_{j,\ell}^t \sum_{i=1}^n \mathbf{W}_{ij} \\
&= \frac{1}{n}\sum_{j=1}^n \boldsymbol{g}_{j,\ell}^t
\end{aligned} \tag{2.12}$$

We can also observe that, for $\ell = 1$ we get by definition of $\boldsymbol{g}_{1,\ell}^t = \nabla f_i^t(\boldsymbol{x}_{i,\ell}^t)$:

$$\frac{1}{n}\sum_{i=1}^n \boldsymbol{g}_{i,1}^t = \frac{1}{n}\sum_{i=1}^n \nabla f_i^t(\boldsymbol{x}_{i,1}^t) = \widehat{\nabla F}_1^t \tag{2.13}$$

By induction on ℓ, assume that $\frac{1}{n}\sum_{i=1}^{n} \boldsymbol{g}_{i,\ell}^t = \widehat{\nabla F}_\ell^t$, then:

$$\begin{aligned}
\frac{1}{n}\sum_{i=1}^{n} \boldsymbol{g}_{i,\ell+1}^t &= \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{d}_{i,\ell}^t + \widehat{\nabla F}_{\ell+1}^t - \widehat{\nabla F}_\ell^t && \text{(Definition of } \boldsymbol{g}_{i,\ell+1}^t\text{)} \\
&= \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{g}_{i,\ell}^t + \widehat{\nabla F}_{\ell+1}^t - \widehat{\nabla F}_\ell^t && \text{(By equation 2.12)} \\
&= \widehat{\nabla F}_\ell^t + \widehat{\nabla F}_{\ell+1}^t - \widehat{\nabla F}_\ell^t && \text{(By induction hypothesis)} \\
&= \widehat{\nabla F}_{\ell+1}^t && (2.14)
\end{aligned}$$

We are now proving equation 2.5. As mentioned earlier, it suffices to prove:

$$\sqrt{\sum_{i=1}^{n} \left\| \boldsymbol{d}_{i,\ell}^t - \widehat{\nabla F}_\ell^t \right\|} \leq \frac{\hat{C}_d}{\ell} \tag{2.15}$$

where $\hat{C}_d = \ell_0\sqrt{n}\max\left\{|\lambda_2(\mathbf{W})|\left(\frac{\beta D}{1-|\lambda_2(\mathbf{W})|}+G\right),(6C_p+10AD)\beta\right\}$ for all $\ell \geq 1$ using induction. For $\ell = 1$ to $\ell = \ell_0$, we shall prove that the left hand side of the inequality is bounded. To proceed, we define the $d \times n$ matrices:

$$\mathbf{E}_\ell^t = \left(\boldsymbol{d}_{1,\ell}^t \cdots \boldsymbol{d}_{n,\ell}^t\right) - \left(\widehat{\nabla F}_\ell^t \cdots \widehat{\nabla F}_\ell^t\right), \tag{2.16}$$

$$\mathbf{Z}_\ell^t = \left(\nabla f_1(\boldsymbol{x}_{n,\ell}^t) \cdots \nabla f_n(\boldsymbol{x}_{n,\ell}^t)\right) \tag{2.17}$$

and observe

$$\sum_{i=1}^{n} \left\| \boldsymbol{d}_{i,\ell}^t - \widehat{\nabla F}_\ell^t \right\|^2 = \left\| \text{vec}(\mathbf{E}_\ell^t) \right\|^2 \tag{2.18}$$

where vec is the *vectorization* of a matrix is a linear transformation which converts the matrix into a column vector. Specifically, the vectorization of a $p \times q$ matrix $\mathbf{M}$, denoted vec$(\mathbf{M})$, is the $pq \times 1$ column vector obtained by stacking the columns of the matrix $\mathbf{M}$ on top of one another. Furthermore, $\mathbf{E}_1^t = \mathbf{Z}_1(\mathbf{W} - \frac{1}{n}\mathbf{1}\mathbf{1}^\top)$, and we have the following recursion for $\ell \geq 2$,

$$\begin{aligned}
\mathbf{E}_\ell^t &= \left(\boldsymbol{d}_{1,\ell}^t \cdots \boldsymbol{d}_{n,\ell}^t\right) - \left(\widehat{\nabla F}_\ell^t \cdots \widehat{\nabla F}_\ell^t\right) \\
&= \left[\left(\boldsymbol{d}_{1,\ell-1}^t \cdots \boldsymbol{d}_{n,\ell-1}^t\right) + \mathbf{Z}_\ell^t - \mathbf{Z}_{\ell-1}^t\right]\mathbf{W} - \left(\widehat{\nabla F}_\ell^t \cdots \widehat{\nabla F}_\ell^t\right) && (2.19) \\
&= \left(\mathbf{E}_{\ell-1}^t + \mathbf{Z}_\ell^t - \mathbf{Z}_{\ell-1}^t\right)\left(\mathbf{W} - \frac{1}{n}\mathbf{1}\mathbf{1}^\top\right) && (2.20)
\end{aligned}$$

where we have used the equivalence below:

$$\frac{1}{n}\mathbf{Z}_\ell^t\mathbf{1}\mathbf{1}^\top = \frac{1}{n}\left(\boldsymbol{d}_{1,\ell}^t \cdots \boldsymbol{d}_{n,\ell}^t\right)\mathbf{1}\mathbf{1}^\top = \left(\widehat{\nabla F}_\ell^t \cdots \widehat{\nabla F}_\ell^t\right) \tag{2.21}$$

For $\ell = 1$, $\left\|\text{vec}(\mathbf{E}_1^t)\right\| \leq |\lambda_2(\mathbf{W})| \sqrt{n}G$ since $\left\|\text{vec}(\mathbf{Z}_1^t)\right\| \leq \sqrt{n}G$. For $\ell \geq 2$, we have

$$\left\|\text{vec}(\mathbf{E}_\ell^t)\right\| \leq \left\|\left(\mathbf{W} - \frac{1}{n}\mathbf{1}\mathbf{1}^\top\right) \otimes \mathbf{I}\right\| \left(\left\|\text{vec}(\mathbf{E}_{\ell-1}^t)\right\| + \left\|\text{vec}\left(\mathbf{Z}_\ell^t - \mathbf{Z}_{\ell-1}^t\right)\right\|\right) \tag{2.22}$$

Since $\left\|\left(\mathbf{W} - \frac{1}{n}\mathbf{1}\mathbf{1}^\top\right) \otimes \mathbf{I}\right\| \leq |\lambda_2(\mathbf{W})|$ and the β-smoothness of f_i^t implies $\left\|\text{vec}\left(\mathbf{Z}_\ell^t - \mathbf{Z}_{\ell-1}^t\right)\right\| \leq \sqrt{n}D\beta$, this leads to

$$\left\|\text{vec}(\mathbf{E}_\ell^t)\right\| \leq |\lambda_2(\mathbf{W})| \cdot \left(\left\|\text{vec}(\mathbf{E}_{\ell-1}^t)\right\| + \sqrt{n}D\beta\right) \tag{2.23}$$

Evaluating the recursion above shows

$$\|\text{vec}(\mathbf{E}_\ell)\| \leq \frac{|\lambda_2(\mathbf{W})| \sqrt{n}D\beta}{1 - |\lambda_2(\mathbf{W})|} + |\lambda_2(\mathbf{W})| \sqrt{n}G \tag{2.24}$$

and thus proving the base step for $\ell = 1$ to $\ell = \ell_0$.

For the induction step with $\ell > \ell_0$, we suppose that $\sqrt{\sum_{i=1}^n \left\|\boldsymbol{d}_{i,\ell}^t - \widehat{\nabla F}_\ell^t\right\|^2} \leq \frac{\hat{C}_d}{\ell}$ until some $\ell \geq \ell_0$.

$$\begin{aligned}
\sum_{i=1}^n \left\|\boldsymbol{d}_{i,\ell+1}^t - \widehat{\nabla F}_{\ell+1}^t\right\|^2 &= \sum_{i=1}^n \left\|\sum_{j=1}^n \mathbf{W}_{ij}\boldsymbol{g}_{j,\ell+1}^t - \widehat{\nabla F}_{\ell+1}^t\right\|^2 && \text{(Definition of } \boldsymbol{d}_{i,\ell+1}^t\text{)} \\
&= \sum_{i=1}^n \left\|\sum_{j=1}^n \mathbf{W}_{ij}\boldsymbol{g}_{j,\ell+1}^t - \frac{1}{n}\sum_{j=1}^n \boldsymbol{g}_{j,\ell+1}^t\right\|^2 && \text{(By equation 2.14)} \\
&\leq |\lambda_2(\mathbf{W})|^2 \sum_{i=1}^n \left\|\boldsymbol{g}_{i,\ell+1}^t - \frac{1}{n}\sum_{j=1}^n \boldsymbol{g}_{j,\ell+1}^t\right\|^2 && \text{(By claim 1)} \\
&= |\lambda_2(\mathbf{W})|^2 \sum_{i=1}^n \left\|\boldsymbol{g}_{i,\ell+1}^t - \widehat{\nabla F}_{\ell+1}^t\right\|^2 && \text{(By equation 2.14)} \\
&\leq |\lambda_2(\mathbf{W})|^2 \sum_{i=1}^n \left\|\boldsymbol{d}_{i,\ell}^t + \nabla f_i^t(\boldsymbol{x}_{i,\ell+1}^t) - \nabla f_i^t(\boldsymbol{x}_{i,\ell}^t) - \widehat{\nabla F}_{\ell+1}^t\right\|^2 \\
& && \text{(Definition of } \boldsymbol{g}_{i,\ell+1}^t\text{)}
\end{aligned}$$

Define $\delta f_{i,\ell+1}^t = \nabla f_i^t(\boldsymbol{x}_{i,\ell+1}) - \nabla f_i^t(\boldsymbol{x}_{i,\ell}^t)$ and observe that $\widehat{\nabla F}_{\ell+1}^t - \widehat{\nabla F}_\ell^t = \sum_{i=1}^n \delta f_{i,\ell+1}^t$. Then, we can observe the following:

$$\sum_{i=1}^n \left\|\boldsymbol{d}_{i,\ell}^t + \delta f_{i,\ell+1}^t - \widehat{\nabla F}_{\ell+1}^t\right\|^2 \leq \sum_{i=1}^n \left(\left\|\boldsymbol{d}_{i,\ell}^t - \widehat{\nabla F}_\ell^t\right\| + \left\|\delta f_{i,\ell+1}^t - \widehat{\nabla F}_{\ell+1}^t + \widehat{\nabla F}_\ell^t\right\|\right)^2$$

(Triangular inequality)

$$\leq \sum_{i=1}^{n} \left(\left\| \boldsymbol{d}_{i,\ell}^t - \widehat{\nabla F}_\ell^t \right\|^2 + \left\| \delta f_{i,\ell+1}^t - \frac{1}{n} \sum_{j=1}^{n} \delta f_{j,\ell+1}^t \right\|^2 \right.$$
$$\left. + 2 \left\| \boldsymbol{d}_{i,\ell}^t - \widehat{\nabla F}_\ell^t \right\| \left\| \delta f_{i,\ell+1}^t - \frac{1}{n} \sum_{j=1}^{n} \delta f_{j,\ell+1}^t \right\| \right)$$
(Cauchy-Schwartz)

Observe that for all $1 \leq i \leq n$, we have the following chain:

$$\begin{aligned} \left\| \delta f_{i,\ell+1}^t \right\| &= \left\| \nabla f_i^t(\boldsymbol{x}_{i,\ell+1}^t) - \nabla f_i^t(\boldsymbol{x}_{i,\ell}^t) \right\| \\ &\leq \beta \left\| \boldsymbol{x}_{i,\ell+1}^t - \boldsymbol{x}_{i,\ell}^t \right\| && (\beta\text{-smoothness of } f_i^t) \\ &\leq \frac{\beta(3C_p + 5AD)}{\ell} && \text{(By equation 2.31)} \end{aligned}$$

Using the triangular inequality, we observe that

$$\begin{aligned} \left\| \delta f_{i,\ell+1}^t - \frac{1}{n} \sum_{j=1}^{n} \delta f_{j,\ell+1}^t \right\| &= \left\| \left(1 - \frac{1}{n}\right) \delta f_{i,\ell+1}^t - \frac{1}{n} \sum_{j \in [n]: j \neq i} \delta f_{j,\ell+1}^t \right\| \\ &\leq \left(1 - \frac{1}{n}\right) \left\| \delta f_{i,\ell+1}^t \right\| + \frac{1}{n} \sum_{j \in [n]: j \neq i} \left\| \delta f_{j,\ell+1}^t \right\| && (2.25) \\ &\leq 2 \left(1 - \frac{1}{n}\right) \left(\frac{(3C_p + 5AD)\beta}{\ell} \right) && (2.26) \\ &\leq \left(\frac{(6C_p + 10AD)\beta}{\ell} \right) && (2.27) \end{aligned}$$

Finally, by using equation 2.27 and applying the induction hypothesis, we can bound the following:

$$\begin{aligned} \sum_{i=1}^{n} \left\| \boldsymbol{d}_{i,\ell+1} - \widehat{\nabla F}_{\ell+1}^t \right\|^2 &\leq |\lambda_2(\mathbf{W})|^2 \left[\sum_{i=1}^{n} \left\| d_{i,\ell}^t - \widehat{\nabla F}_\ell^t \right\|^2 + \sum_{i=1}^{n} \left\| \delta f_{i,\ell+1}^t - \frac{1}{n} \sum_{j=1}^{n} \delta f_{j,\ell+1}^t \right\|^2 \right. \\ &\quad \left. + 2 \sum_{i=1}^{n} \left\| \delta f_{i,\ell+1}^t - \frac{1}{n} \sum_{j=1}^{n} \delta f_{j,\ell+1}^t \right\| \left\| d_{i,\ell}^t - \widehat{\nabla F}_\ell^t \right\| \right] \\ &\leq |\lambda_2(\mathbf{W})|^2 \left[\sum_{i=1}^{n} \left\| d_{i,\ell}^t - \widehat{\nabla F}_\ell^t \right\|^2 + n \left[\frac{(6C_p + 10AD)\beta}{\ell} \right]^2 \right. \\ &\quad \left. + 2 \frac{(6C_p + 10AD)\beta}{\ell} \sum_{i=1}^{n} \left\| d_{i,\ell}^t - \widehat{\nabla F}_\ell^t \right\| \right] && \text{(By equation 2.27)} \\ &\leq |\lambda_2(\mathbf{W})|^2 \left[\sum_{i=1}^{n} \left\| d_{i,\ell}^t - \widehat{\nabla F}_\ell^t \right\|^2 + n \left[\frac{(6C_p + 10AD)\beta}{\ell} \right]^2 \right. \end{aligned}$$

$$+2\sqrt{n}\frac{(6C_p+10AD)\beta}{\ell}\sqrt{\sum_{i=1}^{n}\left\|d_{i,\ell}^t-\widehat{\nabla F}_\ell^t\right\|} \qquad \text{(Cauchy-Schwarz)}$$

$$\leq |\lambda_2(\mathbf{W})|^2\left(\left(\frac{\hat{C}_d}{\ell}\right)^2+n\left(\frac{(6C_p+10AD)\,\beta}{\ell}\right)^2+\sqrt{n}\frac{\hat{C}_d 2\,(6C_p+10AD)\,\beta}{\ell^2}\right)$$

(Applyting induction hypothesis)

$$\leq |\lambda_2(\mathbf{W})|^2\frac{\left(\hat{C}_d+\sqrt{n}(6C_p+10AD)\beta\right)^2}{\ell^2}$$

$$\leq |\lambda_2(\mathbf{W})|^2\left(\frac{\ell_0+1}{\ell_0\cdot\ell}\hat{C}_d\right)^2 \qquad (2.28)$$

Using the equation 2.8, we conclude:

$$\sqrt{\sum_{i=1}^{n}\left\|\boldsymbol{d}_{i,\ell+1}-\widehat{\nabla F}_{\ell+1}^t\right\|^2}\leq |\lambda_2(\mathbf{W})|\left(\frac{\ell_0+1}{\ell_0\cdot\ell}\hat{C}_d\right)$$

$$\leq \frac{\hat{C}_d}{\ell+1} \qquad \text{(By equation 2.8)}$$

□

Claim 2. *It holds that*

$$\left\|\overline{\boldsymbol{x}}_{i,\ell+1}^t-\overline{\boldsymbol{x}}_{i,\ell}^t\right\|\leq\frac{AD}{\ell} \qquad (2.29)$$

$$\left\|\boldsymbol{y}_{i,\ell+1}^t-\boldsymbol{y}_{i,\ell}^t\right\|\leq\frac{3C_p+2AD}{\ell+1} \qquad (2.30)$$

$$\left\|\boldsymbol{x}_{i,\ell+1}^t-\boldsymbol{x}_{i,\ell}^t\right\|\leq\frac{3C_p+5AD}{\ell} \qquad (2.31)$$

$$\left\|\boldsymbol{d}_{i,\ell+1}^t-\boldsymbol{d}_{i,\ell}^t\right\|\leq\frac{B}{\ell+4} \qquad (2.32)$$

where $B=4C_d+2\beta(2C_p+AD)$.

Proof of claim.

$$\left\|\overline{\boldsymbol{x}}_{\ell+1}^t-\overline{\boldsymbol{x}}_\ell^t\right\|=\eta_\ell\left\|\left[\frac{1}{n}\left(\sum_{j=1}^{n}\boldsymbol{v}_{j,\ell}^t\right)\right]-\overline{\boldsymbol{x}}_\ell^t\right\| \qquad \text{(By Lemma 3)}$$

$$\leq \eta_\ell D \qquad \left(\tfrac{1}{n}\textstyle\sum_{j=1}^{n}\boldsymbol{v}_{j,\ell}^t\in\mathcal{K},\ \overline{\boldsymbol{x}}_{\ell-1}\in\mathcal{K}\text{ and }D=\sup_{x,y\in\mathcal{K}^2}\|x-y\|\right)$$

$$\leq \frac{AD}{\ell} \qquad \left(\text{Definition of }\eta_\ell=\min\left\{1,\tfrac{A}{\ell}\right\}\right)$$

$$\left\|\boldsymbol{y}_{j,\ell+1}^t-\boldsymbol{y}_{j,\ell}^t\right\|\leq\left\|\boldsymbol{y}_{j,\ell+1}^t-\overline{\boldsymbol{x}}_{\ell+1}^t\right\|+\left\|\overline{\boldsymbol{x}}_{\ell+1}^t-\overline{\boldsymbol{x}}_\ell^t\right\|+\left\|\overline{\boldsymbol{x}}_\ell^t-\boldsymbol{y}_{j,\ell}^t\right\| \qquad \text{(Triangle inequality)}$$

$$
\begin{aligned}
&\leq \frac{C_p}{\ell+1} + \left\|\overline{\boldsymbol{x}}_{\ell+1}^t - \overline{\boldsymbol{x}}_{\ell}^t\right\| + \frac{C_p}{\ell} && \text{(By Lemma 2)}\\
&\leq \frac{C_p}{\ell+1} + \frac{C_p}{\ell} + \frac{AD}{\ell} && \text{(By equation 2.29)}\\
&\leq \frac{3C_p + 2AD}{\ell+1} && (2.33)
\end{aligned}
$$

$$
\begin{aligned}
\left\|\boldsymbol{x}_{i,\ell+1}^t - \boldsymbol{x}_{i,\ell}\right\| &= \left\|\boldsymbol{y}_{i,\ell}^t + \eta_\ell\left(\boldsymbol{v}_{i,\ell}^t - \boldsymbol{y}_{i,\ell}\right)^t - \boldsymbol{y}_{i,\ell-1}^t - \eta_\ell\left(\boldsymbol{v}_{i,\ell-1}^t - \boldsymbol{y}_{i,\ell-1}^t\right)\right\| && \text{(Definition of } \boldsymbol{x}_{i,\ell}^t)\\
&\leq \left\|\boldsymbol{y}_{i,\ell}^t - \boldsymbol{y}_{i,\ell-1}^t\right\| + \eta_\ell\left\|\boldsymbol{v}_{i,\ell}^t - \boldsymbol{y}_{i,\ell}^t\right\| + \eta_{\ell-1}\left\|\boldsymbol{v}_{i,\ell-1}^t - \boldsymbol{y}_{i,\ell-1}^t\right\|\\
&\leq \frac{3C_p + 2AD}{\ell} + \frac{AD}{\ell} + \frac{AD}{\ell-1} && \text{(By equation 2.30)}\\
&\leq \frac{3C_p + 5AD}{\ell} && (2.34)
\end{aligned}
$$

$$
\begin{aligned}
\left\|\boldsymbol{d}_{i,\ell+1}^t - \boldsymbol{d}_{i,\ell}^t\right\| &\leq \left\|\boldsymbol{d}_{i,\ell+1}^t - \nabla F_{\ell+1}^t\right\| + \left\|\nabla F_{\ell+1}^t - \nabla F_{\ell}^t\right\| + \left\|\nabla F_{\ell}^t - \boldsymbol{d}_{i,\ell}^t\right\| && \text{(Triangle inequality)}\\
&\leq \frac{C_d}{\ell+1} + \left\|\nabla F_{\ell+1}^t - \nabla F_{\ell}^t\right\| + \frac{C_d}{\ell} && \text{(By Lemma 2)}\\
&\leq \frac{C_d}{\ell+1} + \frac{C_d}{\ell} + \frac{1}{n}\sum_{j=1}^{n}\left\|\nabla f_j^t(\boldsymbol{y}_{j,\ell+1}^t) - \nabla f_j^t(\boldsymbol{y}_{j,\ell}^t)\right\| && \text{(Definition of } \nabla F_\ell^t)\\
&\leq \frac{C_d}{\ell+1} + \frac{C_d}{\ell} + \frac{\beta}{n}\sum_{j=1}^{n}\left\|\boldsymbol{y}_{j,\ell+1}^t - \boldsymbol{y}_{j,\ell}^t\right\| && (f_j^t \text{ is } \beta\text{-smooth})\\
&\leq \frac{C_d}{\ell+1} + \frac{C_d}{\ell} + \frac{\beta C_p}{\ell+1} + \frac{\beta C_p}{\ell} + \frac{\beta AD}{\ell} && \text{(By equation 2.30)}\\
&\leq \frac{4C_d + 4\beta C_p + 2\beta AD}{\ell+4} && \text{(When } \ell \geq 7)\\
&= \frac{4C_d + 2\beta(2C_p + AD)}{\ell+4} && (2.35)
\end{aligned}
$$

□

Lemma 3. *For every* $1 \leq t \leq T$ *and* $1 \leq \ell \leq L$*, it holds that*

$$
\overline{\boldsymbol{x}}_{\ell+1}^t = \overline{\boldsymbol{x}}_{\ell}^t + \eta_\ell\left(\frac{1}{n}\sum_{j=1}^{n}\boldsymbol{v}_{j,\ell}^t - \overline{\boldsymbol{x}}_{\ell}^t\right)
$$

Proof. By Steps 8 and 9 in Algorithm 1, we have

$$
\overline{\boldsymbol{x}}_{\ell+1}^t = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_{i,\ell+1}^t \qquad \text{(Definition of } \overline{\boldsymbol{x}}_{\ell+1}^t)
$$

$$
\begin{aligned}
&= \frac{1}{n}\sum_{i=1}^{n}\left((1-\eta_\ell)\boldsymbol{y}_{i,\ell}^t + \eta_\ell \boldsymbol{v}_{i,\ell}^t\right) && \text{(Definition of } \boldsymbol{x}_{i,\ell}^t\text{)}\\
&= \frac{1}{n}\sum_{i=1}^{n}\left[(1-\eta_\ell)\left(\sum_{j=1}^{n}\mathbf{W}_{ij}\boldsymbol{x}_{j,\ell}^t\right) + \eta_\ell \boldsymbol{v}_{i,\ell}^t\right] && \text{(Definition of } \boldsymbol{y}_{i,\ell}^t\text{)}\\
&= (1-\eta_\ell)\frac{1}{n}\sum_{i=1}^{n}\left[\sum_{j=1}^{n}\mathbf{W}_{ij}\boldsymbol{x}_{j,\ell}^t\right] + \frac{1}{n}\eta_\ell\sum_{i=1}^{n}\boldsymbol{v}_{i,\ell}^t\\
&= (1-\eta_\ell)\frac{1}{n}\sum_{j=1}^{n}\left[\boldsymbol{x}_{j,\ell}^t\sum_{i=1}^{n}\mathbf{W}_{ij}\right] + \frac{1}{n}\eta_\ell\sum_{i=1}^{n}\boldsymbol{v}_{i,\ell}^t\\
&= (1-\eta_\ell)\frac{1}{n}\sum_{j=1}^{n}\boldsymbol{x}_{j,\ell}^t + \frac{1}{n}\eta_\ell\sum_{i=1}^{n}\boldsymbol{v}_{i,\ell}^t && \left(\textstyle\sum_{i=1}^{n} W_{ij} = 1 \text{ for every } j\right)\\
&= (1-\eta_\ell)\overline{\boldsymbol{x}}_\ell^t + \frac{1}{n}\eta_\ell\sum_{i=1}^{n}\boldsymbol{v}_{i,\ell}^t\\
&= \overline{\boldsymbol{x}}_\ell^t + \eta_\ell\left(\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{v}_{i,\ell}^t - \overline{\boldsymbol{x}}_\ell^t\right)
\end{aligned}
$$

where we use a property of W which is $\sum_{i=1}^{n} W_{ij} = 1$ for every j. □

We are now proving Theorem 1.

Theorem 1. *Let $\mathcal{K}$ be a convex set with diameter D. Assume that functions F^t are β-smooth and $\|\nabla F^t\| \leq G$ for every t.*
Then, with the step-sizes $\eta_\ell = \min\{1, A/\ell\}$ where $A = \max\left\{\frac{G}{\beta D}, \frac{(C_d+\beta C_p)}{2\beta D}, 3\right\}$ for all $1 \leq \ell \leq L$, $C_p = \ell_0\sqrt{n}AD$ and $C_d = \ell_0\sqrt{n}\max\left\{|\lambda_2(\mathbf{W})|\left(\frac{\beta D}{1-|\lambda_2(\mathbf{W})|} + G\right), (6C_p + 10AD)\beta\right\} + 3\beta\left(C_p + 2AD\right)$, Algorithm 1 guarantees that for every $1 \leq i \leq n$,

$$
\sum_{t=1}^{T} F^t(\boldsymbol{x}_i^t) \leq \min_{\boldsymbol{x}\in\mathcal{K}}\sum_{t=1}^{T} F^t(\boldsymbol{x}) + \frac{2\beta AD^2T}{L} + 3(A+1)\cdot\mathcal{R}^T + \frac{GC_pT}{L}
$$

where $\mathcal{R}^T$ is the regret of online linear minimization oracles. Consequently, choosing $L = \mathcal{O}\left(\sqrt{T}\right)$ and oracles as gradient descent (or follow-the-perturbed leader if aiming for projection-free algorithm) with regret $\mathcal{R}^T = \mathcal{O}\left(GD\sqrt{T}\right)$, for every $1 \leq i \leq n$, we have

$$
\sum_{t=1}^{T} F^t(\boldsymbol{x}_i^t) \leq \min_{\boldsymbol{x}\in\mathcal{K}}\sum_{t=1}^{T} F^t(\boldsymbol{x}) + \mathcal{O}\left((G + C_d + \beta C_p + 3\beta)GD\sqrt{T}\right).
$$

Proof. Let $\boldsymbol{x}^*$ be an optimal solution in hindsight that is $\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}\in\mathcal{K}} \sum_{t=1}^{T} F^t(\boldsymbol{x})$. Fix a time step $t \in \{1, 2, \ldots, T\}$. By Lemma 3,

$$\overline{\boldsymbol{x}}_{\ell+1}^t = \overline{\boldsymbol{x}}_\ell^t + \eta_\ell \left(\frac{1}{n} \sum_{j=1}^{n} \boldsymbol{v}_{j,\ell}^t - \overline{\boldsymbol{x}}_\ell^t \right). \tag{2.36}$$

For every $1 \le \ell \le L$, we have:

$$\begin{aligned}
F^t(\overline{\boldsymbol{x}}_{\ell+1}^t) - F^t(\overline{\boldsymbol{x}}_\ell^t) &\le \eta_\ell \left\langle \nabla F^t(\overline{\boldsymbol{x}}_\ell^t), \frac{1}{n} \sum_{j=1}^{n} \boldsymbol{v}_{j,\ell}^t - \overline{\boldsymbol{x}}_\ell^t \right\rangle + \frac{\beta}{2} \eta_\ell^2 \left\| \frac{1}{n} \sum_{j=1}^{n} \boldsymbol{v}_{j,\ell}^t - \overline{\boldsymbol{x}}_\ell^t \right\|^2 \\
&\qquad \text{(using } \beta\text{-smoothness of } F^t\text{)} \\
&= \eta_\ell \left\langle \nabla F^t(\overline{\boldsymbol{x}}_\ell^t), \boldsymbol{x}^* - \overline{\boldsymbol{x}}_\ell^t \right\rangle + \eta_\ell \left\langle \nabla F^t(\overline{\boldsymbol{x}}_\ell^t), \frac{1}{n} \sum_{j=1}^{n} \boldsymbol{v}_{j,\ell}^t - \boldsymbol{x}^* \right\rangle + \frac{\beta}{2} \eta_\ell^2 \left\| \frac{1}{n} \sum_{j=1}^{n} \boldsymbol{v}_{j,\ell}^t - \overline{\boldsymbol{x}}_\ell^t \right\|^2 \\
&\le \eta_\ell \langle \nabla F^t(\overline{\boldsymbol{x}}_\ell^t), \boldsymbol{x}^* - \overline{\boldsymbol{x}}_\ell^t \rangle + \eta_\ell \left\langle \nabla F^t(\overline{\boldsymbol{x}}_\ell^t), \frac{1}{n} \sum_{j=1}^{n} \boldsymbol{v}_{j,\ell}^t - \frac{1}{n} \sum_{j=1}^{n} \boldsymbol{x}^* \right\rangle + \frac{\beta}{2} \eta_\ell^2 D^2 \\
&\qquad \text{(diameter of } \mathcal{K} \text{ is bounded)} \\
&= \eta_\ell \langle \nabla F^t(\overline{\boldsymbol{x}}_\ell^t), \boldsymbol{x}^* - \overline{\boldsymbol{x}}_\ell^t \rangle + \frac{\eta_\ell}{n} \sum_{j=1}^{n} \langle \nabla F^t(\overline{\boldsymbol{x}}_\ell^t), \boldsymbol{v}_{j,\ell}^t - \boldsymbol{x}^* \rangle + \frac{\beta}{2} \eta_\ell^2 D^2 \\
&\le \eta_\ell \left(F^t(\boldsymbol{x}^*) - F^t(\overline{\boldsymbol{x}}_\ell^t) \right) + \frac{\eta_\ell}{n} \sum_{j=1}^{n} \langle \boldsymbol{d}_{j,\ell}^t, \boldsymbol{v}_{j,\ell}^t - \boldsymbol{x}^* \rangle \\
&\quad + \frac{\eta_\ell}{n} \sum_{j=1}^{n} \langle \nabla F^t(\overline{\boldsymbol{x}}_\ell^t) - \boldsymbol{d}_{j,\ell}^t, \boldsymbol{v}_{j,\ell}^t - \boldsymbol{x}^* \rangle + \frac{\beta}{2} \eta_\ell^2 D^2 \qquad \text{(convexity of } F^t\text{)} \\
&\le \eta_\ell \left(F^t(\boldsymbol{x}^*) - F^t(\overline{\boldsymbol{x}}_\ell^t) \right) + \frac{\eta_\ell}{n} \sum_{j=1}^{n} \langle \boldsymbol{d}_{j,\ell}^t, \boldsymbol{v}_{j,\ell}^t - \boldsymbol{x}^* \rangle \\
&\quad + \frac{\eta_\ell}{n} \sum_{j=1}^{n} \| \nabla F^t(\overline{\boldsymbol{x}}_\ell^t) - \boldsymbol{d}_{j,\ell}^t \| \cdot \| \boldsymbol{v}_{j,\ell}^t - \boldsymbol{x}^* \| + \frac{\beta}{2} \eta_\ell^2 D^2 \qquad \text{(Cauchy-Schwarz)}
\end{aligned}$$

Moreover, for every j, ℓ, t, we have (recall that $F_\ell^t = \frac{1}{n} \sum_{j=1}^{n} f_j^t(\boldsymbol{y}_{j,\ell}^t)$):

$$\begin{aligned}
\| \nabla F^t(\overline{\boldsymbol{x}}_\ell^t) - \boldsymbol{d}_{j,\ell}^t \| &\le \| \nabla F_\ell^t - \boldsymbol{d}_{j,\ell}^t \| + \| \nabla F^t(\overline{\boldsymbol{y}}_\ell^t) - \nabla F_\ell^t \| \qquad \text{(triangle inequality)} \\
&\le \Delta d_\ell + \left\| \frac{1}{n} \sum_{j=1}^{n} \nabla f_j^t(\overline{\boldsymbol{x}}_\ell^t) - \frac{1}{n} \sum_{j=1}^{n} \nabla f_j^t(\boldsymbol{y}_{j,\ell}^t) \right\| \quad \text{(definition of } \Delta d_\ell\text{, Lemma 2)} \\
&\le \Delta d_\ell + \frac{1}{n} \sum_{j=1}^{n} \| \nabla f_j^t(\overline{\boldsymbol{x}}_\ell^t) - \nabla f_j^t(\boldsymbol{y}_{j,\ell}^t) \| \qquad \text{(triangle inequality)} \\
&\le \Delta d_\ell + \frac{1}{n} \sum_{j=1}^{n} \beta \| \overline{\boldsymbol{x}}_\ell^t - \boldsymbol{y}_{j,\ell}^t \| \qquad (\beta\text{-smoothness of } f_j^t) \\
&= \Delta d_\ell + \beta \Delta p_\ell \qquad \text{(definition of } \Delta p_\ell\text{, Lemma 2)}
\end{aligned}$$

Hence,

$$F^t(\overline{\boldsymbol{x}}^t_{\ell+1}) - F^t(\overline{\boldsymbol{x}}^t_{\ell}) \leq \eta_\ell \left(F^t(\boldsymbol{x}^*) - F^t(\overline{\boldsymbol{x}}^t_{\ell}) \right) + \frac{\eta_\ell}{n} \sum_{j=1}^{n} \langle \boldsymbol{d}^t_{j,\ell}, \boldsymbol{v}^t_{j,\ell} - \boldsymbol{x}^* \rangle$$
$$+ \eta_\ell (\Delta d_\ell + \beta \Delta p_\ell) D + \frac{\beta}{2} \eta_\ell^2 D^2.$$

Define $h^t_\ell = F^t(\overline{\boldsymbol{x}}^t_{\ell+1}) - F^t(\boldsymbol{x}^*)$. By Lemma 2, we deduce that

$$h^t_\ell \leq (1 - \eta_\ell) h^t_{\ell-1} + \frac{\eta_\ell}{n} \sum_{j=1}^{n} \langle \boldsymbol{d}^t_{j,\ell}, \boldsymbol{v}^t_{j,\ell} - \boldsymbol{x}^* \rangle + \eta_\ell \frac{(C_d + \beta C_p) D}{\ell} + \frac{\beta}{2} \eta_\ell^2 D^2. \tag{2.37}$$

Using Inequality (2.37) recursively and by the choice of η_ℓ's, we deduce the following bound by standard techniques.

Claim 3. *For every t, it holds that*

$$h^t_L = F^t(\overline{\boldsymbol{x}}^t_{L+1}) - F^t(\boldsymbol{x}^*) \leq \frac{2\beta A D^2}{L} + \frac{1}{n} \sum_{j=1}^{n} \sum_{\ell=1}^{L} \eta_\ell \left[\prod_{\ell'=\ell+1}^{L} (1 - \eta_{\ell'}) \right] \langle \boldsymbol{d}^t_{j,\ell}, \boldsymbol{v}^t_{j,\ell} - \boldsymbol{x}^* \rangle$$

Proof of claim. Fix a time t. We prove by induction the following inequality for every $1 \leq \ell \leq L$

$$h^t_\ell = F^t(\overline{\boldsymbol{x}}^t_{\ell+1}) - F^t(\boldsymbol{x}^*) \leq \frac{2\beta A D^2}{\ell} + \frac{1}{n} \sum_{j=1}^{n} \sum_{\ell'=1}^{\ell} \eta_{\ell'} \left[\prod_{k=\ell'+1}^{\ell} (1 - \eta_k) \right] \langle \boldsymbol{d}^t_{j,\ell'}, \boldsymbol{v}^t_{j,\ell'} - \boldsymbol{x}^* \rangle \tag{2.38}$$

The claim follows by applying $\ell = L$.

For the base case where $\ell = 1$, Inequality (2.38) reads

$$h^t_1 = F^t(\overline{\boldsymbol{x}}^t_2) - F^t(\boldsymbol{x}^*) \leq 2\beta A D^2 + \frac{\eta_1}{n} \sum_{j=1}^{n} \langle \boldsymbol{d}^t_{j,1}, \boldsymbol{v}^t_{j,1} - \boldsymbol{x}^* \rangle$$

where by the choice of step-sizes, $\eta_1 = \min\{1, A\} = 1$. As F^t is G-Lipschitz and set $\mathcal{K}$ has diameter D, the left-hand side of the above inequality is at most $G\|\overline{\boldsymbol{x}}^t_2 - \boldsymbol{x}^*\| \leq GD$; whereas the right-hand side is

$$2\beta A D^2 + \frac{1}{n} \sum_{j=1}^{n} \langle \boldsymbol{d}^t_{j,1}, \boldsymbol{v}^t_{j,1} - \boldsymbol{x}^* \rangle \geq 2\beta A D^2 - \frac{1}{n} \sum_{j=1}^{n} \|\boldsymbol{d}^t_{j,1}\| \cdot \|\boldsymbol{v}^t_1 - \boldsymbol{x}^*\| \geq 2\beta A D^2 - GD \geq GD$$

since $\forall 1 \leq j \leq n$

$$
\begin{aligned}
\|\boldsymbol{d}_{j,1}^t\| &= \left\|\sum_{i=1}^n \mathbf{W}_{ij}\boldsymbol{g}_{i,1}^t\right\| && \text{(Definition of } \boldsymbol{d}_{j,1}^t\text{)} \\
&= \left\|\sum_{i=1}^n \mathbf{W}_{ij}\nabla f_i^t(\boldsymbol{x}_{i,1}^t)\right\| && \text{(Definition of } \boldsymbol{g}_{j,1}^t\text{)} \\
&\leq \sum_{i=1}^n \mathbf{W}_{ij}\left\|\nabla f_i^t(\boldsymbol{x}_{i,1}^t)\right\| && \text{(Jensen's inequality)} \\
&\leq G && (\textstyle\sum_{i=1}^n \mathbf{W}_{ij} = 1 \text{ and } \left\|\nabla f_i^t(\boldsymbol{x}_{i,1}^t)\right\| \leq G)
\end{aligned}
$$

and $A \geq \frac{G}{\beta D}$ (by the definition of A). Hence, the base case of Inequality (2.38) holds.

Assume the induction hypothesis that Inequality (2.38) holds for $\ell - 1$. We have

$$
\begin{aligned}
h_\ell^t &\leq (1-\eta_\ell)h_{\ell-1}^t + \frac{\eta_\ell}{n}\sum_{j=1}^n \langle \boldsymbol{d}_{j,\ell}^t, \boldsymbol{v}_{j,\ell}^t - \boldsymbol{x}^*\rangle + \eta_\ell\frac{(C_d+\beta C_p)D}{\ell} + \frac{\beta}{2}\eta_\ell^2 D^2 \quad \text{(by Inequality (2.37))} \\
&\leq (1-\eta_\ell)\left(\frac{2\beta A D^2}{\ell-1}\right) + \frac{(1-\eta_\ell)}{n}\left(\sum_{j=1}^n\sum_{\ell'=1}^{\ell-1}\eta_{\ell'}\left[\prod_{k=\ell'+1}^{\ell-1}(1-\eta_k)\right]\langle \boldsymbol{d}_{j,\ell'}^t, \boldsymbol{v}_{j,\ell'}^t - \boldsymbol{x}^*\rangle\right) \\
&\quad + \frac{\eta_\ell}{n}\sum_{j=1}^n \langle \boldsymbol{d}_{j,\ell}^t, \boldsymbol{v}_{j,\ell}^t - \boldsymbol{x}^*\rangle + \eta_\ell\frac{(C_d+\beta C_p)D}{\ell} + \frac{\beta}{2}\eta_\ell^2 D^2 \quad \text{(using the induction hypothesis)} \\
&= (1-\eta_\ell)\left(\frac{2\beta A D^2}{\ell-1}\right) + \frac{(1-\eta_\ell)}{n}\left(\sum_{j=1}^n\sum_{\ell'=1}^{\ell-1}\eta_{\ell'}\left[\prod_{k=\ell'+1}^{\ell-1}(1-\eta_k)\right]\langle \boldsymbol{d}_{j,\ell'}^t, \boldsymbol{v}_{j,\ell'}^t - \boldsymbol{x}^*\rangle\right) \\
&\quad + \frac{\eta_\ell}{n}\sum_{j=1}^n\left[\prod_{k=\ell+1}^{\ell}(1-\eta_k)\right]\langle \boldsymbol{d}_{j,\ell}^t, \boldsymbol{v}_{j,\ell}^t - \boldsymbol{x}^*\rangle + \eta_\ell\frac{(C_d+\beta C_p)D}{\ell} + \frac{\beta}{2}\eta_\ell^2 D^2 \\
&= (1-\eta_\ell)\left(\frac{2\beta A D^2}{\ell-1}\right) \\
&\quad + \frac{1}{n}\sum_{j=1}^n\left(\sum_{\ell'=1}^{\ell-1}\eta_{\ell'}\left[\prod_{k=\ell'+1}^{\ell-1}(1-\eta_k)\right](1-\eta_\ell)\langle \boldsymbol{d}_{j,\ell'}^t, \boldsymbol{v}_{j,\ell'}^t - \boldsymbol{x}^*\rangle + \left[\prod_{k=\ell+1}^{\ell}(1-\eta_k)\right]\langle \boldsymbol{d}_{j,\ell}^t, \boldsymbol{v}_{j,\ell}^t - \boldsymbol{x}^*\rangle\right) \\
&\quad + \eta_\ell\frac{(C_d+\beta C_p)D}{\ell} + \frac{\beta}{2}\eta_\ell^2 D^2 \\
&= (1-\eta_\ell)\left(\frac{2\beta A D^2}{\ell-1}\right) \\
&\quad + \frac{1}{n}\sum_{j=1}^n\left(\sum_{\ell'=1}^{\ell-1}\eta_{\ell'}\left[\prod_{k=\ell'+1}^{\ell}(1-\eta_k)\right]\langle \boldsymbol{d}_{j,\ell'}^t, \boldsymbol{v}_{j,\ell'}^t - \boldsymbol{x}^*\rangle + \left[\prod_{k=\ell+1}^{\ell}(1-\eta_k)\right]\langle \boldsymbol{d}_{j,\ell}^t, \boldsymbol{v}_{j,\ell}^t - \boldsymbol{x}^*\rangle\right) \\
&\quad + \eta_\ell\frac{(C_d+\beta C_p)D}{\ell} + \frac{\beta}{2}\eta_\ell^2 D^2
\end{aligned}
$$

$$
\begin{aligned}
&= (1-\eta_\ell)\left(\frac{2\beta AD^2}{\ell-1}\right) + \frac{1}{n}\sum_{j=1}^{n}\left(\sum_{\ell'=1}^{\ell}\eta_{\ell'}\left[\prod_{k=\ell'+1}^{\ell}(1-\eta_k)\right]\langle \boldsymbol{d}_{j,\ell'}^t, \boldsymbol{v}_{j,\ell'}^t - \boldsymbol{x}^*\rangle\right) \\
&\quad + \eta_\ell\frac{(C_d+\beta C_p)D}{\ell} + \frac{\beta}{2}\eta_\ell^2 D^2 \\
&\le \frac{2\beta AD^2}{\ell} + \frac{1}{n}\sum_{j=1}^{n}\sum_{\ell'=1}^{\ell}\eta_{\ell'}\left[\prod_{k=\ell'+1}^{\ell}(1-\eta_{\ell'})\right]\langle \boldsymbol{d}_{j,\ell'}^t, \boldsymbol{v}_{j,\ell'}^t - \boldsymbol{x}^*\rangle
\end{aligned}
$$

The last inequality holds since

$$
\begin{aligned}
&(1-\eta_\ell)\frac{2\beta AD^2}{\ell-1} + \eta_\ell\frac{(C_d+\beta C_p)D}{\ell} + \frac{\beta}{2}\eta_\ell^2 D^2 \\
&\le \left(1-\frac{A}{\ell}\right)\frac{2\beta AD^2}{\ell-1} + \frac{A(C_d+\beta C_p)D}{\ell^2} + \frac{\beta A^2}{2\ell^2}D^2 \\
&\le \frac{2\beta AD^2}{\ell-1} - \frac{2\beta A^2D^2}{\ell(\ell-1)} + \frac{A(C_d+\beta C_p)D + \frac{\beta}{2}A^2D^2}{\ell^2} \\
&\le \frac{2\beta AD^2}{\ell-1} - \frac{2\beta A^2D^2 - A(C_d+\beta C_p)D - \frac{\beta}{2}A^2D^2}{\ell(\ell-1)} \\
&\le \frac{2\beta AD^2}{\ell-1} - \frac{\beta A^2D^2}{\ell(\ell-1)} \le \frac{2\beta AD^2}{\ell-1} - \frac{2\beta AD^2}{\ell(\ell-1)} \\
&\le \frac{2\beta AD^2}{\ell}
\end{aligned}
$$

where the fourth and the fifth inequalities are due to $A \ge \frac{(C_d+\beta C_p)}{2\beta D}$ and $A \ge 3$, respectively. The claim follows. □

Summing the inequality in Claim 3 over all $1 \le t \le T$, we get

$$
\begin{aligned}
\sum_{t=1}^{T}\left(F^t(\overline{\boldsymbol{x}}_{L+1}^t) - F^t(\boldsymbol{x}^*)\right) &\le \frac{2\beta AD^2T}{L} + \sum_{t=1}^{T}\left(\frac{1}{n}\sum_{j=1}^{n}\sum_{\ell=1}^{L}\eta_\ell\left[\prod_{\ell'=\ell+1}^{L}(1-\eta_{\ell'})\right]\langle \boldsymbol{d}_{j,\ell}^t, \boldsymbol{v}_{j,\ell}^t - \boldsymbol{x}^*\rangle\right) \\
&= \frac{2\beta AD^2T}{L} + \frac{1}{n}\sum_{j=1}^{n}\sum_{\ell=1}^{L}\eta_\ell\left[\prod_{\ell'=\ell+1}^{L}(1-\eta_{\ell'})\right]\left(\sum_{t=1}^{T}\langle \boldsymbol{d}_{j,\ell}^t, \boldsymbol{v}_{j,\ell}^t - \boldsymbol{x}^*\rangle\right) \\
&\le \frac{2\beta AD^2T}{L} + \frac{1}{n}\sum_{j=1}^{n}\sum_{\ell=1}^{L}\eta_\ell\left[\prod_{\ell'=\ell+1}^{L}(1-\eta_{\ell'})\right]\cdot\mathcal{R}^T \\
&\le \frac{2\beta AD^2T}{L} + 3(A+1)\cdot\mathcal{R}^T \qquad (2.39)
\end{aligned}
$$

The second holds by the fact that the linear oracles $\mathcal{O}_\ell$ for $1 \le \ell \le L$ has regret $\mathcal{R}^T$, i.e., $\sum_{t=1}^T \langle \boldsymbol{d}_{j,\ell'}^t, \boldsymbol{v}_{j,\ell'}^t \rangle \le \min_{\boldsymbol{x}\in\mathcal{K}} \sum_{t=1}^T \langle \boldsymbol{d}_{j,\ell'}^t, \boldsymbol{x} \rangle + \mathcal{R}^T \le \sum_{t=1}^T \langle \boldsymbol{d}_{j,\ell'}^t, \boldsymbol{x}^* \rangle + \mathcal{R}^T$. The last inequality is due to the following claim.

Claim 4. *By the choice of the step-sizes η_ℓ, it holds that $\sum_{\ell=1}^L \eta_\ell \prod_{\ell'=\ell+1}^L (1-\eta_{\ell'}) \le 3(A+1)$.*

Proof of claim. First, we bound the term $\prod_{\ell'=\ell+1}^L (1-\eta_\ell)$. In the first case if $\frac{A}{\ell'} \ge 1$ for some $\ell \le \ell' \le L$ then $\eta_{\ell'} = 1$; so $\prod_{\ell'=\ell+1}^L (1-\eta_{\ell'}) = 0$. We consider the second case where $\frac{A}{\ell'} < 1$ for every $\ell \le \ell' \le L$; so $\eta_{\ell'} = A/\ell'$. Using the inequality $1 - z \le e^{-z}$ for $z \ge 0$,

$$\prod_{\ell'=\ell+1}^{L} (1-\eta_{\ell'}) \le e^{-A\sum_{\ell'=\ell+1}^{L} 1/\ell'} \le e^{-A\int_{\ell+2}^{L} d\ell'/\ell'} \le e^{-A(\ln L - \ln(\ell+2))} = \left(\frac{\ell+2}{L}\right)^A. \tag{2.40}$$

So, in both cases, the term $\prod_{\ell'=\ell+1}^L (1-\eta_{\ell'})$ is upper bounded by $(\frac{\ell+2}{L})^A$.

We deduce that

$$\begin{aligned}
\sum_{\ell=1}^{L} \eta_\ell \prod_{\ell'=\ell+1}^{L} (1-\eta_{\ell'}) &\le \eta_L + \eta_{L-1} + \eta_{L-2} + \sum_{\ell=1}^{L-3} \frac{A}{\ell} \left(\frac{\ell+2}{L}\right)^A \\
&\le 3 \min\left\{1, \frac{A}{L-2}\right\} + \frac{A}{L} \sum_{\ell=1}^{L-3} \frac{\ell+2}{\ell} \left(\frac{\ell+2}{L}\right)^{A-1} \\
&\le 3 + \frac{A}{L} \sum_{\ell=1}^{L-3} \left(\frac{\ell+2}{L}\right)^{A-1} \\
&\le 3 + 3A.
\end{aligned}$$

□

By Inequality (2.39) and using Lemma 2 together with the fact that F^t is Lipschitz, for every $1 \le i \le n$, we have

$$\begin{aligned}
\sum_{t=1}^{T} \left(F^t(\boldsymbol{x}_i^t) - F^t(\boldsymbol{x}^*)\right) &= \sum_{t=1}^{T} \left(F^t(\boldsymbol{x}_{i,L+1}^t) - F^t(\boldsymbol{x}^*)\right) \\
&= \sum_{t=1}^{T} \left(F^t(\overline{\boldsymbol{x}}_{L+1}^t) - F^t(\boldsymbol{x}^*)\right) + \sum_{t=1}^{T} \left(F^t(\boldsymbol{x}_{i,L+1}^t) - F^t(\overline{\boldsymbol{x}}_{L+1}^t)\right) \\
&\le \frac{2\beta A D^2 T}{L} + 3(A+1)\cdot\mathcal{R}^T + \sum_{t=1}^{T} G \left\|\boldsymbol{x}_{i,L+1}^t - \overline{\boldsymbol{x}}_{L+1}^t\right\|
\end{aligned}$$

$$
\begin{aligned}
&\leq \frac{2\beta AD^2T}{L} + 3(A+1)\cdot\mathcal{R}^T + \sum_{t=1}^{T} G(1-\eta_L)\left\|\boldsymbol{y}_{i,L}^t - \overline{\boldsymbol{x}}_L^t\right\| + \eta_L\left\|\boldsymbol{v}_{i,L}^t - \overline{\boldsymbol{v}}_L^t\right\| \\
&\leq \frac{2\beta AD^2T}{L} + 3(A+1)\cdot\mathcal{R}^T + \sum_{t=1}^{T} G\left\|\boldsymbol{y}_{i,L}^t - \overline{\boldsymbol{x}}_L^t\right\| + \eta_L\left\|\boldsymbol{v}_{i,L}^t - \overline{\boldsymbol{v}}_L^t\right\| \\
&\leq \frac{2\beta AD^2T}{L} + 3(A+1)\cdot\mathcal{R}^T + \frac{GT(C_p+AD)}{L}
\end{aligned}
$$

This completes the theorem proof. □

2.4 Analysis in Section 2.2.3

In the analysis, we aim to show that, given unbiased gradient estimates, the regret bound is asymptotically not worse than the case where we have access to exact gradients. In order to prove that, we use a part of the analysis of Theorem 1 for exact gradients and argue that when we switch to gradient estimates, we encounter only an additional loss of $\mathcal{O}\left(\sqrt{T}\right)$ in term of the regret.

First, we prove a variance bound between the random variables $\widetilde{\boldsymbol{d}}_{i,\ell}^t$ in Algorithm 3 and their exact counterparts $\boldsymbol{d}_{i,\ell}^t$ in Algorithm 1.

Lemma 4. *Given the assumptions of Theorem 3, for every $1 \leq t \leq T$, $1 \leq i \leq n$ and $1 \leq \ell \leq L$, it holds that*

$$
\mathbb{E}\left[\|\widetilde{\boldsymbol{d}}_{i,\ell}^t - \boldsymbol{d}_{i,\ell}^t\|^2\right] \leq 4(\widetilde{\beta}^2+\beta^2)(3C_p+5AD)^2 + \sigma^2.
$$

Proof. Fix an arbitrary time t. For any $1 \leq i \leq n$, we have

$$
\begin{aligned}
\mathbb{E}\|\widetilde{\boldsymbol{d}}_{i,\ell+1}^t - \boldsymbol{d}_{i,\ell+1}^t\|^2 &= \mathbb{E}\left\|\sum_{j=1}^{n}\mathbf{W}_{ij}\left[\widetilde{\nabla} f_j^t(\boldsymbol{x}_{j,\ell+1}^t) - \widetilde{\nabla} f_j^t(\boldsymbol{x}_{j,\ell}^t) - \nabla f_j^t(\boldsymbol{x}_{j,\ell+1}^t) + \nabla f_j^t(\boldsymbol{x}_{j,\ell}^t) + (\widetilde{\boldsymbol{d}}_{j,\ell}^t - \boldsymbol{d}_{j,\ell}^t)\right]\right\|^2 \\
&= \mathbb{E}\left\|\sum_{j=1}^{n}\mathbf{W}_{ij}\left[\widetilde{\nabla} f_j^t(\boldsymbol{x}_{j,\ell+1}^t) - \widetilde{\nabla} f_j^t(\boldsymbol{x}_{j,\ell}^t) - \nabla f_j^t(\boldsymbol{x}_{j,\ell+1}^t) + \nabla f_j^t(\boldsymbol{x}_{j,\ell}^t)\right]\right\|^2 \\
&\quad + \mathbb{E}\left\|\sum_{j=1}^{n}\mathbf{W}_{ij}\left(\widetilde{\boldsymbol{d}}_{j,\ell}^t - \boldsymbol{d}_{j,\ell}^t\right)\right\|^2 \\
&\leq \max_{1\leq j\leq n}\left\{\mathbb{E}\left\|\widetilde{\nabla} f_j^t(\boldsymbol{x}_{j,\ell+1}^t) - \widetilde{\nabla} f_j^t(\boldsymbol{x}_{j,\ell}^t) - \nabla f_j^t(\boldsymbol{x}_{j,\ell+1}^t) + \nabla f_j^t(\boldsymbol{x}_{j,\ell}^t)\right\|^2\right\} \\
&\quad + \max_{1\leq j\leq n}\left\{\mathbb{E}\left\|\widetilde{\boldsymbol{d}}_{j,\ell}^t - \boldsymbol{d}_{j,\ell}^t\right\|^2\right\}
\end{aligned}
$$

$$
\begin{aligned}
&\leq \max_{1\leq j\leq n}\left\{2\mathbb{E}\left\|\widetilde{\nabla} f_i^t(\boldsymbol{x}_{j,\ell+1}^t)-\widetilde{\nabla} f_i^t(\boldsymbol{x}_{j,\ell}^t)\right\|^2+2\mathbb{E}\left\|\nabla f_i^t(\boldsymbol{x}_{j,\ell+1}^t)-\nabla f_j^t(\boldsymbol{x}_{j,\ell}^t)\right\|^2\right\}\\
&\quad+\max_{1\leq j\leq n}\left\{\mathbb{E}\left\|\widetilde{\boldsymbol{d}}_{j,\ell}^t-\boldsymbol{d}_{j,\ell}^t\right\|^2\right\}\\
&\leq \max_{1\leq j\leq n}\left\{\mathbb{E}\left[2(\widetilde{\beta}^2+\beta^2)\left\|\boldsymbol{x}_{j,\ell+1}^t-\boldsymbol{x}_{j,\ell}^t\right\|^2\right]\right\}+\max_{1\leq j\leq n}\left\{\mathbb{E}\left\|\widetilde{\boldsymbol{d}}_{j,\ell}^t-\boldsymbol{d}_{j,\ell}^t\right\|^2\right\}
\end{aligned}
\tag{2.41}
$$

The second equality holds since $\mathbb{E}[\widetilde{\nabla} f_i^t(\boldsymbol{x}_{i,\ell+1}^t)-\widetilde{\nabla} f_i^t(\boldsymbol{x}_{i,\ell}^t)-(\nabla f_i^t(\boldsymbol{x}_{i,\ell+1}^t)-\nabla f_i^t(\boldsymbol{x}_{i,\ell}^t))]=0$. The first inequality follows the fact that $\|\boldsymbol{a}+\boldsymbol{b}\|^2\leq 2(\|\boldsymbol{a}\|^2+\|\boldsymbol{b}\|^2)$. The last inequality is due to the β-Lipschitz and $\widetilde{\beta}$-Lipschitz of ∇f_i^t and $\widetilde{\nabla} f_i^t$, respectively.

Moreover, by using equation 2.31, we obeserve

$$
\left\|\boldsymbol{x}_{i,\ell+1}^t-\boldsymbol{x}_{i,\ell}^t\right\|\leq\frac{3Cp+5AD}{\ell}
\tag{2.42}
$$

where in the last inequality, $\|\frac{1}{n}\sum_{j=1}^n \boldsymbol{v}_{j,\ell}^t-\overline{\boldsymbol{x}}_\ell^t\|\leq D$ for every t,ℓ since both $\frac{1}{n}\sum_{j=1}^n \boldsymbol{v}_{j,\ell}^t$ and $\overline{\boldsymbol{x}}_\ell^t$ are in $\mathcal{K}$.

Therefore, combining (2.41) and (2.42), we get

$$
\max_{1\leq j\leq n}\left\{\mathbb{E}\left\|\widetilde{\boldsymbol{d}}_{i,\ell+1}^t-\boldsymbol{d}_{i,\ell+1}^t\right\|^2\right\}\leq 2(\widetilde{\beta}^2+\beta^2)\frac{(3C_p+5AD)^2}{\ell^2}+\max_{1\leq j\leq n}\left\{\mathbb{E}\left\|\widetilde{\boldsymbol{d}}_{j,\ell}^t-\boldsymbol{d}_{j,\ell}^t\right\|^2\right\}
\tag{2.43}
$$

Applying (2.43) recursively on ℓ, we deduce that

$$
\begin{aligned}
\max_{1\leq j\leq n}\left\{\mathbb{E}\left\|\widetilde{\boldsymbol{d}}_{i,\ell+1}^t-\boldsymbol{d}_{i,\ell+1}^t\right\|^2\right\}&\leq 2(\widetilde{\beta}^2+\beta^2)(3C_p+5AD)^2\sum_{l=1}^{\ell}\frac{1}{l^2}+\max_{1\leq j\leq n}\left\{\mathbb{E}\left\|\widetilde{\boldsymbol{d}}_{j,1}^t-\boldsymbol{d}_{j,1}^t\right\|^2\right\}\\
&\leq 4(\widetilde{\beta}^2+\beta^2)(3C_p+5AD)^2+\max_{1\leq j\leq n}\left\{\mathbb{E}\left\|\widetilde{\boldsymbol{d}}_{j,1}^t-\boldsymbol{d}_{j,1}^t\right\|^2\right\}
\end{aligned}
$$

since $\sum_{l=1}^{\ell}\frac{1}{l^2}\leq\frac{\pi^2}{6}\leq 1.7$.

Besides, for any $1\leq i\leq n$

$$
\begin{aligned}
\max_{1\leq j\leq n}\left\{\mathbb{E}\left\|\widetilde{\boldsymbol{d}}_{j,1}^t-\boldsymbol{d}_{j,1}^t\right\|^2\right\}&=\max_{1\leq j\leq n}\left\{\mathbb{E}\left\|\sum_i \mathbf{W}_{ij}(\widetilde{\boldsymbol{g}}_{i,1}^t-\boldsymbol{g}_{i,1}^t)\right\|^2\right\}\\
&\leq\max_{1\leq j\leq n}\left\{\sum_i \mathbf{W}_{ij}\mathbb{E}\left\|\widetilde{\boldsymbol{g}}_{i,1}^t-\boldsymbol{g}_{i,1}^t\right\|^2\right\}\\
&=\max_{1\leq j\leq n}\left\{\sum_i \mathbf{W}_{ij}\mathbb{E}\left\|\widetilde{\nabla} f_i^t(\boldsymbol{x}_{i,1}^t)-\nabla f_j^t(\boldsymbol{x}_{i,1}^t)\right\|^2\right\}\leq\sigma^2
\end{aligned}
$$

since $\sum_j \mathbf{W}_{ij} = 1$.

Hence,

$$\mathbb{E}\left[\|\widetilde{\boldsymbol{d}}_{i,\ell+1}^t - \boldsymbol{d}_{i,\ell+1}^t\|^2\right] \leq 4(\widetilde{\beta}^2 + \beta^2)(3C_p + 5AD)^2 + \sigma^2.$$

□

Before presenting the proof of Theorem 3, we mention a useful recent variance reduction technique proposed by [Che+18], Stated as follows.

Lemma 5 ([Che+18], Theorem 3)**.** *Let $\{\boldsymbol{d}_\ell\}_{\ell\geq 1}$ be a sequence of points in $\mathbb{R}^n$ such that $\|\boldsymbol{d}_\ell - \boldsymbol{d}_{\ell-1}\| \leq {}^{B}/_{\ell+3}$ for all $\ell \geq 1$ with fixed constant $B \geq 0$. Let $\{\widetilde{\boldsymbol{d}}_\ell\}$ be a sequence of random variables such that $\mathbb{E}[\widetilde{\boldsymbol{d}}_\ell|\mathcal{H}_{\ell-1}] = \boldsymbol{d}_\ell$ and $\mathbb{E}\left[\left\|\widetilde{\boldsymbol{d}}_\ell - \boldsymbol{d}_\ell\right\|^2|\mathcal{H}_{\ell-1}\right] \leq \sigma^2$ for every $\ell \geq 1$, where $\mathcal{H}_{\ell-1}$ is the history up to $\ell - 1$. Let $\{\widetilde{\boldsymbol{a}}_\ell\}_{\ell\geq 0}$ be a sequence of random variables defined recursively as*

$$\widetilde{\boldsymbol{a}}_\ell = (1 - \rho_\ell)\widetilde{\boldsymbol{a}}_{\ell-1} + \rho^t\widetilde{\boldsymbol{d}}_\ell$$

For $\ell \geq 1$ where $\rho_\ell = \frac{2}{(\ell+3)^{2/3}}$ and $\widetilde{\boldsymbol{a}}_0$ is fixed. Then we have

$$\mathbb{E}\left[\left\|\widetilde{\boldsymbol{a}}_\ell - \boldsymbol{d}_\ell\right\|^2\right] \leq \frac{Q}{(\ell+4)^{2/3}},$$

where $Q = \max\left\{4\left\|\widetilde{\boldsymbol{a}}^0 - \boldsymbol{d}^0\right\|^2, 4\sigma^2 + \frac{3B^2}{2}\right\}$.

Theorem 3. *Let $\mathcal{K}$ be a convex set with diameter D. Assume that for every $1 \leq t \leq T$,*

1. *functions f_i^t are β-smooth, i.e. ∇f_i^t is β-Lipschitz, (so F^t is β-smooth);*
2. *$\|\nabla f_i^t\| \leq G$ (so $\|\nabla F^t\| \leq G$);*
3. *the gradient estimates are unbiased with bounded variance σ^2, i.e., $\mathbb{E}[\widetilde{\nabla} f_i^t(\boldsymbol{x}_{i,\ell}^t)] = \nabla f_i^t(\boldsymbol{x}_{i,\ell}^t)$ and $\left\|\widetilde{\nabla} f_i^t(\boldsymbol{x}_{i,\ell}^t)] - \nabla f_i^t(\boldsymbol{x}_{i,\ell}^t)\right\| \leq \sigma$ for every $1 \leq i \leq n$ and $1 \leq \ell \leq L$;*
4. *the gradient estimates are $\widetilde{\beta}$-Lipschitz.*

Then, choosing the step-sizes $\eta_\ell = \min\{1, {}^{A}/_{\ell}\}$ where $A = \max\{\frac{G}{\beta D}, \frac{(C_d+\beta C_p)}{2\beta D}, 3\}$ for all $1 \leq \ell \leq L$, Algorithm 1 guarantees that for every $1 \leq i \leq n$,

$$\sum_{t=1}^{T}\mathbb{E}[F^t(\boldsymbol{x}_i^t)] \leq \min_{\boldsymbol{x}\in\mathcal{K}}\sum_{t=1}^{T}F^t(\boldsymbol{x}) + \frac{2\beta AD^2T}{L} + 3(A+1)\cdot\mathcal{O}\left(\sqrt{T}\right) + \mathcal{O}\left(\frac{ATDQ^{1/2}}{L^{1/3}}\right)$$

where $Q = 6(\widetilde{\beta}^2 + \beta^2)(2C_p + AD)^2 + \sigma^2 + {}^{3B^2}/_2, B = 4C_d + 2\beta\left[2C_p + AD\right], C_p = \ell_0\sqrt{n}AD$ *and* $C_d = \ell_0\sqrt{n}\max\left\{|\lambda_2(\mathbf{W})|\left(\frac{\beta D}{1-|\lambda_2(\mathbf{W})|} + G\right), (6C_p + 10AD)\beta\right\} + 3\beta\left(C_p + 2AD\right)$.

Consequently, by choosing $L = T^{3/2}$*, we obtain the regret of* $\mathcal{O}\left(\sqrt{T}\right)$.

Proof. Recall that, in our notions, stochastic variables/parameters are denoted by the same letter as its exact counterpart with an additional tilde symbol. Let $\boldsymbol{x}^* \in \arg\min_{\boldsymbol{x}\in\mathcal{K}} \sum_{t=1}^T F^t(\boldsymbol{x})$. Based on the proof of Theorem 1, in particular Claim 3 over all $1 \leq t \leq T$, we get

$$
\begin{aligned}
\sum_{t=1}^T \left(F^t(\overline{\boldsymbol{x}}_{L+1}^t) - F^t(\boldsymbol{x}^*)\right) &\leq \frac{2\beta AD^2T}{L} + \sum_{t=1}^T \left(\frac{1}{n}\sum_{j=1}^n\sum_{\ell=1}^L \eta_\ell \left[\prod_{\ell'=\ell+1}^L (1-\eta_{\ell'})\right] \langle \boldsymbol{d}_{j,\ell}^t, \boldsymbol{v}_{j,\ell}^t - \boldsymbol{x}^*\rangle\right) \\
&= \frac{2\beta AD^2T}{L} + \frac{1}{n}\sum_{j=1}^n\sum_{\ell=1}^L \eta_\ell \left[\prod_{\ell'=\ell+1}^L (1-\eta_{\ell'})\right] \left(\sum_{t=1}^T \langle \widetilde{\boldsymbol{a}}_{j,\ell}^t, \boldsymbol{v}_{j,\ell}^t - \boldsymbol{x}^*\rangle\right) \\
&\quad + \frac{1}{n}\sum_{j=1}^n\sum_{\ell=1}^L \eta_\ell \left[\prod_{\ell'=\ell+1}^L (1-\eta_{\ell'})\right] \left(\sum_{t=1}^T \langle \boldsymbol{d}_{j,\ell}^t - \widetilde{\boldsymbol{a}}_{j,\ell}^t, \boldsymbol{v}_{j,\ell}^t - \boldsymbol{x}^*\rangle\right) \\
&\leq \frac{2\beta AD^2T}{L} + \frac{1}{n}\sum_{j=1}^n\sum_{\ell=1}^L \eta_\ell \left[\prod_{\ell'=\ell+1}^L (1-\eta_{\ell'})\right] \cdot \mathcal{R}^T \\
&\quad + \frac{1}{n}\sum_{j=1}^n\sum_{\ell=1}^L \eta_\ell \left[\prod_{\ell'=\ell+1}^L (1-\eta_{\ell'})\right] \left(\sum_{t=1}^T \langle \boldsymbol{d}_{j,\ell}^t - \widetilde{\boldsymbol{a}}_{j,\ell}^t, \boldsymbol{v}_{j,\ell}^t - \boldsymbol{x}^*\rangle\right) \\
&\leq \frac{2\beta AD^2T}{L} + 3(A+1)\cdot\mathcal{R}^T + \frac{1}{n}\sum_{j=1}^n\sum_{t=1}^T\sum_{\ell=1}^L \eta_\ell \left[\prod_{\ell'=\ell+1}^L (1-\eta_{\ell'})\right] \left(\langle \boldsymbol{d}_{j,\ell}^t - \widetilde{\boldsymbol{a}}_{j,\ell}^t, \boldsymbol{v}_{j,\ell}^t - \boldsymbol{x}^*\rangle\right) \\
&\leq \frac{2\beta AD^2T}{L} + 3(A+1)\cdot\mathcal{R}^T + \frac{1}{n}\sum_{j=1}^n\sum_{t=1}^T\sum_{\ell=1}^L \eta_\ell \left[\prod_{\ell'=\ell+1}^L (1-\eta_{\ell'})\right] \left(\frac{1}{\gamma_\ell}\|\boldsymbol{d}_{j,\ell}^t - \widetilde{\boldsymbol{a}}_{j,\ell}^t\|^2 + \gamma_\ell\|\boldsymbol{v}_{j,\ell}^t - \boldsymbol{x}^*\|^2\right) \\
&\qquad \text{(Young's Inequality)} \\
&\leq \frac{2\beta AD^2T}{L} + 3(A+1)\cdot\mathcal{R}^T + \frac{1}{n}\sum_{j=1}^n\sum_{t=1}^T\sum_{\ell=1}^L \eta_\ell \left[\prod_{\ell'=\ell+1}^L (1-\eta_{\ell'})\right] \left(\frac{1}{\gamma_\ell}\|\boldsymbol{d}_{j,\ell}^t - \widetilde{\boldsymbol{a}}_{j,\ell}^t\|^2 + \gamma_\ell D^2\right)
\end{aligned} \tag{2.44}
$$

The second inequality follows the fact that online linear oracles has regret $\mathcal{R}^T$. The third inequality holds by Claim 4.

Claim 5. *Let* $\gamma_\ell = \frac{Q^{1/2}}{D(\ell+3)^{1/3}}$ *for* $1 \le \ell \le L$ *where* $Q = 6(\widetilde{\beta}^2 + \beta^2)(2C_p + AD)^2 + \sigma^2 + 3B^2/2$*, it holds that*

$$\mathbb{E}\left[\sum_{\ell=1}^{L} \eta_\ell \left[\prod_{\ell'=\ell+1}^{L} (1-\eta_{\ell'})\right] \left(\frac{1}{\gamma_\ell}\|\boldsymbol{d}_{j,\ell}^t - \widetilde{\boldsymbol{a}}_{j,\ell}^t\|^2 + \gamma_\ell D^2\right)\right] \le \frac{\mathcal{O}\left(ADQ^{1/2}\right)}{L^{1/3}}.$$

Proof of claim. Fix a time step t. We aim to apply Lemma 5. We first verify the conditions in that lemma: $\mathbb{E}[\widetilde{\boldsymbol{d}}_{j,\ell}^t] = \boldsymbol{d}_{j,\ell}^t$, by Lemma 4, $\mathbb{E}\left[\|\widetilde{\boldsymbol{d}}_{i,\ell}^t - \boldsymbol{d}_{i,\ell}^t\|^2\right] \le 6(\widetilde{\beta}^2 + \beta^2)(2C_p + AD)^2 + \sigma^2$, and by Claim 2, $\left\|\boldsymbol{d}_{i,\ell}^t - \boldsymbol{d}_{i,\ell-1}^t\right\| \le \frac{B}{\ell+3}$. Hence, applying Lemma 5 (recall that $\rho_\ell = \frac{2}{(\ell+3)^{2/3}}$ for $1 \le \ell \le L$), we have

$$\mathbb{E}[\|\boldsymbol{d}_{j,\ell}^t - \widetilde{\boldsymbol{a}}_{j,\ell}^t\|^2] \le \frac{Q}{(\ell+4)^{2/3}} \tag{2.45}$$

where $Q = 6(\widetilde{\beta}^2 + \beta^2)(3C_p + 5AD)^2 + \sigma^2 + 3B^2/2$.

Hence,

$$\begin{aligned}
&\mathbb{E}\left[\sum_{\ell=1}^{L}\eta_\ell \left[\prod_{\ell'=\ell+1}^{L} (1-\eta_{\ell'})\right] \left(\frac{1}{\gamma_\ell}\|\boldsymbol{d}_{j,\ell}^t - \widetilde{\boldsymbol{a}}_{j,\ell}^t\|^2 + \gamma_\ell\|\boldsymbol{v}_{j,\ell}^t - \boldsymbol{x}^*\|^2\right)\right] \\
&\quad\le \sum_{\ell=1}^{L} \eta_\ell \left[\prod_{\ell'=\ell+1}^{L} (1-\eta_{\ell'})\right] \left(\frac{1}{\gamma_\ell}\cdot\frac{Q}{(\ell+4)^{2/3}} + \gamma_\ell D^2\right) \\
&\quad\le \sum_{\ell=1}^{L-1} \frac{A}{\ell}\left(\frac{\ell+2}{L}\right)^A \left(\frac{1}{\gamma_\ell}\cdot\frac{Q}{(\ell+4)^{2/3}} + \gamma_\ell D^2\right) + \eta_L\left(\frac{1}{\gamma_L}\cdot\frac{Q}{(L+4)^{2/3}} + \gamma_L D^2\right) \\
&\quad= \sum_{\ell=1}^{L-1} \frac{A}{L}\cdot\frac{\ell+2}{\ell}\cdot\left(\frac{\ell+2}{L}\right)^{A-1} \left(\frac{1}{\gamma_\ell}\cdot\frac{Q}{(\ell+4)^{2/3}} + \gamma_\ell D^2\right) + \eta_L\left(\frac{1}{\gamma_L}\cdot\frac{Q}{(L+4)^{2/3}} + \gamma_L D^2\right)
\end{aligned}$$

where in the second inequality we use Inequality (2.40), specifically

$$\prod_{\ell'=\ell+1}^{L} (1-\eta_{\ell'}) \le e^{-A\sum_{\ell'=\ell+1}^{L} 1/\ell'} \le e^{-A\int_{\ell+2}^{L} d\ell'/\ell'} \le e^{-A(\ln L - \ln(\ell+2))} = \left(\frac{\ell+2}{L}\right)^A.$$

Choosing $\gamma_\ell = \frac{Q^{1/2}}{D(\ell+3)^{1/3}}$, we obtain:

$$\mathbb{E}\left[\sum_{\ell=1}^{L}\eta_\ell \left[\prod_{\ell'=\ell+1}^{L} (1-\eta_{\ell'})\right] \left(\frac{1}{\gamma_\ell}\|\boldsymbol{d}_{j,\ell}^t - \widetilde{\boldsymbol{a}}_{j,\ell}^t\|^2 + \gamma_\ell D^2\right)\right]$$

$$\leq \sum_{\ell=1}^{L-1} \frac{A}{L} \cdot \frac{\ell+2}{\ell} \cdot \left(\frac{\ell+2}{L}\right)^{A-1} \frac{2DQ^{1/2}}{(\ell+3)^{1/3}} + \frac{A}{L} \cdot \frac{2DQ^{1/2}}{(L+3)^{1/3}}$$
$$\leq \frac{1}{L} \cdot \mathcal{O}\left(ADQ^{1/2}\right) \sum_{\ell=1}^{L} \frac{1}{\ell^{1/3}} \leq \frac{1}{L} \cdot \mathcal{O}\left(ADQ^{1/2}\right) \int_{\ell=1}^{L} \frac{d\ell}{\ell^{1/3}} \leq \frac{1}{L} \cdot \mathcal{O}\left(ADQ^{1/2}\right) \cdot L^{2/3}.$$

where the second inequality holds since $A \geq 3$. The claim follows. □ Combining Claim 5 with the Inequality (2.44), we have:

$$\sum_{t=1}^{T} \left(F^t(\overline{\boldsymbol{x}}_{L+1}^t) - F^t(\boldsymbol{x}^*)\right) \leq \frac{2\beta AD^2T}{L} + 3(A+1) \cdot \mathcal{R}^T + \mathcal{O}\left(\frac{ATDQ^{1/2}}{L^{1/3}}\right).$$

Using Lemma 2 together with the fact that F^t is Lipschitz, for every $1 \leq i \leq n$, we have

$$\begin{aligned}
\sum_{t=1}^{T} \left(F^t(\boldsymbol{x}_i^t) - F^t(\boldsymbol{x}^*)\right) &= \sum_{t=1}^{T} \left(F^t(\boldsymbol{x}_{i,L+1}^t) - F^t(\boldsymbol{x}^*)\right) \\
&= \sum_{t=1}^{T} \left(F^t(\overline{\boldsymbol{x}}_{L+1}^t) - F^t(\boldsymbol{x}^*)\right) + \sum_{t=1}^{T} \left(F^t(\boldsymbol{x}_{i,L+1}^t) - F^t(\overline{\boldsymbol{x}}_{L+1}^t)\right) \\
&\leq \frac{2\beta AD^2T}{L} + 3(A+1) \cdot \mathcal{R}^T + \mathcal{O}\left(\frac{ATDQ^{1/2}}{L^{1/3}}\right) + \sum_{t=1}^{T} G\|\boldsymbol{x}_{i,L+1}^t - \overline{\boldsymbol{x}}_{L+1}^t\| \\
&\leq \frac{2\beta AD^2T}{L} + 3(A+1) \cdot \mathcal{R}^T + \mathcal{O}\left(\frac{ATDQ^{1/2}}{L^{1/3}}\right) \\
&\quad + \sum_{t=1}^{T} G(1-\eta_L) \left\|\boldsymbol{y}_{i,L+1}^t - \overline{\boldsymbol{x}}_{L+1}^t\right\| + \eta_L \left\|\boldsymbol{v}_{i,L}^t - \overline{\boldsymbol{v}}_L^t\right\| \\
&\leq \frac{2\beta AD^2T}{L} + 3(A+1) \cdot \mathcal{R}^T + \mathcal{O}\left(\frac{ATDQ^{1/2}}{L^{1/3}}\right) \\
&\quad + \sum_{t=1}^{T} G \left\|\boldsymbol{y}_{i,L+1}^t - \overline{\boldsymbol{x}}_{L+1}^t\right\| + \eta_L \left\|\boldsymbol{v}_{i,L}^t - \overline{\boldsymbol{v}}_L^t\right\| \\
&\leq \frac{2\beta AD^2T}{L} + 3(A+1) \cdot \mathcal{R}^T + \mathcal{O}\left(\frac{ATDQ^{1/2}}{L^{1/3}}\right) + \frac{GT(C_p + AD)}{L}.
\end{aligned}$$

As the chosen oracles has regret $\mathcal{R}^T = \mathcal{O}\left(\sqrt{T}\right)$ and $L = T^{3/2}$, the theorem follows. □

3 Meta Frank Wolfe on a Random Walk Journey

3.1 Introduction

In chapter 2, we present a decentralized version of Meta Frank Wolfe that requires synchronized communications between neighbours. This chapter investigates Markov chain Meta Frank Wolfe, a version of Meta Frank Wolfe in which random samples are taken along the trajectory of a Markov chain. Recall that, we are looking for a decentralized online algorithm that can learn the model without requiring the participation of a centralized body and in which nodes only share information with their immediate neighbors. To accomplish this goal, we devised a random walk for each online round t that, at each node it visits, triggers an update to the global model based on the local data of the triggered node.

3.2 Uniform Random Walk Meta Frank Wolfe

The method starts at one node that is chosen evenly at random to be activated at the first iteration in order to carry out the update. This node is the starting point for the algorithm. After that, during each iteration k, the global model iterate $\boldsymbol{x}_k^t$ is given to a node i^{k+1} that is picked at random in order for that node to update the model with the data that is stored locally and obtain the updated model $\boldsymbol{x}_{k+1}^t$. In light of the fact that connections must be maintained at all times, the currently activated node i^k must be a neighbor of the node i^{k-1} that came before it in the activation process, more formally $i^k \in \mathcal{N}(i^{k-1})$. A random walk is defined by the mechanism described above, and we model it using a Markov chain. The sequence of active nodes $i^k \in \mathcal{V}$ represents the states of the Markov chain with the transition matrix $\mathbf{P}$; this structure is passed down from the underlying linked graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, which has the same connection structure. We construct the uniform transition matrix $\mathbf{P}$ using the Metropolis-Hastings [Has70]:

$$\mathbf{P}_{ij} = \begin{cases} 1/(1+\max\{d_i, d_j\}) & \text{if } (i,j) \in E, \\ 0 & \text{if } (i,j) \notin E \text{ and } i \neq j, \\ 1 - \sum_{j \in N(i)} \mathbf{P}_{ij} & \text{if } i = j, \end{cases}$$

In order to ensure that there is convergence, we assume that the Markov chain $\left(i^k\right)_{k\in\mathbb{N}}$ defined on the finite space $\mathcal{V}$ with homogeneous transition matrix $\mathbf{P}$ is irreducible and aperiodic and has a stationary distribution $\boldsymbol{\pi}^*$ on $\mathcal{V}$.

The formal description is given in Algorithm 5.

Algorithm 5 Random Walk Meta Frank-Wolfe algorithm (RWMFW)

Initialization: Initial node $i^{(1)}$ chosen uniformly at random for $[N]$, Initial model $\boldsymbol{x}_1^1$ chosen uniformly at random from $\mathcal{K}$

1: **for** $t = 1$ to T **do** ▷ The variable t represents rounds
2: Initialize $\boldsymbol{x}_1^t$
3: **for** $k = 1$ to K **do** ▷ The variable k represents iterations
4: $\boldsymbol{v}_k^t \leftarrow$ output of the online oracle O_k
5: $\boldsymbol{x}_{k+1}^t \leftarrow \boldsymbol{x}_k^t + \eta_k(\boldsymbol{v}_k^t - \boldsymbol{x}_k^t)$
6: Feedback $\langle \nabla f_{i^k}^t(\boldsymbol{x}_k^t), \cdot \rangle$ to the oracle O_k ▷ This step is done once the $f_{i^k}^t$ is revealed to node i^k: after the step 15. See Remark 4
7: Choose node i^{k+1} with probability $p \sim \mathbf{P}_{i^k,*}$
8: ▷ $\mathbf{P}_{i^k,*}$ = "row i^k of $\mathbf{P}$" which defines a probability distribution
9: Send $\boldsymbol{x}_{k+1}^t$ to node i^{k+1}
10: $\hat{\boldsymbol{x}}_{k+1}^t \leftarrow \hat{\boldsymbol{x}}_k^t + \eta_k \boldsymbol{x}_{k+1}^t$
11: Send $\hat{\boldsymbol{x}}_k^t$ to node i^{k+1}
12: Send the oracle $O_1, \cdots, O_k$ updates to node i^{k+1} ▷ See Remark 5
13: **end for**
14: $\overline{\boldsymbol{x}}_{K+1}^t \leftarrow \frac{\hat{\boldsymbol{x}}_{K+1}^t}{\sum_{k=1}^{K+1} \eta_k}$ ▷ $\overline{\boldsymbol{x}}_{K+1}^t = \frac{\sum_{k=1}^{K+1} \eta_k \boldsymbol{x}_k^t}{\sum_{k=1}^{K+1} \eta_k}$
15: Every node plays $\overline{\boldsymbol{x}}_{K+1}^t$
16: **end for**

Remark 4: (The feedback step 6 in Algorithm 5).

The feedback to the oracles can only occur once the decision at time t has been committed (step 15) and the feedback function f_i^t is revealed to each agent. This means that once the walk has ended, each node provides the oracles with the agreed upon feedback.

Remark 5: (Shared oracles).

Furthermore, the current iteration of the algorithm requires that all agents must have access to the same set of oracles, which is not ideal in decentralized settings. This means that each agent must pass on the updates of the oracles to the next agent in the walk. However, if agents have shared memory access to the oracles, this becomes more efficient. Our research aims to fully decentralize the process, and therefore, we aim to modify the algorithm so that each agent has their own private set of oracles. This is not currently possible, but our work represents a step towards achieving this goal.

Remark 6.

The unconventional form of the final decision, $\overline{\boldsymbol{x}}_{K+1}^t$, arises from the analytical technique that draws heavily from the analysis of Markov Chain Gradient Descent (MCGD) in offline settings, as presented in [SSY18]. The primary challenge in this work is to adapt this technique to the online setting by utilizing the meta-actions from the Meta Frank-Wolfe algorithm.

Theorem 4. *Assume that for every round* $1 \leq t \leq T$

1. *the local loss function* f_i^t *for each node* $i \in \mathcal{V}$ *is a convex function on* $\mathcal{K}$, β*-smooth, and* G*-Liptschitz continuous*
2. *the markov chain* $\left(i^k\right)_{k\in\mathbb{N}}$ *defined on the finite state space* $\mathcal{V}$ *with homogeneous transition matrix* $\mathbf{P}$ *is irreducible and aperiodic and has stationary distribution* $\boldsymbol{\pi}^*$ *on* $\mathcal{V}$
3. *the online linear optimization oracles have regret* $\mathcal{R}^T$

Then, with the step-size $\eta_k = 1/k^q$ *with* $q \in [1/2, 1)$ *and the number of iterations* $K \geq K_P$ *where* K_P *is a constant*[1] *that depends on* $\lambda(\mathbf{P})$ *and* $\lambda_2(\mathbf{P})$*, the Algorithm 5 yields in an online fashion a sequence of decisions* $\left\{\overline{\boldsymbol{x}}_K^t\right\}_{t\in[T]}$ *with regret*

$$\sum_{t=1}^{T} \mathbb{E}\left[F^t\left(\overline{\boldsymbol{x}}_K^t\right)\right] - \min_{\boldsymbol{x}^*\in\mathcal{K}} \sum_{t=1}^{T} F^t\left(\boldsymbol{x}^*\right) \in \mathcal{O}\left(\frac{T}{K^{1-q}} + \mathcal{R}^T\right) \tag{3.1}$$

where $\overline{\boldsymbol{x}}_K^t = \frac{1}{\sum_{\ell=1}^K \eta_\ell} \sum_{\ell=1}^K \eta_\ell \boldsymbol{x}_\ell^t$.

Moreover, if we take $K \in \mathcal{O}\left(\left(\sqrt{T}\right)^{\frac{1}{1-q}}\right)$ *and if* $\mathcal{R}^T \in \mathcal{O}\left(\sqrt{T}\right)$*, then the algorithm's regret is sublinear, in particular :*

$$\sum_{t=1}^{T} \mathbb{E}\left[F^t\left(\overline{\boldsymbol{x}}_K^t\right)\right] - \min_{\boldsymbol{x}^*\in\mathcal{K}} \sum_{t=1}^{T} F^t\left(\boldsymbol{x}^*\right) \in \mathcal{O}\left(\sqrt{T}\right) \tag{3.2}$$

To get the smallest value for K*, one should choose* $q = \frac{1}{2}$ *which implies* $K \in \mathcal{O}(T)$.

Theorem 4 is a result about the performance of Algorithm 5, in terms of its regret. Recall that the regret of an optimization algorithm is defined as the difference between the value of the function F^t at the solution chosen by the algorithm at each time step t and the value of the optimal solution that could have been achieved with perfect knowledge of the problem. The goal of the optimization process is to find a sequence of solutions that minimizes the cumulative regret over all time steps.

[1] For further details on K_P, see the proof of Lemma 1 in [SSY18] (stated as Lemma 6 in this chapter).

The theorem states that under certain assumptions about the local loss functions, the markov chain, and the online linear optimization oracles being used, the Algorithm 5 can achieve a regret bound of the form $\mathcal{O}\left(\frac{T}{K^{1-q}} + \mathcal{R}^T\right)$, where K is the number of iterations per round and $\mathcal{R}^T$ is the regret of the online linear optimization oracles. This means that the cumulative regret of the algorithm decreases at a rate of $\mathcal{O}\left(\frac{T}{K^{1-q}}\right)$ as the number of time steps T increases, provided that K is chosen appropriately.

The theorem also states that if K is chosen to be of the form $\mathcal{O}\left(\left(\sqrt{T}\right)^{\frac{1}{1-q}}\right)$ and if the regret of the online linear optimization oracles is $\mathcal{O}\left(\sqrt{T}\right)$, then the algorithm's regret is sublinear, meaning that it decreases at a rate faster than $\mathcal{O}(T)$ as the number of time steps T increases. In this case, the cumulative regret of the algorithm is bounded by $\mathcal{O}\left(\sqrt{T}\right)$, which means that the per-round regret decreases at a rate of $\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$ as the number of time steps increases.

One of the assumptions made in the theorem is that the local loss functions f_i^t for each node $i \in \mathcal{V}$ are convex, β-smooth, and G-Lipschitz continuous. This assumption is often made in the analysis of optimization algorithms because it allows for the use of tools from convex analysis to study the properties of the optimization problem.

The assumption that the Markov chain $\left(i^k\right)_{k\in\mathbb{N}}$ is irreducible and aperiodic and has a stationary distribution $\boldsymbol{\pi}^*$ on $\mathcal{V}$ is used to ensure that the algorithm is able to explore the entire space of solutions and that it is able to find good solutions even if it starts in a suboptimal state.

The eigenvalues of the transition matrix $\mathbf{P}$ can be used to measure the convergence rate of the markov chain and to obtain a more precise bound on the regret of the Algorithm 5. For example, if the transition matrix $\mathbf{P}$ has a small eigenvalue, it may indicate that the Markov chain converges slowly, which could lead to a larger regret for the algorithm. On the other hand, if the transition matrix $\mathbf{P}$ has a large eigenvalue, it may indicate that the Markov chain converges quickly, which could lead to a smaller regret for the algorithm. The analysis of the theorem provides the dependence to $\lambda(\mathbf{P})$ defined in Lemma 6. Hence, we obtain :

$$\sum_{t=1}^{T} \mathbb{E}\left[F^t\left(\overline{\boldsymbol{x}}_K^t\right)\right] - \min_{\boldsymbol{x}^*\in\mathcal{K}} \sum_{t=1}^{T} F^t\left(\boldsymbol{x}^*\right) \in \mathcal{O}\left(\ln\left(\frac{1}{\lambda(\mathbf{P})}\right)\sqrt{T}\right) \tag{3.3}$$

3.3 Proof of Theorem 4

Lemma 6 (Mixing Time, see Lemma 1 in [SSY18])**.** *Assume that the Markov chain $(X_k)_{k\geq 0}$ is time-homogeneous, irreducible, and aperiodic with the transition matrix $\mathbf{P} \in \mathbb{R}^{n\times n}$ and stationary distribution $\boldsymbol{\pi}^*$. Let $\lambda_i(\mathbf{P})$ be the ith largest eigenvalue of $\mathbf{P}$, and*

$$\lambda(\mathbf{P}) = \frac{\max\left\{|\lambda_2(\mathbf{P})|, |\lambda_n(\mathbf{P})|\right\} + 1}{2} \in [0, 1)$$

Then, we can bound the largest entry-wise absolute value of deviation matrix $\delta^k = \mathbf{\Pi}^* - \mathbf{P}^k \in \mathbb{R}^{n\times n}$ as

$$\left\|\delta^k\right\|_\infty \leq C_P \cdot \lambda^k(\mathbf{P}) \tag{3.4}$$

for $k \geq K_P$, where C_P is a constant that depends on the Jordan canonical form of $\mathbf{P}$, and K_P is a constant that depends on $\lambda(\mathbf{P})$ and $\lambda_2(\mathbf{P})$.

Claim 6. *Let f_i^t a G-Lipschitz convex function, and $(\boldsymbol{x}_i^t, \boldsymbol{x}^*) \in \mathcal{K}^2$, then*

$$\left|f_i^t(\boldsymbol{x}_i^t) - f_i^t(\boldsymbol{x}^*)\right| \leq H = GD$$

Proof.

$$\begin{aligned} f_i^t(\boldsymbol{x}_i^t) - f_i^t(\boldsymbol{x}^*) &\leq G\left\|\boldsymbol{x}_i^t - \boldsymbol{x}^*\right\| && (G\text{-Lipschitz}) \\ &\leq GD \\ &\text{(Both } \boldsymbol{x}_i^t \text{ and } \boldsymbol{x}^* \text{ are in the bounded constrained set } \mathcal{K} \Rightarrow \left\|\boldsymbol{x}_i^t - \boldsymbol{x}^*\right\| \leq D) \end{aligned}$$

□

Lemma 7. *Let $(\boldsymbol{x}_k^t)_{k\geq 0}$ be generated by Algorithm 5. For any $\boldsymbol{x}^*$ being the minimizer of F^t constrained on $\mathcal{K}$, and $i \in [n]$, and $\forall k, t \in \mathbb{N}^2$, then*

$$\left|f_i^t(\boldsymbol{x}_k^t) - f_i^t(\boldsymbol{x}^*)\right| \leq \mathcal{H}_k$$

with $\mathcal{H}_k = \frac{H}{k^q} + \frac{\beta D^2}{2k} + \frac{1}{k^q}\sum_{\ell=1}^k \langle \nabla f_{j^\ell}^t(\boldsymbol{x}_\ell^t), \boldsymbol{v}_\ell^t - \boldsymbol{x}^*\rangle$

Proof of Lemma 7.

$$f_j^t(\boldsymbol{x}_{k+1}^t) - f_j^t(\boldsymbol{x}^*) = f_j^t(\boldsymbol{x}_k^t + \eta_k(\boldsymbol{v}_k^t - \boldsymbol{x}_k^t)) - f_j^t(\boldsymbol{x}^*) \qquad \text{(By Algorithm 5 line 5)}$$

$$\leq f_j^t(\boldsymbol{x}_k^t) - f_j^t(\boldsymbol{x}^*) + \eta_k \langle \nabla f_j^t(\boldsymbol{x}_k^t), \boldsymbol{v}_k^t - \boldsymbol{x}_k^t \rangle + \eta_k^2 \frac{\beta}{2} \left\| \boldsymbol{v}_k^t - \boldsymbol{x}_k^t \right\|^2$$

(β-smoothness)

$$\leq f_j^t(\boldsymbol{x}_k^t) - f_j^t(\boldsymbol{x}^*) + \eta_k \langle \nabla f_j^t(\boldsymbol{x}_k^t), \boldsymbol{v}_k^t - \boldsymbol{x}_k^t \rangle + \eta_k^2 \frac{\beta D^2}{2}$$

($\mathcal{K}$ is bounded with diameter D)

Observe that we can decompose $\langle \nabla f_j^t(\boldsymbol{x}_k), \boldsymbol{v}_k^t - \boldsymbol{x}_k \rangle$ as

$$\langle \nabla f_j^t(\boldsymbol{x}_k^t), \boldsymbol{v}_k^t - \boldsymbol{x}_k \rangle = \langle \nabla f_j^t(\boldsymbol{x}_k), \boldsymbol{v}_k^t - \boldsymbol{x}^* \rangle + \langle \nabla f_j^t(\boldsymbol{x}_k), \boldsymbol{x}^* - \boldsymbol{x}_k^t \rangle$$

Since $\langle \nabla f_j^t(\boldsymbol{x}_k^t), \boldsymbol{x}^* - \boldsymbol{x}_k^t \rangle \leq f_j^t(\boldsymbol{x}^*) - f_j^t(\boldsymbol{x}_k^t)$, we get

$$\langle \nabla f_j^t(\boldsymbol{x}_k^t), \boldsymbol{v}_k^t - \boldsymbol{x}_k^t \rangle \leq f_j^t(\boldsymbol{x}^*) - f_j^t(\boldsymbol{x}_k^t) + \langle \nabla f_j^t(\boldsymbol{x}_k^t), \boldsymbol{v}_k^t - \boldsymbol{x}^* \rangle$$

Combining all the inequalities, therefore

$$f_j^t(\boldsymbol{x}_{k+1}^t) - f_j^t(\boldsymbol{x}^*) \leq (1 - \eta_k)(f_j^t(\boldsymbol{x}_k^t) - f_j^t(\boldsymbol{x}^*)) + \eta_k \langle \nabla f_j^t(\boldsymbol{x}_k^t), \boldsymbol{v}_k^t - \boldsymbol{x}^* \rangle + \eta_k^{2q} \frac{\beta D^2}{2} \tag{3.5}$$

Applying recursively on k the Inequality (3.5). We have

$$\begin{aligned}
f_j^t(\boldsymbol{x}_{k+1}^t) - f_j^t(\boldsymbol{x}^*) &\leq (f_j^t(\boldsymbol{x}_1^t) - f_j^t(\boldsymbol{x}^*)) \left[\prod_{\ell=1}^{k} (1 - \eta_\ell) \right] \\
&\quad + \sum_{\ell=1}^{k} \eta_\ell \prod_{m=\ell+1}^{k} (1 - \eta_m) \left[\langle \nabla f_{i^\ell}^t(\boldsymbol{x}_\ell^t), \boldsymbol{v}_\ell^t - \boldsymbol{x}^* \rangle + \frac{\beta D^2}{2} \eta_\ell \right] \\
&\leq H \left[\prod_{\ell=1}^{k} (1 - \eta_\ell) \right] + \sum_{\ell=1}^{k} \eta_\ell \prod_{m=\ell+1}^{k} (1 - \eta_m) \left[\langle \nabla f_{i^\ell}^t(\boldsymbol{x}_\ell^t), \boldsymbol{v}_\ell^t - \boldsymbol{x}^* \rangle + \frac{\beta D^2}{2} \eta_\ell \right] \\
&\qquad \text{(By Claim 6)} \\
&\overset{(a)}{\leq} \frac{H}{k^q} + \sum_{\ell=1}^{k} \frac{1}{\ell^q} \frac{\ell^q}{k^q} \left[\langle \nabla f_{i^\ell}^t(\boldsymbol{x}_\ell^t), \boldsymbol{v}_\ell^t - \boldsymbol{x}^* \rangle + \frac{\beta D^2}{2} \frac{1}{\ell^q} \right] \\
&\leq \frac{H}{k^q} + \frac{1}{k^q} \sum_{\ell=1}^{k} \langle \nabla f_{i^\ell}^t(\boldsymbol{x}_\ell^t), \boldsymbol{v}_\ell^t - \boldsymbol{x}^* \rangle + \frac{1}{k^q} \sum_{\ell=1}^{k} \frac{\beta D^2}{2\ell^q} \\
&\leq \frac{H}{k^q} + \frac{\beta D^2}{2k^q k^{(1-q)}} + \frac{1}{k^q} \sum_{\ell=1}^{k} \langle \nabla f_{i^\ell}^t(\boldsymbol{x}_\ell^t), \boldsymbol{v}_\ell^t - \boldsymbol{x}^* \rangle \qquad \left(\textstyle\sum_{\ell=1}^{k} \frac{1}{\ell^q} = \mathcal{O}\left(\frac{1}{k^{(1-q)}}\right)\right) \\
&\leq \frac{H}{k^q} + \frac{\beta D^2}{2k} + \frac{1}{k^q} \sum_{\ell=1}^{k} \langle \nabla f_{i^\ell}^t(\boldsymbol{x}_\ell^t), \boldsymbol{v}_\ell^t - \boldsymbol{x}^* \rangle
\end{aligned}$$

(a) follows from the fact that

$$(1 - \frac{1}{m^q}) \leq (1 - \frac{1}{m}) \qquad \text{(Because } q < 1\text{)}$$

$$\prod_{m=\ell+1}^{k}(1-\frac{1}{m^q}) \leq \prod_{m=\ell+1}^{k}(1-\frac{1}{m})$$

$$\prod_{m=\ell+1}^{k}(1-\frac{1}{m^q}) \leq \frac{\ell}{k} \qquad (\prod_{m=\ell+1}^{k}(1-\tfrac{1}{m}) = \tfrac{\ell}{k})$$

$$\prod_{m=\ell+1}^{k}(1-\frac{1}{m^q}) \leq \left(\frac{\ell}{k}\right)^q \qquad (\tfrac{\ell}{k} \leq 1 \text{ and } q<1 \Rightarrow \left(\tfrac{\ell}{k}\right)^q \geq \tfrac{\ell}{k})$$

□

Recall that in order to optimize $F^t = \frac{1}{n}\sum_{i=1}^{n} f_i^t$, the algorithm proceeds by doing a random walk while updating locally $f_{i^k}^t$ (the local function of node i^k for iteration k. Therefore, to analyse the convergence, we will bound the following :

$$\begin{aligned}
\sum_k \eta_k \mathbb{E}\left[f_{i^k}^t(\boldsymbol{x}_{k+1}^t) - f_{i^k}^t(\boldsymbol{x}^*)\right] &\leq \sum_k \eta_k \mathcal{H}_k && \text{(By lemma 7)}\\
&\leq \sum_k \frac{1}{k^{2q}} \sum_{\ell=1}^{k} \langle \nabla f_{j^\ell}^t(\boldsymbol{x}_\ell^t), \boldsymbol{v}_\ell^t - \boldsymbol{x}^*\rangle + \frac{H}{k^{2q}} + \frac{\beta D^2}{2k} \\
&\text{(By definition of } \eta_k = {}^1\!/_{k^q} \text{ and } \mathcal{H}_k = \tfrac{H}{k^q} + \tfrac{\beta D^2}{2k} + \tfrac{1}{k^q}\textstyle\sum_{\ell=1}^{k}\langle \nabla f_{j^\ell}^t(\boldsymbol{x}_\ell^t), \boldsymbol{v}_\ell^t - \boldsymbol{x}^*\rangle) \\
&\leq \zeta(2q)H + \sum_k \frac{\beta D^2}{2k^2} + \sum_k \frac{1}{k^{2q}} \sum_{\ell=1}^{k} \langle \nabla f_{j^\ell}^t(\boldsymbol{x}_\ell^t), \boldsymbol{v}_\ell^t - \boldsymbol{x}^*\rangle \\
&\qquad (\zeta \text{ being the Riemann Zeta Function}) \\
&= \zeta(2q)H + \frac{\beta D^2 \pi^2}{12} + \sum_k \frac{1}{k^{2q}} \sum_{\ell=1}^{k} \langle \nabla f_{j^\ell}^t(\boldsymbol{x}_\ell^t), \boldsymbol{v}_\ell^t - \boldsymbol{x}^*\rangle
\end{aligned}$$

Thus,

$$\sum_k \eta_k \mathbb{E}\left[f_{i^k}^t(\boldsymbol{x}_{k+1}^t) - f_{i^k}^t(\boldsymbol{x}^*)\right] \leq \zeta(2q)H + \frac{\beta D^2 \pi^2}{12} + \sum_k \frac{1}{k^{2q}} \sum_{\ell=1}^{k} \langle \nabla f_{j^\ell}^t(\boldsymbol{x}_\ell^t), \boldsymbol{v}_\ell^t - \boldsymbol{x}^*\rangle \tag{3.6}$$

In our analysis, we will make use of $\mathcal{T}_k$ defined as follows

$$\mathcal{T}_k = \min\left\{\max\left\{\left\lceil \frac{\ln\left(2C_P H \cdot k\right)}{\ln\left({}^1\!/_{\lambda(\mathbf{P})}\right)}\right\rceil, K_P\right\}, k\right\} \tag{3.7}$$

with H a constant such that $H \geq |f_i^t(\boldsymbol{x}_k^t) - f_i^t(\boldsymbol{x}^*)|$. Specifically, by Claim 6 the value of H is $H = GD$.

Claim 7. *With $\mathcal{T}_k$ defined in (3.7), for all $k \geq K_P$ and for all $i, j \in \mathcal{V}^2$, the following inequality holds*

$$\frac{f_i^t(\boldsymbol{x}_{k-\mathcal{T}_k}^t) - f_i^t(\boldsymbol{x}^*)}{n} - \frac{1}{2k} \leq \left(f_i^t(\boldsymbol{x}_{k-\mathcal{T}_k}^t) - f_i^t(\boldsymbol{x}^*)\right)\left[\mathbf{P}^{\mathcal{T}_k}\right]_{i,j} \leq \frac{f_i^t(\boldsymbol{x}_{k-\mathcal{T}_k}^t) - f_i^t(\boldsymbol{x}^*)}{n} + \frac{1}{2k} \quad (3.8)$$

Proof. Notice that $\mathcal{T}_k$ (3.7) is defined using constants (C_P, K_P and $\lambda(\mathbf{P})$) derived from the transition matrix $\mathbf{P}$ and H which upper-bounds $f_i^t(\boldsymbol{x}_{k+1}^t) - f_i^t(\boldsymbol{x}^*)$ for all $i \in [n]$. Therefore, by using the Mixing Time lemma 6, we get

$$\begin{aligned}
\left|\left[\mathbf{P}^{\mathcal{T}_k}\right]_{i,j} - \frac{1}{n}\right| &\leq C_P \cdot (\lambda(\mathbf{P}))^{\mathcal{T}_k} && \text{(By lemma 6, equation 3.4)} \\
&= C_P \cdot (\lambda(\mathbf{P}))^{\min\left\{\max\left\{\left\lceil \frac{\ln(2C_P H \cdot k)}{\ln(1/\lambda(\mathbf{P}))}\right\rceil, K_P\right\}, k\right\}} && \text{(By equation 3.7)} \\
&\leq C_P \cdot (\lambda(\mathbf{P}))^{\frac{\ln(2C_P H \cdot k)}{\ln(1/\lambda(\mathbf{P}))}} \\
\ln\left(\left|\left[\mathbf{P}^{\mathcal{T}_k}\right]_{i,j} - \frac{1}{n}\right|\right) &\leq \ln(C_P) + \ln\left((\lambda(\mathbf{P}))^{\frac{\ln(2C_P H \cdot k)}{\ln(1/\lambda(\mathbf{P}))}}\right) \\
&= \ln(C_P) + \ln(\lambda(\mathbf{P}))\frac{\ln(2C_P H \cdot k)}{\ln(1/\lambda(\mathbf{P}))} \\
&= \ln(C_P) + \ln(\lambda(\mathbf{P}))\frac{-\ln(2C_P H \cdot k)}{\ln(\lambda(\mathbf{P}))} \\
&= \ln(C_P) - \ln(2C_P H \cdot k) \\
\left|\left[\mathbf{P}^{\mathcal{T}_k}\right]_{i,j} - \frac{1}{n}\right| &\leq e^{\ln\left(\frac{C_P}{2C_P H \cdot k}\right)} \\
&= \frac{1}{2H \cdot k}
\end{aligned}$$

Since $H \geq |f_i^t(\boldsymbol{x}_k^t) - f_i^t(\boldsymbol{x}^*)|$ for all k and by Claim 6, $H = GD > 0$, therefore for all $i, j \in [n]^2$, we obtain the following

$$\frac{f_i^t(\boldsymbol{x}_{k-\mathcal{T}_k}^t) - f_i^t(\boldsymbol{x}^*)}{n} - \frac{1}{2k} \leq \left(f_i^t(\boldsymbol{x}_{k-\mathcal{T}_k}^t) - f_i^t(\boldsymbol{x}^*)\right)\left[\mathbf{P}^{\mathcal{T}_k}\right]_{i,j} \leq \frac{f_i^t(\boldsymbol{x}_{k-\mathcal{T}_k}^t) - f_i^t(\boldsymbol{x}^*)}{n} + \frac{1}{2k}$$

□

The idea of the proof at this point is to bound the model to the optimal with local functions f_i^t and show that it does converge. Then, by using the Markov chain's property and the claim 7, we can obtain a bound on our objective global function F^t.

$$\begin{aligned}
\eta_k \mathbb{E}\left[f_{i^k}^t(\boldsymbol{x}_{k-\mathcal{T}_k}^t) - f_{i^k}^t(\boldsymbol{x}_k^t)\right] &\leq G\eta_k \mathbb{E}\left\|\boldsymbol{x}_{k-\mathcal{T}_k}^t - \boldsymbol{x}_k^t\right\| && \text{(Recall that } \forall i, f_i^t \text{ is } G\text{-Lipschitz)} \\
&\leq G\eta_k \mathbb{E}\left[\sum_{\ell=k-\mathcal{T}_k}^{k-1} \left\|\boldsymbol{x}_{\ell+1}^t - \boldsymbol{x}_\ell^t\right\|\right] && \text{(Triangle inequality)}
\end{aligned}$$

$$
\begin{aligned}
&= G\eta_k \sum_{\ell=k-\mathcal{T}_k}^{k-1} \mathbb{E}\left\|\boldsymbol{x}_{\ell+1}^t - \boldsymbol{x}_\ell^t\right\| && \text{(Linearity of expectation)}\\
&= G\eta_k \sum_{\ell=k-\mathcal{T}_k}^{k-1} \mathbb{E}\left\|\boldsymbol{x}_\ell^t + \eta_\ell\left(\boldsymbol{v}_\ell^t - \boldsymbol{x}_\ell^t\right) - \boldsymbol{x}_\ell^t\right\| && \text{(By Algorithm 5, line 5)}\\
&= G\eta_k \sum_{\ell=k-\mathcal{T}_k}^{k-1} \eta_\ell \mathbb{E}\left\|\boldsymbol{v}_\ell^t - \boldsymbol{x}_\ell^t\right\|\\
&\leq GD\eta_k \sum_{\ell=k-\mathcal{T}_k}^{k-1} \eta_\ell\\
&&& \text{(The constrained set } \mathcal{K} \text{ is bounded with diameter } D)\\
&\leq \frac{GD}{2} \sum_{\ell=k-\mathcal{T}_k}^{k-1} \left[(\eta_\ell)^2 + (\eta_k)^2\right] && \text{(Cauchy-Shwartz)}\\
&\leq \frac{GD}{2}\mathcal{T}_k(\eta_k)^2 + \frac{GD}{2} \sum_{\ell=k-\mathcal{T}_k}^{k-1} (\eta_\ell)^2
\end{aligned}
$$

The asymptotic definition of $\mathcal{T}_k$ defined in (3.7) is as follows

$$
\mathcal{T}_k \in \mathcal{O}\left(\frac{\ln k}{\ln\left(1/\lambda(\mathbf{P})\right)}\right)
$$

$$
\sum_k \eta_k \mathbb{E}\left[f_{i^k}^t(\boldsymbol{x}_{k-\mathcal{T}_k}^t) - f_{i^k}^t(\boldsymbol{x}_k^t)\right] \leq \frac{GD}{2}\sum_k \mathcal{T}_k(\eta_k)^2 + \frac{GD}{2}\sum_k \sum_{\ell=k-\mathcal{T}_k}^{k-1} (\eta_\ell)^2
$$

$$
\begin{aligned}
\sum_k \sum_{\ell=k-\mathcal{T}_k}^{k-1} (\eta_\ell)^2 &\leq \sum_{k>\mathcal{T}_k} \mathcal{T}_k(\eta_{k-\mathcal{T}_k})^2\\
&\leq \sum_k \mathcal{T}_k(\eta_k)^2\\
&\leq \mathcal{O}\left(\frac{2}{\ln(1/\lambda(\mathbf{P}))}\sum_k (\eta_k)^2 \ln k\right)\\
&\leq \mathcal{O}\left(\frac{2}{\ln(1/\lambda(\mathbf{P}))}\sum_k \frac{\sqrt{k}}{k^{2q}}\right)\\
&= \mathcal{O}\left(\frac{2\zeta(2q-1/2)}{\ln\left(1/\lambda(\mathbf{P})\right)}\right)\\
&< \infty
\end{aligned}
$$

Therefore,

$$
\sum_k \eta_k \mathbb{E}\left[f_{i^k}^t(\boldsymbol{x}_{k-\mathcal{T}_k}^t) - f_{i^k}^t(\boldsymbol{x}_k^t)\right] \leq \mathcal{O}\left(\frac{2GD\zeta(2q-1/2)}{\ln\left(1/\lambda(\mathbf{P})\right)}\right) \tag{3.9}
$$

By combining equation 3.6 and equation 3.9, we obtain

$$\sum_k \eta_k \mathbb{E}\left[f_{i^k}^t(\boldsymbol{x}_{k-\mathcal{T}_k}^t) - f_{i^k}^t(\boldsymbol{x}^*)\right] \leq \mathcal{O}\left(\frac{2GD\zeta(2q-1/2)}{\ln\left(1/\lambda(\mathbf{P})\right)}\right) + H\zeta(2q) + \frac{\beta D^2\pi^2}{12} + \sum_k \frac{1}{k^{2q}} \sum_{\ell=1}^{k} \langle \nabla f_{j^\ell}^t(\boldsymbol{x}_\ell^t), \boldsymbol{v}_\ell^t - \boldsymbol{x}^* \rangle \tag{3.10}$$

Let $\mathcal{X}_k = \left(x_k^t, i^{k-1}\right)$

$$\begin{aligned}
\mathbb{E}_{i^k}\left[f_{i^k}^t(\boldsymbol{x}_{k-\mathcal{T}_k}^t) - f_{i^k}^t(\boldsymbol{x}^*) | \mathcal{X}_1, \ldots \mathcal{X}_{k-\mathcal{T}_k}\right] &= \sum_{i=1}^{n} \left(f_i^t(\boldsymbol{x}_{k-\mathcal{T}_k}^t) - f_i^t(\boldsymbol{x}^*)\right) \mathbb{P}\left[i^k = i | \mathcal{X}_1, \ldots, \mathcal{X}_{k-\mathcal{T}_k}\right] \\
&\qquad \text{(By definition of expectation)} \\
&= \sum_{i=1}^{n} \left(f_i^t(\boldsymbol{x}_{k-\mathcal{T}_k}^t) - f_i^t(\boldsymbol{x}^*)\right) \mathbb{P}\left[i^k = i | \mathcal{X}_{k-\mathcal{T}_k}\right] \\
&\qquad \text{(Using Markov's property)} \\
&= \sum_{i=1}^{n} \left(f_i^t(\boldsymbol{x}_{k-\mathcal{T}_k}^t) - f_i^t(\boldsymbol{x}^*)\right) \left[\mathbf{P}^{\mathcal{T}_k}\right]_{i^k-\mathcal{T}_k,i} \\
&\qquad \left(\mathbb{P}\left[i^k = i | \mathcal{X}_{k-\mathcal{T}_k}\right] = \left[\mathbf{P}^{\mathcal{T}_k}\right]_{i^k-\mathcal{T}_k,i}\right) \\
&\geq \sum_{i=1}^{n} \left[\frac{(f_i^t(\boldsymbol{x}_{k-\mathcal{T}_k}^t) - f_i^t(\boldsymbol{x}^*)}{n} - \frac{1}{2k}\right] \qquad \text{(By claim 7)} \\
&= (F^t(\boldsymbol{x}_{k-\mathcal{T}_k}^t - F^t(\boldsymbol{x}^*)) - \frac{n}{2k} \\
&\qquad \text{(By definition of } F^t = \tfrac{1}{n}\textstyle\sum_{i=1}^n f_i^t\text{)}
\end{aligned}$$

Therefore, we get the following inequality

$$\mathbb{E}_{i^k}\left[f_{i^k}^t(\boldsymbol{x}_{k-\mathcal{T}_k}^t) - f_{i^k}^t(\boldsymbol{x}^*) | \mathcal{X}_1, \ldots \mathcal{X}_{k-\mathcal{T}_k}\right] \geq (F^t(\boldsymbol{x}_{k-\mathcal{T}_k}^t - F^t(\boldsymbol{x}^*)) - \frac{n}{2k} \tag{3.11}$$

Equation 3.11 allows us to bring the established bounds on the local functions f_i^t to our objective function being $F^t = \frac{1}{n}\sum_{i=1}^n f_i^t$

Hence, by using equation 3.10 and the previous inequality 3.11, we obtain

$$\begin{aligned}
\sum_{i=1}^{n} \eta_k \mathbb{E}\left[F^t(\boldsymbol{x}_{k-\mathcal{T}_k}^t) - F^t(\boldsymbol{x}^*)\right] &\leq \sum_k \eta_k \mathbb{E}\left[f_{i^k}^t(\boldsymbol{x}_{k-\mathcal{T}_k}^t) - f_{i^k}^t(\boldsymbol{x}^*)\right] + \sum_{i=1}^{n} \eta_k \frac{n}{2k} \\
&\leq \mathcal{O}\left(\frac{2GD\zeta(2q-1/2)}{\ln\left(1/\lambda(\mathbf{P})\right)}\right) + H\zeta(2q) + \frac{n\zeta(q+1)}{2} \\
&\quad + \frac{\beta D^2\pi^2}{12} + \sum_k \frac{1}{k^{2q}} \sum_{\ell=1}^{k} \langle \nabla f_{j^\ell}^t(\boldsymbol{x}_\ell^t), \boldsymbol{v}_\ell^t - \boldsymbol{x}^* \rangle
\end{aligned} \tag{3.12}$$

Similarly, by using equation 3.9 and the previous inequality 3.11, we obtain

$$\sum_k \eta_k \mathbb{E}\left[F^t(\boldsymbol{x}_k^t) - F^t(\boldsymbol{x}_{k-\mathcal{T}_k}^t)\right] \leq \mathcal{O}\left(\frac{2nGD\zeta(2q-1/2)}{\ln\left(1/\lambda(\mathbf{P})\right)}\right) \tag{3.13}$$

We were able to bound the distance between $F^t(\boldsymbol{x}_{k-\mathcal{T}_k}^t)\cdots F^t(\boldsymbol{x}_k^t)$ with equation 3.13 and the distance $F^t(\boldsymbol{x}_{k-\mathcal{T}_k}^t)\cdots F^t(\boldsymbol{x}^*)$ with equation 3.12. Combining both equation 3.12 and equation 3.13 we are able to bound the distance between $F^t(\boldsymbol{x}_k^t)\cdots F^t(\boldsymbol{x}^*)$:

$$\begin{aligned}\sum_k \eta_k \mathbb{E}\left[F^t(\boldsymbol{x}_k^t) - F^t(\boldsymbol{x}^*)\right] \leq \mathcal{O}\left(\frac{2(n+1)GD\zeta(2q-1/2)}{\ln\left(1/\lambda(\mathbf{P})\right)}\right) + H\zeta(2q) + \frac{n\zeta(q+1)}{2} \\ + \frac{\beta D^2\pi^2}{12} + \sum_k \frac{1}{k^{2q}} \sum_{\ell=1}^{k} \langle \nabla f_{j^\ell}^t(\boldsymbol{x}_\ell^t), \boldsymbol{v}_\ell^t - \boldsymbol{x}^*\rangle\end{aligned} \tag{3.14}$$

Define $\overline{\boldsymbol{x}}_K^t$ and $\overline{\boldsymbol{x}}^t$ as

$$\overline{\boldsymbol{x}}_K^t = \frac{1}{\sum_{k=1}^K \eta_k} \sum_{k=1}^{K} \eta_k \boldsymbol{x}_k \tag{3.15}$$

$$\overline{\boldsymbol{x}}^t = \frac{1}{\sum_k \eta_k} \sum_{k} \eta_k \boldsymbol{x}_k \tag{3.16}$$

$$\begin{aligned}\left(\sum_k \eta_k\right)\left[F^t(\overline{\boldsymbol{x}}^t) - F^t(\boldsymbol{x}^*)\right] &\leq \sum_k \eta_k \mathbb{E}\left[F^t(\boldsymbol{x}_k^t) - F^t(\boldsymbol{x}^*)\right] && \text{(By Jensen's inequality)} \\ &\leq \mathcal{O}\left(\frac{2(n+1)GD\zeta(2q-1/2)}{\ln\left(1/\lambda(\mathbf{P})\right)}\right) + H\zeta(2q) + \frac{n\zeta(q+1)}{2} + \frac{\beta D^2\pi^2}{12} \\ &\quad + \sum_k \frac{1}{k^{2q}} \sum_{\ell=1}^{k} \langle \nabla f_{j^\ell}^t(\boldsymbol{x}_\ell^t), \boldsymbol{v}_\ell^t - \boldsymbol{x}^*\rangle\end{aligned}$$

$$\begin{aligned}\frac{1}{T}\sum_{t=1}^{T}\left(\sum_{k=1}^{K}\eta_k\right)\mathbb{E}\left[F^t(\overline{\boldsymbol{x}}_K^t) - F^t(\boldsymbol{x}^*)\right] \leq \mathcal{O}\left(\frac{2(n+1)GD\zeta(2q-1/2)}{\ln\left(1/\lambda(\mathbf{P})\right)}\right) + H\zeta(2q) + \frac{n\zeta(q+1)}{2} + \frac{\beta D^2\pi^2}{12} \\ + \frac{1}{T}\sum_{t=1}^{T}\sum_{k=1}^{K}\frac{1}{k^{2q}}\sum_{\ell=1}^{k}\langle \nabla f_{j^\ell}^t(\boldsymbol{x}_\ell^t), \boldsymbol{v}_\ell^t - \boldsymbol{x}^*\rangle\end{aligned}$$

$$
\begin{aligned}
&\overset{(a)}{=} \mathcal{O}\left(\frac{2(n+1)GD\zeta(2q-1/2)}{\ln\left(1/\lambda(\mathbf{P})\right)}\right) + H\zeta(2q) + \frac{n\zeta(q+1)}{2} + \frac{\beta D^2\pi^2}{12} \\
&\quad + \sum_{k=1}^{K} \frac{1}{k^{2q}} \sum_{\ell=1}^{k} \frac{1}{T} \sum_{t=1}^{T} \langle \nabla f_{j^\ell}^t(\boldsymbol{x}_{j^\ell}^t), \boldsymbol{v}_\ell^t - \boldsymbol{x}^* \rangle \\
&\overset{(b)}{\leq} \mathcal{O}\left(\frac{2(n+1)GD\zeta(2q-1/2)}{\ln\left(1/\lambda(\mathbf{P})\right)}\right) + H\zeta(2q) + \frac{n\zeta(q+1)}{2} + \frac{\beta D^2\pi^2}{12} \\
&\quad + \sum_{k=1}^{K} \frac{\mathcal{R}^T}{k^q T} \\
&\leq \mathcal{O}\left(\frac{2(n+1)GD\zeta(2q-1/2)}{\ln\left(1/\lambda(\mathbf{P})\right)}\right) + H\zeta(2q) + \frac{n\zeta(q+1)}{2} + \frac{\beta D^2\pi^2}{12} \\
&\quad + \frac{\mathcal{R}^T}{T} \sum_{k=1}^{K} \eta_k
\end{aligned}
$$

We get equality (a) because the feedbacks for the oracles from the walk $j^1, \ldots, j^k$ are

$$
\langle \nabla f_{j^1}^t(\boldsymbol{x}_{j^1}^t), \cdot \rangle \quad , \quad \ldots \quad , \quad \langle \nabla f_{j^k}^t(\boldsymbol{x}_{j^k}^t), \cdot \rangle.
$$

We get inequality (b) from the assumption that oracles have regret $\mathcal{R}^T$. Finaly, we conclude with the following:

$$
\begin{aligned}
\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[F^t(\overline{\boldsymbol{x}}^t) - F^t(\boldsymbol{x}^*)\right] &\leq \frac{\mathcal{O}\left(\frac{2(n+1)GD\zeta(2q-1/2)}{\ln\left(1/\lambda(\mathbf{P})\right)}\right) + H\zeta(2q) + \frac{n\zeta(q+1)}{2} + \frac{\beta D^2\pi^2}{12} + \frac{\mathcal{R}^T}{T}\left(\sum_{k=1}^{K}\eta_k\right)}{\left(\sum_{k=1}^{K}\eta_k\right)} \\
&= \mathcal{O}\left(\frac{1}{K^{1-q}} + \frac{\mathcal{R}^T}{T}\right)
\end{aligned}
$$

If we take $K = \mathcal{O}\left(\left(\sqrt{T}\right)^{1/(1-q)}\right)$ then we finally get

$$
\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[F^t(\overline{\boldsymbol{x}}^t) - F^t(\boldsymbol{x}^*)\right] = \mathcal{O}\left(\frac{1}{\sqrt{T}}\right) \tag{3.17}
$$

4 Experiments and Applications

In this chapter, we first demonstrate the performance of both Algorithm 3 and Algorithm 5 for optimizing the online multiclass logistic regression model on real-world standard image datasets (MNIST,CIFAR$_{10}$) by comparing with centralized online algorithms. We outperform the best-known decentralized constrained online algorithms in terms of regret bounds. In addition, we run this experiment on a network of computing units with limited memory and processing power, specifically Raspberry Pi 3b+, demonstrating the algorithm's frugality and its suitability to edge computing applications. In conclusion, we discuss the applicability of Algorithm 3 to an actual smart building application.

4.1 Decentralized Online Multiclass Logistic Regression

This section begins by describing the settings of the Decentralized Online Multiclass Logistic Regression, derived from [Xie+20]. Then, we respond to the following questions based on our experimental findings:

1. Are both Algorithm 3 and Algorithm 5's theoretical results regarding the regret bound validated?
2. How do they compare to the centralized model[1] ([Xie+20]) and the best-known algorithm in identical settings ([Zha+17])?
3. How do they scale depending on the size and the topologies of graphs?
4. Are these algorithms resource-efficient enough to be embedded in devices with limited resources?

4.1.1 Settings of Decentralized Online multiclass Logistic Regression

[Setting 1] The Online Multiclass Logistic Regression.

We consider the online version of the multiclass logistic problem on two standard image datasets; MNIST [Lec+98] and CIFAR10 [Kri09]. At each time step $t = 1, \ldots, T$, we receive a subset

[1] The centralized model is equivalent to when the given graph is of size one. Note that when given such a graph, Algorithm 3 is equivalent to the MFW algorithm from [Che+18].

$\mathcal{B}^t$ whose size is the same for all the iterations. Each image in $\mathcal{B}^t$ is of the form $(\boldsymbol{x}, y) \in \mathbb{R}^d \times \{1, \ldots, C\}$. Here $\boldsymbol{x}$ is an image feature vector, and y is the class corresponding to that image. The multiclass logistic loss function f is defined as follows:

$$f(\mathbf{X}, \mathcal{B}) = -\sum_{b \in \mathcal{B}} \sum_{c=1}^{C} \{y_b = c\} \log \frac{\exp(\mathbf{X}_c^\intercal \boldsymbol{x}_b)}{\sum_{i=1}^{C} \exp(\mathbf{X}_i^\intercal \boldsymbol{x}_b)}$$

and the constraint is set to be $\mathcal{K} = \{\mathbf{X} \in \mathbb{R}^{d \times C} : \|\mathbf{X}\|_1 \leq r\}$ for some constant $r \in \mathbb{R}^+$, where $\|\mathbf{X}\|_1$ denotes the matrix ℓ_1 norm, i. e. $\|\mathbf{X}\|_1 = \max_{1 \leq j \leq C} \sum_{i=1}^{d} |[\mathbf{X}]_{i,j}|$. We consider the adversarial online settings where the data points are sorted by class label and then the sorted batches $\{\mathcal{B}^1, \ldots, \mathcal{B}^T\}$ are selected sequentially from these datasets. For the MNIST dataset, we set $|\mathcal{B}^t| = 600$ and $r = 8$. For CIFAR10, we set $|\mathcal{B}^t| = 500$ and $r = 32$. Next, we describe the problem in the distributed settings.

[Setting 2] Distributed Online Logistic Regression.

Given a network of n-agents, at each time $t = 1, \ldots, T$, we split each batch $\mathcal{B}^t$ into n parts $\{\mathcal{B}_1^t, \ldots, \mathcal{B}_n^t\}$ that are evenly sized that is, for every player i we have $|\mathcal{B}_i^t| \approx \lfloor {}^{|\mathcal{B}^t|}/_n \rfloor$. Thus, at each time t, each agent $i \in [n]$ only observes its private loss function f_i^t defined as $f_i^t(\mathbf{X}) = f(\mathbf{X}, \mathcal{B}_i^t)$. For the regret computation, we are interested in the following loss function: $F^t(\mathbf{X}) = \frac{1}{n} \sum_{i=1}^{n} f_i^t(\mathbf{X})$. Note that at time step t, each player i takes a local decision $\mathbf{X}_i^t$ in a distributed setting. Thus, while computing the regret of the algorithm, we consider the loss function that maximizes the total loss, that is

$$\mathbf{E}^t = \max_{\mathbf{X} \in \{\mathbf{X}_j^t \,:\, 1 \leq j \leq n\}} F^t(\mathbf{X})$$

where $\mathbf{E}^t$ denote the cost/error of the algorithm at time t.

4.1.2 Simulated SDMFW

In this section, we demonstrate the performance of the algorithm in terms of minimizing the regret. We want to verify the theoretical bound of the regret on the decentralized online multiclass logistic regression problem and position the method at the cutting edge of the field. We benchmark Algorithm 3 (denoted by SDMFW) on real-world datasets. Our experiments are broadly categorized as follows:

- In the first set of experiments, we show that Algorithm 3 is competitive and achieves the theoretical regret of $\mathcal{O}\left(\sqrt{T}\right)$.
- Next, we compare the performance of SDMFW, to the Distributed Regularized Online Frank-Wolfe algorithm(denoted by DROFW) [Zha+17] on grid networks with varying sizes.
- Finally, we present the performance of our algorithm SDMFW, on different types of network topologies.

Although the algorithm is distributed, all experiments in this section are performed in MATLAB 9.8 (release 2020A) running on a single machine with Ubuntu version 20.04 LTS as an operating system. You can find the source code in `https://gricad-gitlab.univ-grenoble-alpes.fr/youssefp/sdmfw`

To compare with the centralized version of our algorithm, we derived our experiment's setup from [Xie+20]'s setup (Section 6 , [Xie+20]). We reconducted the same online setting (described in [Setting 1]) and distributed the data on a decentralized topology (described in [Setting 2]) in which we can compare with DROFW from [Zha+17].

Choice of hyperparameters.

For both MNIST and CIFAR10 testbeds, we chose the hyperparameters via grid search. For the sake of direct comparison with [Xie+20], we took the same initial point arbitrarily to be 0. Recall that in these experiments, the functions are convex; therefore, the choice of the initial point is not crucial. As for the number of steps L in each iteration, we found that $L = 10$ for MNIST and $L = 50$ for CIFAR10 is enough to ensure our guarantees instead of $L = T^{3/2}$.

Observation of our experiments.

By comparing our results with the experiments of the centralized setup of [Xie+20][2], we conclude that our algorithm is robust to stochastic gradient estimates. It indeed achieves the expected theoretical regret guarantee of $\mathcal{O}\left(\sqrt{T}\right)$. Next, we observe that SDMFW's regret (showing in Figure 4.1(a) vs. Figure 4.1(b) and Figure 4.3(a) vs. Figure 4.3(b) for MNIST and CIFAR10 datasets, respectively) has a lower regret value than the current best-known algorithm in these settings, namely DROFW. Finally, we show the robustness of our algorithm according to the topology and the size of a given network. We focus on cycle graphs since they are the least connected topologies and have a high diameter. Observe in Figure 4.1(c) and Figure 4.3(c) that the regrets of SDMFW for different sizes of the cycles remain close. However, there is a slight degradation when the network size increases. It is also interesting to notice that the obtained

[2]The results of the experiments conducted on centralized settings can be found in [Xie+20], as depicted in Figure 2. Please be aware that the scale of the regret values is different from ours, as we normalized the regret values in our experimentation.

regret of SDMFW on grids is slightly better than the regret on cycles. This implies that the performance of SDMFW is closely related to the connectivity (average fan-out) and the diameter of the underlying graph.

Remark 7.

Recall that we consider regret with hindsight, meaning that the algorithm can achieve negative regret. One does not always expect the regret to decrease. The regret can increase over time. However, if it does increase, it should increase sublinearly (in our case, bounded by $\mathcal{O}\left(\sqrt{T}\right)$). We can also observe small bumps on the curves. The adversarial setting explains this behavior since the data arrives sorted by class labels in this setting. Thus, we expect the online algorithms to increase their regret as soon as the adversary newly introduces batches of data with a never seen class label.

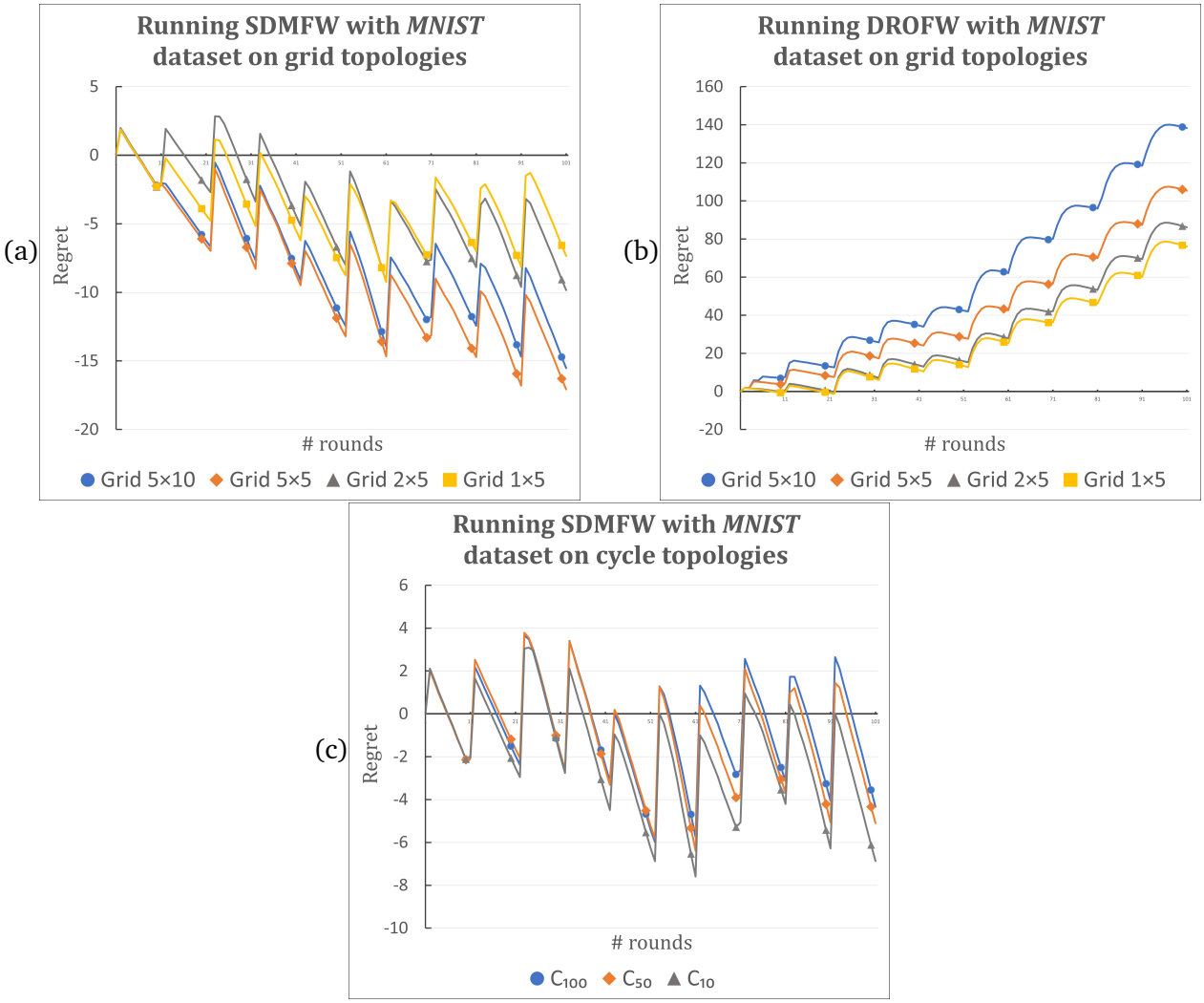
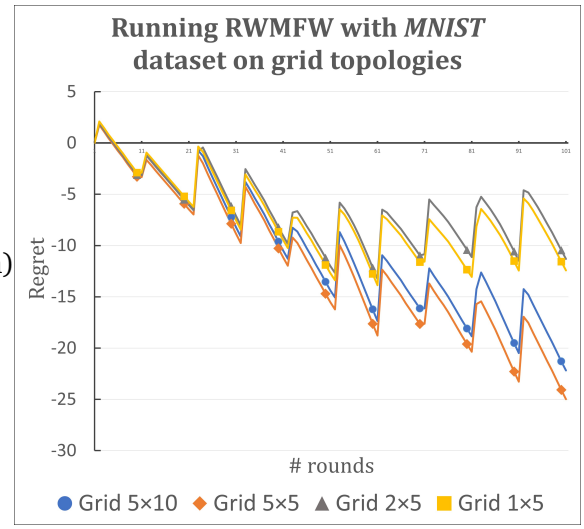


Fig. 4.1: Performance of SDMFW on grid network with varying sizes (a), DROFW with varying network size (b), and DMFW with cycle network with varying sizes (c) on MNIST dataset. ($L = 10$)

(a)



(b)

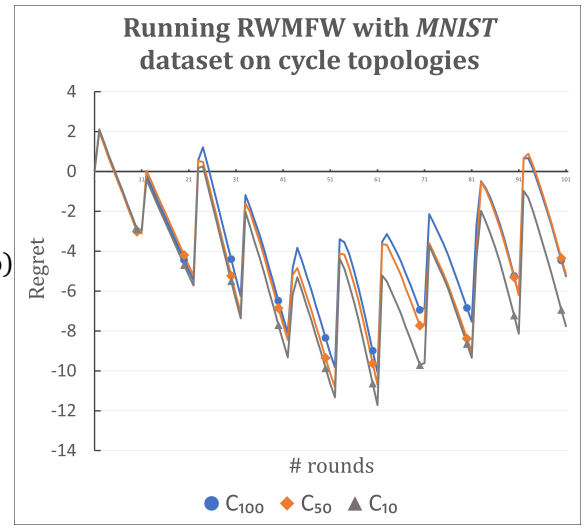


Fig. 4.2: Average performance of 100 runs of RWMFW on grid topology with varying network size (a), RWMFW with a cycle network topology with varying sizes (b) on MNIST dataset. Note that the variability due to the random walk is not presented in this figure: See Figure 4.5 to visualize the spread. $L = 10$

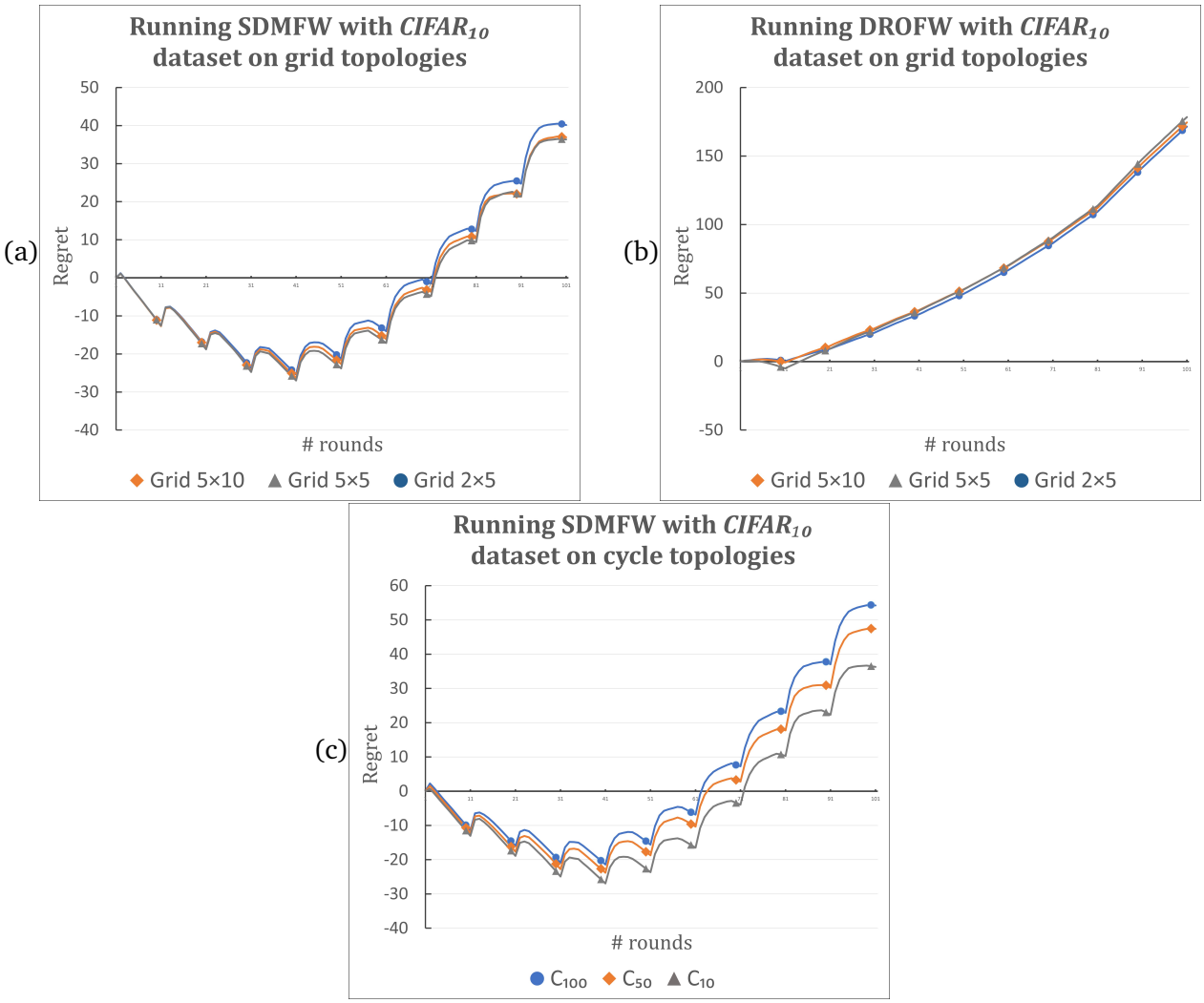
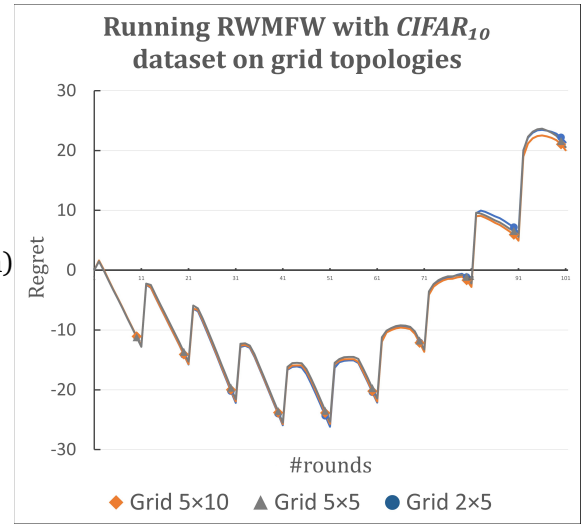


Fig. 4.3: Performance of SDMFW on grid topology with varying network sizes (a), DROFW with varying network size (b), and SDMFW with cycle network with varying sizes (c) on CIFAR10 dataset.($L = 20$)

(a)



(b)

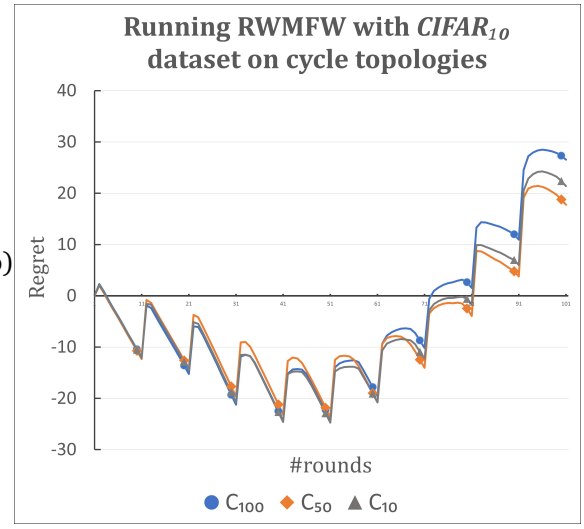
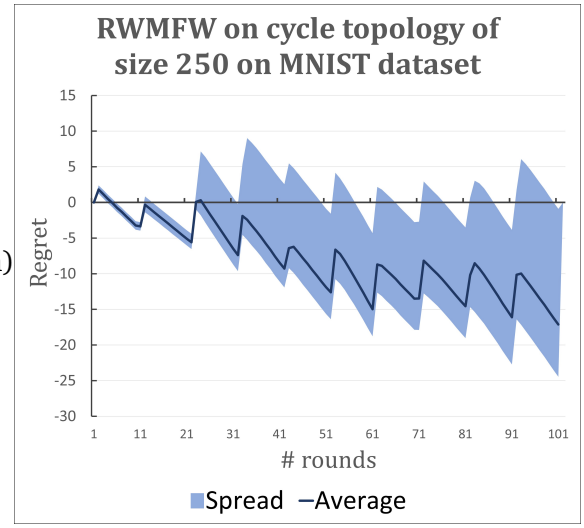


Fig. 4.4: Average performance of 100 runs of RWMFW on grid topology with varying network size (a), RWMFW with a cycle network topology with varying sizes (b) on CIFAR10 dataset. Note that the variability due to the random walk is not presented in this figure: See Figure 4.6 to visualize the spread.($L = 50$)

(a)



(b)

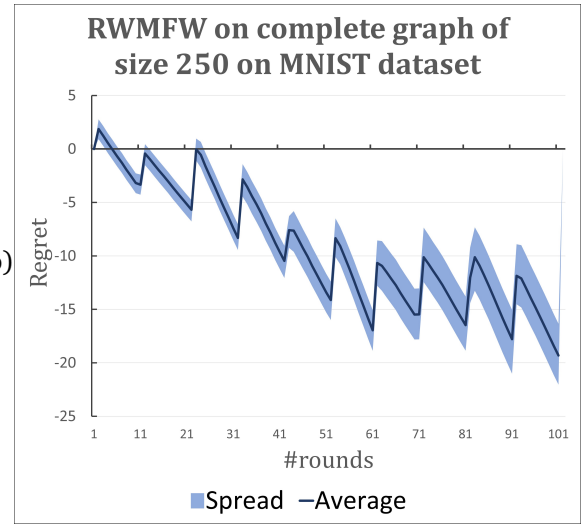
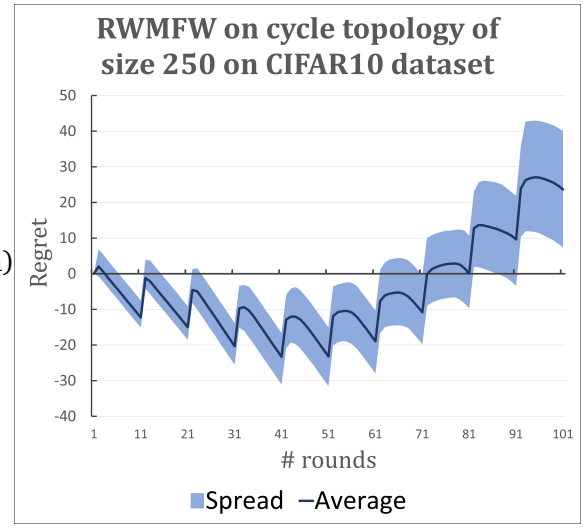


Fig. 4.5: Average performance of 100 runs of RWMFW on cycle network of size 250 (a), RWMFW with a complete network of size 250 (b) on MNIST dataset.($L = 100$)

(a)



(b)

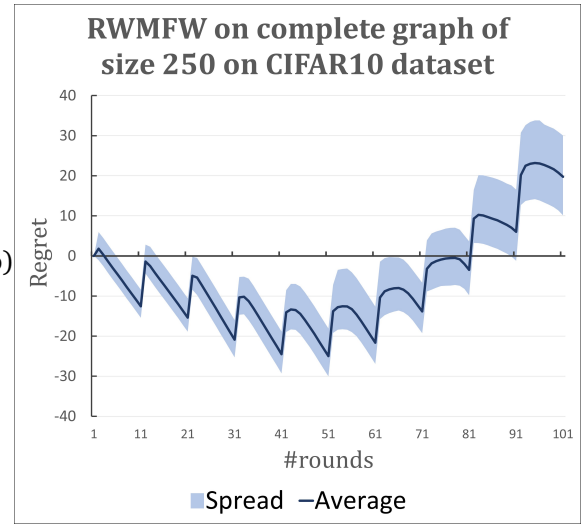


Fig. 4.6: Average performance of 100 runs of RWMFW on cycle network of size 250 (a), RWMFW with a complete network of size 250 (b) on CIFAR10 dataset. ($L = 100$)

4.1.3 Random Walk Approach of the Decentralized Online Multiclass Logistic Regression

In this section, we benchmark Algorithm 5 (denoted by RWMFW) on the same datasets as in section 4.1.2 and with analogous settings. Our experiments are broadly categorized as follows:

- In the first set of experiments, we show that Algorithm 5 is competitive and achieves the theoretical regret of $\mathcal{O}\left(\sqrt{T}\right)$.
- Next, we compare the performance of RWMFW to the centralized Meta Frank Wolfe algorithm (denoted by MFW) [Che+18], and our algorithm SDMFW (Algorithm 3).
- Finally, we present the performance of our algorithm, RWMFW, on different network topologies.

Although the algorithm is distributed, all experiments in this section are conducted in a simulated environment with Python 3.8 running on a single machine with Ubuntu version 20.04 LTS as an operating system. You can find the source code in `https://github.com/pyouss/SimRWMFW`

Similarly to section 4.1.2, to compare with the centralized version of our algorithm, we derived our experiment's setup from [Xie+20]'s setup. We reconducted the same online setting (described in [Setting 1]) and distributed the data on a decentralized topology (described in [Setting 2]) in which we can compare with DROFW from [Zha+17].

Choice of hyperparameters.

For both MNIST and CIFAR10 testbeds, we chose the hyperparameters via grid search. For the sake of direct comparison with [Xie+20] and section 4.1.2, we took the same initial point arbitrarily to be 0. Recall that in these experiments, the functions are convex; therefore, the choice of the initial point is not crucial. As for the number of steps L in each iteration, we found that $L = 50$ is enough to ensure our theoretical guarantees instead of $L = T$. Nevertheless, when given weakly connected graphs such as cycles with a large number of agents ($n > 100$), we observe that choosing higher values of L improves the quality of the incurred regret. Therefore, for such instances, we chose $L = 100$, which corresponds with the theoretical requirement.

Observation of our experiments.

By comparing our results with the experiments in [Xie+20], we conclude that RWMFW competes with even centralized projection-free algorithms. It achieves the expected theoretical regret guarantee of $\mathcal{O}\left(\sqrt{T}\right)$. Next, we observe that RWMFW's regret (showing in Figure 4.2(a) vs. Figure 4.1(b) and Figure 4.4(a) vs. Figure 4.3(b) for MNIST and CIFAR10 datasets, respectively)

has a lower regret value than the current best-known algorithm in these settings, namely DROFW.

Topology matters.

Finally, we study the robustness of our algorithm according to the topology and the size of a given network. We focus on cycle graphs since they are the least connected topologies and with a diameter of order $\mathcal{O}(n)$. Observe in Figure 4.2(b) and Figure 4.4(b) that the regrets of RWMFW for different sizes of the cycles remain close. However, there is a slight degradation when the network size increases. It is also interesting to notice that the obtained regret of RWMFW on grids with a better diameter of order $\mathcal{O}(\sqrt{n})$ is slightly better than the regret on cycles. To further investigate topology dependence, we show with Figure 4.5 and Figure 4.6 that the spread (the blue shaded area) of the regret values with 100 runs of simulations on a cycle graph with 250 nodes is larger much larger than the one with a complete graph topology. This implies that the performance of RWMFW is closely related to the connectivity (average fan-out) of the underlying graph. Nevertheless, the average regret value in Figure 4.5 is much closer to the lower extremity of the spread area, implying that the lower part of this area is dense. Another point to note is that even the upper part of the spread still guarantees our theoretical bounds and still has a lower value than DROFW (Figure 4.1 (b) and Figure 4.3 (b)).

RWMFW *vs* SDMFW.

We observe that RWMFW yields in average slightly better results in terms of regret with the same number of iterations L per round which is expected by the theoretical analysis: RWMFW requires $L = \mathcal{O}(T)$ to bound the regret by $\mathcal{O}\left(\sqrt{T}\right)$ and in contrast SDMFW requires $L = \mathcal{O}\left(T^{\frac{3}{2}}\right)$. In terms of computations, the algorithm SDMFW is scaled by the number of agents n, and the communication is scaled according to the number of edges in the graph $|\mathcal{E}|$. In contrast, the RWMFW algorithm's dependence on the graph's topology is more subtle and is related to the Markov mixing time. To illustrate, given a complete graph, RWMFW's output varies less then when given a cycle. This variation of ouputs is not present in SDMFW which makes it more reliable. Nevertheless, when choosing the same number L of iterations or oracles, RWMFW is more efficient in computation and communication cost. This is because of the protocol of RWMFW in which, at each iteration, only one agent is performing the computation. After the iteration is performed, this agent communicates only the updates to only one other agent.

4.1.4 Deployment of the online multi-class logistic regression on devices with limited resources (Raspberry Pi)

In section 4.1.2, we validated the theoretical bound of chapter 2 on the decentralized online multiclass logistic regression problem. The purpose of this section is to show that learning

algorithms can be deployed on hardware with restricted in both memory and computing. Our goal is not to provide an efficient/practical solution (in particular in the view point of communication) but to assess the feasibility on constrained devices.

To demonstrate the aforementioned, we implemented the experiment described in section 4.1.2 and deployed the algorithm on a network of Raspberry Pi connected by WiFi using the WalT platform [Bru+16] (see Fig.4.10). Simply put, WalT enables simple deployment as well as reproducibility. Practically speaking, you first have to install a *WalT server* in order to manage the deployment of your application into the desired devices (see [Dub22]). The WalT server is used to build and deploy *WalT images* onto the computing units of your choice, referred to as *WalT nodes*. WalT images are similar to Docker containers, except unlike Docker containers, WalT images have a Linux kernel.

Remark 8.

WalT is not necessary for the algorithm's implementation. Despite the fact that we are using a WalT server, it should not be confused with a central server that orchestrates the learning algorithm. It is simply a tool for configuring the experiment's settings and making it easy to replicate.

Communication protocol.

We are utilizing WiFi technology as a communication tool. The devices are all connected to a WiFi access point router. In fact, the physical network topology for the experiment is a star (see Fig. 4.8) where the central node is the router. This router is equipped with a RabbitMQ PUB/SUB broker that enables us to manage device-to-device communication given a specific topology (see Fig. 4.9). To accomplish this, the broker creates n channels for each agent, where n is the total number of working agents. Consider a given graph $\mathcal{G}$, where a node i is a neighbor of j, then i can broadcast a message on j's channel. Once a node is ready to receive a message, it simply pulls the last published message of its corresponding channel. It is important to note that the communication protocol used in this part of the experiment is a centralized one. However, the focus of this section is not to find the optimal communication decentralized protocol. The protocol was chosen for experimental purposes as opposed to its practical network efficiency in real-world applications. Its simplicity and flexibility to simulate any graph topology allows to focus on whether or not the computation scales to the limited local computation and storage resources.

The online stream of the data.

Using the WalT platform, we distribute data in batches to each node as needed. This platform serves as an environment that provides feedback to each agent once they make a decision. In practice, each WalT node has a network file system shared with the WalT server in which a node

can access files using the WalT network. We utilize this feature to store the distributed online dataset batches β_i^t in the WalT server; and at each round t, a node i can read his respective batch β_i^t.

Emphasis and observation of this experiment.

The focus of this study is efficiency in relation to the limited resources a user may have for an edge application. In other words, we would like to be able to execute a nontrivial learning optimization process on devices with limited storage and computing capability. For instance, the centralized classic train and test approach are usually not feasible for restricted devices (see Remark 9). In contrast, our algorithm, SDMFW, fits the online and distributed settings (see 4.1.2) in which the problem gets decomposed into extremely small pieces. Therefore, a nontrivial optimization algorithm for machine learning can be run on an edge device, such as an embedded computing unit like a Raspberry Pi.

Hardware and technology we used for WalT nodes.

We chose the WalT node's hardware to have limited resources regarding working memory (RAM) and computing power (CPU frequency and without GPU). As a result, the *Raspberry Pi 3 Model B+* is a good choice for this constraint; its hardware specifications are as follows:

Raspberry Pi 3 Model B+

- **BUS width :** 32 bits
- **number of cores :** 4
- **CPU capacity :** 1400 MHz
- **RAM memory :** 923 MiB

Since our algorithm is online, a node receives a *small* batch of data, then discards it after processing it. Consequently, WalT nodes do not require storage; in our experiment, they do not have any. RAM memory of 923 MiB is sufficient to store all algorithm requirements and run the algorithm, including the oracle feedback, data buffer, libraries, and learning model. Due to the lack of disk memory, the complete Linux file system is stored on the WalT server. Thus, in order to implement this experiment without WalT, one may need storage for its operating system that is not excessively large for Linux distributions.

Remark 9.

With such limited hardware specifications, it is impossible to read the entire dataset of CIFAR_{10} at once, hence an offline method like Frank Wolfe is unfeasible. (The CIFAR_{10} dataset is larger than 2GB, which surpasses the devices' RAM memory capacity of less than 1GB.)

Discussion and benchmark.

First, we show that it is feasible to optimize the online multiclass logistic regression problem described in 4.1.2 with our algorithm SDMFW using a cluster of Raspberry Pi Model 3 B+. In fact, the bigger the cluster, the smaller the batches that each node has to process. However, the connectivity of the network also has a computational cost. The higher the degree of a node, the more it has a larger number of iterates to federate. As shown in section 4.1.2, the performance of SDMFW in terms of regret is robust to the number of nodes. However, there is a minor trade-off between network connectivity and SDMFW's regret performance. As a result, when deploying on edge devices with limited resources, one should consider maximizing the number of nodes while minimizing connectivity to meet his needs in accuracy.

We benchmarked how long a round takes on average in our experiment. To have a better idea of what is the most expensive in the experiment, we conducted the following timing benchmarks:

- τ_{round} : average time of a round
- τ_{comm} : average time of communication exchanges in a round
- $\tau_{i/o}$: average time for receiving and reading a batch of data from the WalT server
- τ_{proc} : average local execution time done by the Rasberry Pi in a round.

In a round, a node receives a batch of data. There are L iterations in a round, each with two communication exchanges. The residual operations are the local computations for the optimization process. Thus,

$$\tau_{round} = \tau_{comm} + \tau_{i/o} + \tau_{proc} \tag{4.1}$$

	MNIST with batch size 600					
	Line graph			**Complete Graph**		
n	**3**	**5**	**7**	**3**	**5**	**7**
τ_{comm}	5.24	5.802	7.267	5.615	12.048	24.032
τ_{proc}	0.447	0.288	0.223	0.458	0.298	0.223
$\tau_{i/o}$	1.35	1.267	1.238	1.340	1.268	1.232
τ_{round}	7.078	7.393	8.703	7.457	13.642	25.508

Tab. 4.1: Benchmarking the different types of operations' average execution time in seconds of SDMFW with the least connected graph topology (a line) and various sizes (3, 5 and 7). In the experiment, L being the number of iterations in a round, we take L equals to 10 which yields the best known regret bound.

	CIFAR10 with batch size 500					
	Line graph			Complete Graph		
n	**3**	**5**	**7**	**3**	**5**	**7**
τ_{comm}	29.383	32.951	37.997	30.503	66.338	139.883
τ_{proc}	1.508	0.964	0.707	1.506	0.963	0.709
$\tau_{i/o}$	5.48	5.013	4.775	5.46	5.013	4.822
τ_{round}	36.372	38.929	43.479	37.470	72.314	145.415

Tab. 4.2: Benchmarking the different types of operations' average execution time in seconds of SDMFW with the least connected graph topology (a line) and various sizes (3, 5 and 7). In the experiment, L being the number of iterations in a round, we take L equals to 10 which yields the best known regret bound.

Concluding remarks.

We showed that the decentralized online settings allows the learning problem to fit within the constraints of the hardware. Therefore, Algorithm 3 which deals with the uncertainties coming from both the online and the decentralized settings proved to be implementable with computing and memory restrictions. However, the proposed implementation emulates a decentralized peer-to-peer communication using a central router. To achieve full deployment in practice of the algorithm, it would be convenient to improve on the implementation of the communication. Towards improving the communication protocol, we suggest two approaches :

- Note that Algorithm 3 requires local synchronization between neighbors. The synchronization highly slows down the global computation of the algorithm by a cascading waiting effects (agent i waits for agent j which consequently blocks agent k that waits for agent i and so on...) . The conception of RWMFW (Algorithm 5) which drops synchronization is motivated by the synchronization's drawbacks.
- A different direction which is out of scope of this research is to investigate technologies and communication protocols that better suited for decentralized algorithms.

How to reproduce the experiment.

In order to reproduce our experiment, you will have to follow the following steps :

1. Create your own WalT server by following the tutorial described in `https://walt-project.liglab.fr/-/wikis/Resources-and-Documentation` (for more details see [Bru+16]
2. Gather similar/compatible hardware components
3. Clone the WalT images from the dockerhub in your WalT server
4. Clone the git repo, which contains scripts that handle the experimentation with instruction on how to use: `https://github.com/pyouss/WalT-SDMFW-Log-Reg`
5. Using the scripts handler :
   a) Boot the WalT images on your WalT nodes. (Fig. 4.7)

b) Connect the nodes to the broker. (Fig. 4.8)
c) Configure the parameters as desired.
d) Send the desired graph topology, algorithm's parameters (number of rounds, size of batches, ...), and the dataset to the nodes and the WalT platform. (Fig. 4.9)
e) Launch the experiment.

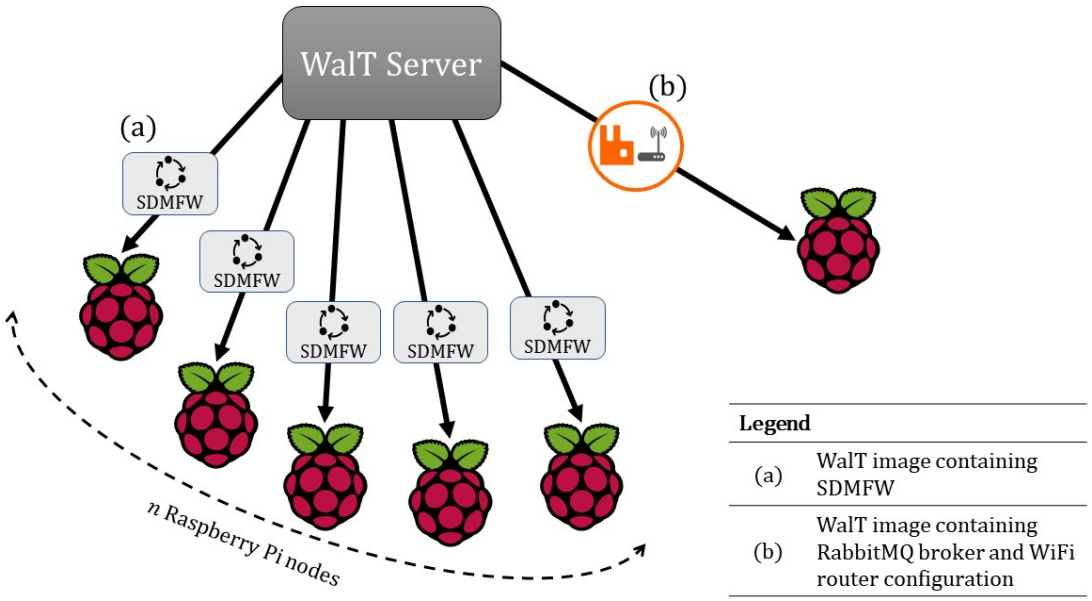


Fig. 4.7: Using the WalT server, we deploy the needed WalT images onto our WalT nodes. Suppose having $n+1$ Raspberry Pi then n of them will boot with a WalT image containing the algorithm SDMFW. The remaining node receives a WalT image configured to act as a WiFi access point (router). In addition to the access point feature, the image contains a RabbitMQ broker, which enables handling the exchanges between the nodes.

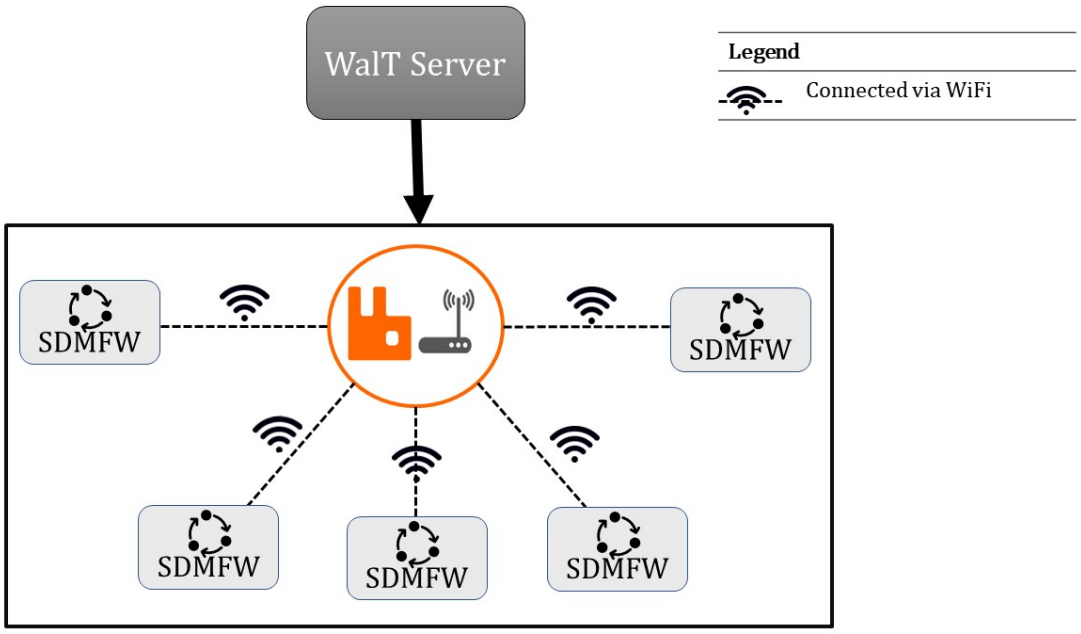


Fig. 4.8: Suppose having $n+1$ Raspberry Pi then n of them are booted with a WalT image containing the algorithm SDMFW. The remaining node is configured to act as WiFi access point (router). All the node connect to the node that acts as a router through WiFi.

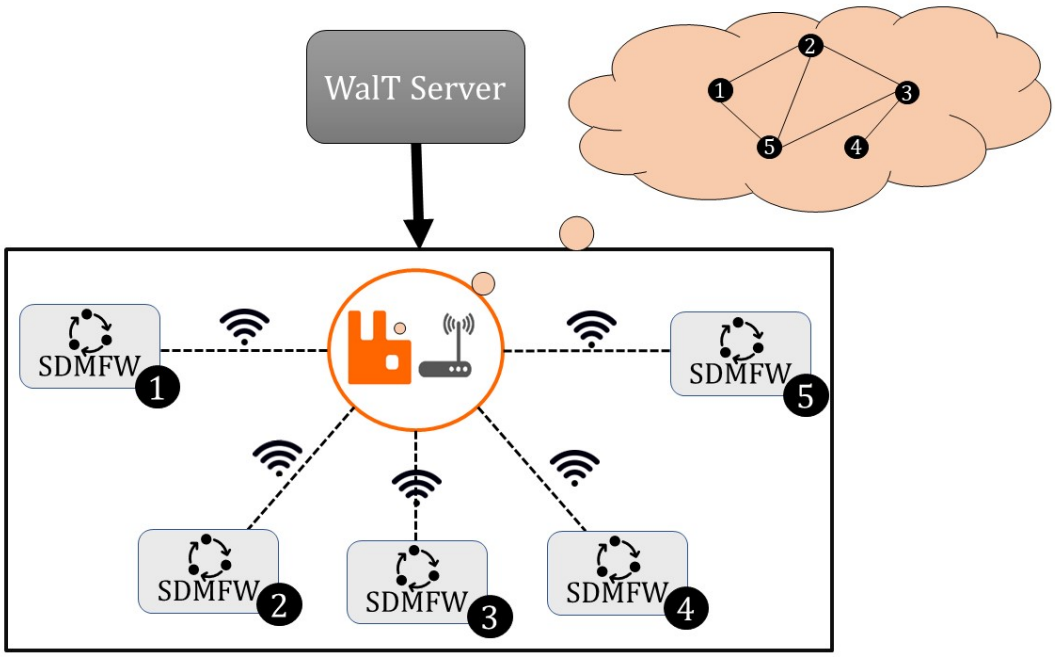


Fig. 4.9: The broker/router node manages the communication to satisfy a given graph topology. For instance, in this diagram, according to the broker, node 1 can connect with nodes 2 and 5, but not with nodes 3 and 4.

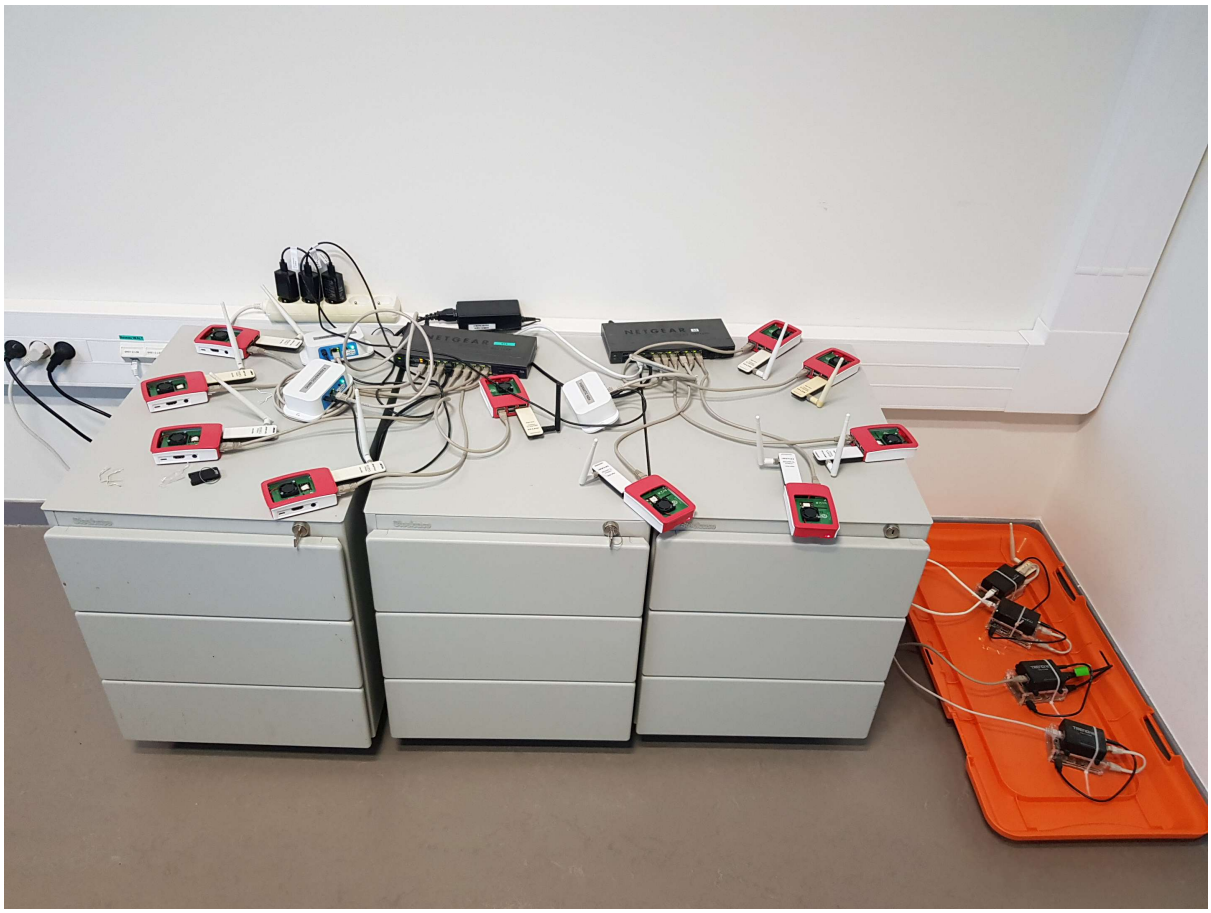


Fig. 4.10: Cluster of 10 Raspberry Pi with WiFi drivers, connected to a WalT server.

4.2 An IoT application: Temperature Forecasting Experiments on Thailand's Smart Building Dataset

All prior analyses and experiments are based on the assumption that the objective functions is convex. However, the majority of predictive models that have proven to be effective in practice are not convex, in particular neural networks. In this section, we apply the Algorithm 3 to the optimization of an online neural network (non-convex objective) by a group of autonomous learners. We demonstrate its usefulness in a smart building application in which zones of the building imitate learners optimizing a temperature forecasting problem. This study was done with the collaboration of Angan Mitra and Tuan Anh Nguyen ([Mit+22]) in which include both theoretical and experimental analysis. The main purpose of this section is to show the utility of the proposed algorithms where convexity assumptions are relaxed.

4.2.1 Settings and formulation

In this experiment, we are using a data set from a building [Pip+20] that includes ambient time series captured on seven floors, with four sensor-equipped zones on each floor. Using a network of n nodes/zones involved in the learning process, we arranged a zone-by-zone knowledge exchange. Each zone has its own knowledge of local temperature history and seeks to produce temperature forecasts via a decentralized online collaborative learning utilizing Algorithm 3. For practical efficiency of prediction accuracy, the learning model is not convex; rather, each node is embedded with a neural network, specifically a two-layer long-short-time-memory (LSTM, see [HS97]) network that processes entire sequences of data, followed by a fully connected layer. Denote the output of the predictive model with its weight vector $\boldsymbol{x}$ for data sequence e by $\mathcal{M}_{\boldsymbol{x}}(e)$ and its ground truth by $\Phi(e)$. Consider the ℓ_1 norm as the objective loss function to optimize :

$$\delta(\mathcal{M}_{\boldsymbol{x}}(e), \Phi(e)) = \begin{cases} \frac{1}{2}\left(\mathcal{M}_{\boldsymbol{x}}(e) - \Phi(e)\right)^2 & \text{if } |\mathcal{M}_{\boldsymbol{x}}(e) - \Phi(e)| \leq 1 \\ |\mathcal{M}_{\boldsymbol{x}}(e) - \Phi(e)| - \frac{1}{2} & \text{otherwise.} \end{cases} \tag{4.2}$$

Consider the constraint set $\mathcal{K} = \left\{\boldsymbol{x} \in \mathbb{R}^d, , \|\boldsymbol{x}\|_1 \leq r\right\}$, where $\boldsymbol{x}$ is the model's weight, d its dimension and $r = 1$. The (normalized) loss incurred by the data of agent i is $\frac{1}{|\mathcal{B}_i^t|}\sum_{e\in\mathcal{B}_i^t}\delta\left(\mathcal{M}_{\boldsymbol{x}}(e), \Phi(e)\right)$. The global loss function incurred by the overall data is

$$F^t(\boldsymbol{x}) = \frac{1}{|\cup_{i=1}^n \mathcal{B}_i^t|}\sum_{e\in\cup_{i=1}^n\mathcal{B}_i^t}\delta\left(\mathcal{M}_{\boldsymbol{x}}(e), \Phi(e)\right)$$

that can be written as $F^t(\boldsymbol{x}) = \frac{1}{n}f_i^t(\boldsymbol{x})$ where $f_i^t = \frac{1}{|\mathcal{B}_i^t|}\sum_{e\in\mathcal{B}_i^t}\delta\left(\mathcal{M}_{\boldsymbol{x}}(e), \Phi(e)\right)$.

Remark 10.

Even though the function δ is a convex function, the function F^t is not convex due to the non-convexity of the model function $\mathcal{M}_{\boldsymbol{x}}$.

Remark 11.

In situations where the optimization problem is non-convex, it may not be possible to uniquely define an optimal solution. This is because non-convex optimization problems can have multiple local minima, and it is not always clear which of these minima is the global minimum. As a result, it is difficult to evaluate the regret with hindsight measurement, as the term $\min_{\boldsymbol{x}^*} \sum_{t=1}^{T} F^t(\boldsymbol{x}^*)$ in the regret definition assumes the ability to find a unique global optimal vector $\boldsymbol{x}^*$. Therefore, in order to evaluate the performance of an algorithm in a non-convex optimization setting, it may be more appropriate to focus on the accuracy of the predictions made by the algorithm, rather than on the regret with hindsight measurement. This can help to provide a more meaningful assessment of the algorithm's performance, even in situations where it is not possible to uniquely define an optimal solution.

4.2.2 Observation and discussion

Prediction performance.

We are given a set $\mathcal{B}$ of prediction points between the 21^{st} and 24^{th} of April and n zones within one configuration (the number of zones n is equivalent to the number of agents). We use the mean absolute error (MAE equation 4.3) and mean square error (MSE equation 4.4) to measure between the prediction $\hat{y}_{i,e}$ of agent i for data point e and the ground truth y_e.

$$\text{MAE}_i = \frac{1}{n\,|\mathcal{B}|} \sum_{i=1}^{n} \sum_{e \in \mathcal{B}} |\hat{y}_{i,e} - y_e| \tag{4.3}$$

$$\text{MSE}_i = \frac{1}{n\,|\mathcal{B}|} \sum_{i=1}^{n} \sum_{e \in \mathcal{B}} (\hat{y}_{i,e} - y_e)^2 \tag{4.4}$$

Figure 4.11 demonstrates empirically the convergence of loss values for different network sizes. However, as seen in table 4.3 and 4.4, increasing the number of nodes in a network does not always lead to better online performance. In fact, the 7-node configuration achieved the lowest MSE (0.64) and MAE (0.77) values for floors 6 and 7. When 3 extra peers from floor 7 joined the group, there was a 40% drop in MSE and 20% reduction in MAE for floor 6 zonal models. In contrast, adding zonal nodes from floor 7 to a 10-node group resulted in a 19% and 25% increase in MSE and MAE values, respectively. This may be due to the fact that the top floor of a building has a different thermal variation than the other floors.

Topology	Metric	Mean	Var	Max	Min
cycle	MAE	1.09	0.48	1.80	0.56
cycle	MSE	0.78	0.21	1.09	0.52
complete	MAE	**0.77**	**0.38**	**1.47**	0.27
complete	MSE	**0.64**	**0.20**	**1.04**	0.39
line	MAE	0.81	0.53	1.95	**0.24**
line	MSE	0.66	0.28	1.26	**0.34**

Tab. 4.3: Impact of Topology on 7 learners configuration.

Topology	Metric	Mean	Var	Max	Min
cycle	MAE	1.51	1.46	6.16	0.36
cycle	MSE	0.94	0.38	1.90	0.48
complete	MAE	**1.26**	**0.82**	3.64	**0.32**
complete	MSE	**0.85**	**0.27**	**1.50**	**0.42**
line	MAE	1.38	0.91	3.17	0.50
line	MSE	0.90	0.35	**1.66**	0.49

Tab. 4.4: Impact of Topology on 13 learners configuration.

Effect of Network Topology.

We examined the impact of network topology on learning in 7-node and 13-node configurations with complete, cycle, and line graph structures containing 28, 7, and 6 edges and 78, 13, and 12 edges, respectively. The results in Table 4.3 and Table 4.4 show that the complete graph consistently resulted in the lowest prediction error and mean absolute error for both configurations. However, we also noticed that the line graph performed better than the cycle graph in some cases and had an error margin of about 10% compared to the complete configuration.

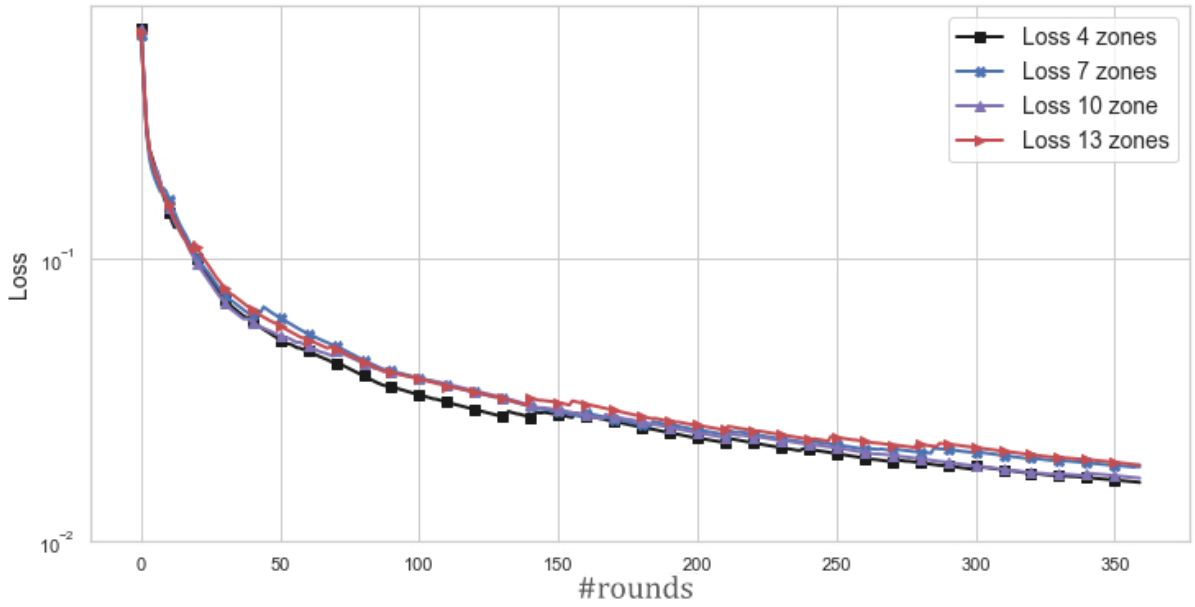


Fig. 4.11: Loss values of different network size on complete topology *(Plot on log-scale)*

5 Conclusion

5.1 Summary

In conclusion, we have proposed two decentralized online convex optimization algorithms, Decentralized Meta Frank Wolfe and Random Walk Meta Frank Wolfe, which are well-suited for addressing optimization problems in the edge paradigm. These algorithms excel at real-time decision making and adaptability to dynamic environments while also maintaining low computational overhead and offering robustness to failures, making them ideal for various edge applications such as smart cities, connected vehicles, and industrial IoT.

To ensure the effectiveness of our algorithms, we provided theoretical bounds on the regret, which measures the deviation from the optimal solution. We also conducted experiments and implemented a proof of concept on devices with low resources to validate their practicality. We even tested their performance when the assumption of convexity was relaxed in a smart building application for temperature forecasting, and our algorithms still performed well.

When considering the trade-offs between Decentralized Meta Frank Wolfe and Random Walk Meta Frank Wolfe, it is important to consider both the performance and computational complexity of each algorithm. Decentralized Meta Frank Wolfe is more stable when considering the topology of the graph, but it requires more communication and computation compared to Random Walk Meta Frank Wolfe. On the other hand, Random Walk Meta Frank Wolfe is more sensitive to the graph topology but requires less computation and communication. One key advantage of Random Walk Meta Frank Wolfe over Decentralized Meta Frank Wolfe is that it does not require synchronization among the nodes in the network. This can be especially useful in scenarios where synchronization is difficult or impossible to achieve, such as in distributed systems with limited bandwidth or high levels of interference. However, it is worth noting that the current version of Random Walk Meta Frank Wolfe has not yet been analyzed for its ability to handle stochastic gradient estimates using variance reduction, which is a feature that Decentralized Meta Frank Wolfe possesses. Ultimately, the optimal choice between the two algorithms will depend on the specific requirements and constraints of the system, including the network topology and technological restrictions.

It is worth noting that the suitability of our proposed algorithms for the edge paradigm also depends on the learning model being used. The chosen learning model should be able to fit within the constraints of the edge, such as limited resources and communication bandwidth, in order for the optimization approach to be practically effective. Therefore, it is important to consider both the optimization algorithm and the learning model when designing solutions for the edge paradigm.

5.2 Perspectives

Overall, our proposed algorithms make a valuable contribution to the field of online decentralized convex optimization, and we believe that further research and development in this area has the potential to unlock even more of their capabilities. By exploring different classes of functions, dynamic graphs, and specific extensions, we can continue to improve the performance and efficiency of our algorithms and make them even more suitable for use in the edge paradigm. In addition, it may also be useful to study the best technologies and protocols for communication in order to fully enable decentralization and make our algorithms even more effective and efficient in distributed systems.

5.2.1 Different Classes of Functions

One possibility is to study different classes of functions, such as submodular functions, with our algorithms ([Fuj05; Bac13; Bac15]). Submodular functions are known to be good at capturing certain types of structure in optimization problems, and Frank Wolfe based methods have been shown to be particularly well-suited for submodular optimization ([MHK18; CHK18]). Incorporating submodular functions into our algorithms has the potential to greatly improve their performance and efficiency, making them even more suitable for use in the edge paradigm.

5.2.2 Decentralized optimization with dynamic graphs

Another direction for future work is to study decentralized online optimization with dynamic graphs. Dynamic graphs, where the structure of the graph changes over time, are common in many real-world systems, such as in distributed sensor networks or in social networks. Optimization algorithms that are able to adapt to these dynamic environments can be more effective and efficient at solving optimization problems in the edge paradigm. There has been some work on decentralized optimization with dynamic graphs [Zhu+21], but there is still much room for further research and development in this area.

5.2.3 Experiments in edge computing applications

A third direction for future work is to conduct more interesting sets of experiments to demonstrate the utility of the algorithms in various edge computing applications. One specific application that could be explored is recommendation systems, where the algorithms could be used to improve the efficiency and effectiveness of personalized recommendations. Another application that could be explored is video streaming, where the algorithms could be used to optimize the delivery of video content to users in real-time. There are many other potential applications of the algorithms in edge computing, such as IoT, smart cities, and industrial automation, and conducting experiments in these areas will be valuable for understanding the full potential of the algorithms.

5.2.4 Improving the Random Walk Meta Frank Wolfe Algorithm

There are several ways that the performance and capabilities of the Random Walk Meta Frank Wolfe algorithm can be improved and extended.

Variance reduction techniques.
One possibility for improving the performance of the Random Walk Meta Frank Wolfe algorithm is to use variance reduction techniques, which can help to reduce the variability of stochastic gradient estimates and improve the robustness of the algorithm in convex optimization. For example, [LZ16; HL16] discuss various variance reduction techniques that can be used in convex optimization. These techniques have the potential to improve the efficiency and accuracy of the algorithm, particularly in cases where the data is noisy or the optimization problem is complex.

Privacy-preserving techniques.
Another extension that could be explored is the use of privacy-preserving techniques. In distributed systems, it is often important to protect the privacy of individuals, particularly when sensitive data is involved. The paper "Private Weighted Random Walk Stochastic Gradient Descent" ([AR21]) proposes a method for making the stochastic gradient descent (SGD) algorithm privacy-preserving by replacing the standard uniform sampling of data points with a weighted random walk. This technique aligns with the concept of local differential privacy, which seeks to prevent the disclosure of sensitive information about individual data points while still allowing for useful analytics to be conducted on the data as a whole. Essentially, local differential privacy helps to preserve the privacy of individuals while still enabling the collection and analysis of data for the benefit of the larger group.

Local oracle updates.
There is an opportunity to enhance the Random Walk Meta Frank Wolfe algorithm by finding ways to minimize oracle update exchanges in the network. In the current implementation of the algorithm, a node updates its set of oracles by exchanging them with one of its neighbors in the network. However, this process can be inefficient and may not be suitable for handling a large number of online rounds, as the number of oracles grows proportionally to the number of rounds. By addressing this issue, it may be possible to improve the efficiency and scalability of the algorithm, making it more suitable for use in large-scale distributed systems.

These extensions have the potential to improve the performance and efficiency of the Random Walk Meta Frank Wolfe algorithm, making it even more suitable for use in distributed systems and the edge paradigm. By continuing to research and develop these and other extensions, it is possible to unlock even more of the capabilities of this powerful optimization algorithm.

5.3 Exploring the benefits and implications of bringing machine learning to the edge

While technical improvements and extensions are crucial for the advancement of our algorithms, it is also essential to consider the larger picture and the potential impact of bringing machine learning to the edge. As decentralized optimization algorithms become increasingly sophisticated, they have the potential to transform the way machine learning is utilized in a range of applications.

As a reminder, machine learning at the edge involves using machine learning algorithms and models on devices located at the edge of a network, which can provide numerous benefits for various applications. One advantage is the ability to process data in real-time on devices at the edge of the network, improving processing speed and efficiency, especially in applications with low latency requirements like autonomous vehicles or industrial automation. Edge computing can also reduce the strain on network infrastructure and save energy by minimizing the amount of data transmitted over the network, as an IoT device with a machine learning model can process and analyze data locally rather than sending it to the cloud.

Edge computing can also enable the development of new types of applications and services not possible with traditional centralized architectures. For example, edge computing could be used to create smart cities where a network of IoT devices uses machine learning to collect and analyze data in real-time to optimize urban systems such as traffic management and energy use.

In addition, edge computing can improve accessibility and enable offline capabilities. For example, an IoT device with a machine learning model can analyze and process data locally even when offline, allowing it to continue providing services like environmental monitoring or navigation assistance in situations with limited or unreliable internet connectivity. Edge computing can also make it easier for individuals to access and use machine learning algorithms and models, even without specialized hardware or infrastructure.

In conclusion, edge computing can offer benefits such as faster and more efficient data processing, reduced energy consumption, and offline capabilities. It can also improve accessibility and make it easier to use machine learning algorithms and models. However, it is important to consider the ethical and ecological implications of edge computing, including concerns about privacy and security and the potential for bias in algorithms and models. In order to mitigate the potential risks of bringing machine learning to the edge, it is important to carefully design and implement machine learning systems that are transparent, accountable, and fair. This involves ensuring that machine learning algorithms are trained on representative and unbiased data, and that their outcomes are monitored and evaluated for potential biases. It also involves implementing strong privacy and security measures to protect sensitive personal information and prevent cyber attacks. In addition, it is important to consider the ethical implications of using machine learning at the edge and ensure that it is being used for legitimate and beneficial purposes.

Bibliography

[AGL17] M. Akbari, B. Gharesifard, and T. Linder. „Distributed Online Convex Optimization on Time-Varying Directed Graphs“. In: *IEEE Transactions on Control of Network Systems* 4.3 (2017), pp. 417–428 (cit. on p. 20).

[AQA17] A. Mokhtari, Q. Ling, and A. Ribeiro. „Network Newton Distributed Optimization Methods“. In: *IEEE Trans. Sig. Proc.* 65.1 (Jan. 2017), pp. 146–161 (cit. on p. 20).

[AR21] Ghadir Ayache and Salim El Rouayheb. „Private Weighted Random Walk Stochastic Gradient Descent“. In: *IEEE Journal on Selected Areas in Information Theory* 2.1 (2021), pp. 452–463 (cit. on pp. 17, 89).

[Bac13] Francis Bach. „Learning with Submodular Functions: A Convex Optimization Perspective“. In: *Foundations and Trends® in Machine Learning* 6.2-3 (2013), pp. 145–373 (cit. on p. 88).

[Bac15] Francis Bach. *Submodular Functions: from Discrete to Continous Domains*. 2015 (cit. on p. 88).

[Baj+17] D. Bajovic, D. Jakovetic, N. Krejic, and N.K. Jerinkic. „Newton-like Method with Diagonal Correction for Distributed Optimization“. In: *SIAM Journal on Optimization* 27.2 (2017), pp. 1171–1203 (cit. on p. 20).

[BCN18] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. „Optimization Methods for Large-Scale Machine Learning“. In: *Siam Reviews* 60.2 (2018), pp. 223–311 (cit. on pp. 2, 4).

[BPC11] Stephen Boyd, Neal Parikh, and Eric Chu. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Foundations and Trends in Machine Learning, 2011 (cit. on p. 20).

[Bra+22] Gábor Braun, Alejandro Carderera, Cyrille W Combettes, et al. *Conditional Gradient Methods*. Nov. 2022. arXiv: 2211.14103 [math.OC] (cit. on p. 4).

[Bru+16] Pierre Brunisholz, Etienne Duble, Franck Rousseau, and Andrzej Duda. „WalT: A Reproducible Testbed for Reproducible Network Experiments“. In: *IEEE INFOCOM International Workshop on Computer and Networking Experimental Research Using Testbeds (CNERT)*. CNERT 2016 Best Demo Award. San Francisco, USA, Apr. 2016 (cit. on pp. 75, 78).

[Cao+13] Y. Cao, W. Yu, W. Ren, and G. Chen. „An Overview of Recent Progress in the Study of Distributed Multi-Agent Coordination“. In: *IEEE Transactions on Industrial Informatics* 9.1 (2013), pp. 427–438 (cit. on p. 7).

[Che+17] Tianyi Chen, Yanning Shen, Qing Ling, and Georgios B. Giannakis. „Online learning for “thing-adaptive” Fog Computing in IoT“. In: *2017 51st Asilomar Conference on Signals, Systems, and Computers*. 2017, pp. 664–668 (cit. on p. 9).

[Che+18] Lin Chen, Christopher Harshaw, Hamed Hassani, and Amin Karbasi. „Projection-Free Online Optimization with Stochastic Gradient: From Convexity to Submodularity“. In: *Proceedings of the 35th International Conference on Machine Learning*. 2018, pp. 814–823 (cit. on pp. 19, 29, 48, 65, 73).

[CHK18] Lin Chen, Hamed Hassani, and Amin Karbasi. „Online continuous submodular maximization“. In: *Proc. 21st International Conference on Artificial Intelligence and Statistics (AISTAT)*. 2018 (cit. on pp. 13, 14, 25, 88).

[Cla10] Kenneth L. Clarkson. „Coresets, Sparse Greedy Approximation, and the Frank-Wolfe Algorithm“. In: *ACM Trans. Algorithms* 6.4 (Sept. 2010) (cit. on p. 4).

[DJ14] D. Mateos-Nunez and J. Cortes. „Distributed Online Convex Optimization Over Jointly Connected Digraphs“. In: *IEEE Transactions on Network Science and Engineering* 1.1 (2014), pp. 23–37 (cit. on p. 20).

[DMP16] L. Deori, K. Margellos, and M. Prandini. „On decentralized convex optimization in a multi-agent setting with separable constraints and its application to optimal charging of electric vehicles“. In: *IEEE Conference on Decision and Control (CDC)*. 2016, pp. 6044–6049 (cit. on p. 7).

[DS16] Paolo Di Lorenzo and Gesualdo Scutari. *NEXT: In-Network Nonconvex Optimization*. 2016 (cit. on p. 15).

[Dub22] Etienne Dublé. *WalT, https://walt-project.liglab.fr*. 2022 (cit. on p. 75).

[EB12] E. Candès and B. Recht. „Exact Matrix Completion via Convex Optimization“. In: *Communication of ACM* 55 (2012), pp. 111–119 (cit. on p. 7).

[EE20] E. Hazan and E. Minasyan. „Faster Projection-free Online Learning“. In: *Proc. of 33rd Conference on Learning Theory*. Vol. 125. 2020, pp. 1877–1893 (cit. on p. 27).

[EMR17] M. Eisen, A. Mokhtari, and A. Ribeiro. „Decentralized Quasi-Newton Methods“. In: *IEEE Transactions on Signal Processing* 65.10 (2017), pp. 2613–2628 (cit. on p. 20).

[Fuj05] Satoru Fujishige. „Submodular functions and optimization“. In: *Annals of Discrete Mathematics* 58 (2005), pp. 3–10 (cit. on p. 88).

[FW56] Marguerite Frank and Philip Wolfe. „An algorithm for quadratic programming“. In: *Naval Research Logistics Quarterly* 3.1-2 (1956), pp. 95–110. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1002/nav.3800030109` (cit. on p. 4).

[Has70] W.K. Hastings. „Monte Carlo sampling methods using Markov chains and their applications“. In: *Biometrika* 57.1 (1970), pp. 97–109 (cit. on pp. 25, 53).

[Haz16] Elad Hazan. „Introduction to online convex optimization“. In: *Foundations and Trends® in Optimization* 2.3-4 (2016), pp. 157–325 (cit. on pp. 6, 16, 19, 20, 28).

[HBJ18] Lie He, An Bian, and Martin Jaggi. „Cola: Decentralized linear learning“. In: *Advances in Neural Information Processing Systems*. 2018, pp. 4536–4546 (cit. on pp. 8, 9, 24).

[HBM19a] Hadrien Hendrikx, Francis Bach, and Laurent Massoulié. „Accelerated decentralized optimization with local updates for smooth and strongly convex objectives“. In: *The 22nd International Conference on Artificial Intelligence and Statistics*. 2019, pp. 897–906 (cit. on p. 20).

[HBM19b] Hadrien Hendrikx, Francis Bach, and Laurent Massoulié. „An accelerated decentralized stochastic proximal algorithm for finite sums“. In: *Advances in Neural Information Processing Systems*. 2019, pp. 954–964 (cit. on p. 20).

[HCM13] S. Hosseini, A. Chapman, and M. Mesbahi. „Online distributed optimization via dual averaging“. In: *52nd IEEE Conference on Decision and Control*. 2013, pp. 1484–1489 (cit. on p. 20).

[HCM16] S. Hosseini, A. Chapman, and M. Mesbahi. „Online Distributed Convex Optimization on Dynamic Networks“. In: *IEEE Transactions on Automatic Control* 61.11 (2016), pp. 3545–3550 (cit. on p. 20).

[HK12] Elad Hazan and Satyen Kale. „Projection-Free Online Learning“. In: *Proceedings of the 29th International Coference on International Conference on Machine Learning*. ICML'12. Edinburgh, Scotland: Omnipress, 2012, pp. 1843–1850 (cit. on p. 7).

[HL16] Elad Hazan and Haipeng Luo. *Variance-Reduced and Projection-Free Stochastic Optimization*. 2016 (cit. on p. 89).

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. „Long Short-Term Memory“. In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780 (cit. on p. 82).

[IAG08] I.D. Schizas, A. Ribeiro, and G.B. Giannakis. „Consensus in Ad Hoc WSNs With Noisy Links-Part I: Distributed Estimation of Deterministic Signals“. In: *Trans. Sig. Proc.* 56.1 (2008), pp. 350–364 (cit. on p. 20).

[Jag+14] Martin Jaggi, Virginia Smith, Martin Takác, et al. „Communication-efficient distributed dual coordinate ascent“. In: *Advances in neural information processing systems*. 2014, pp. 3068–3076 (cit. on p. 20).

[Jag11] Martin Jaggi. „Sparse convex optimization methods for machine learning“. In: 2011 (cit. on p. 4).

[Jag13] Martin Jaggi. „Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization“. In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by Sanjoy Dasgupta and David McAllester. Vol. 28. Proceedings of Machine Learning Research 1. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 427–435 (cit. on p. 4).

[Jin+20] Yibo Jin, Lei Jiao, Zhuzhong Qian, et al. „Provisioning Edge Inference as a Service via Online Learning“. In: *2020 17th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. Como, Italy: IEEE Press, 2020, pp. 1–9 (cit. on p. 9).

[JMX15] D. Jakovetic, J. M. F. Moura, and J. Xavier. „Linear Convergence Rate of a Class of Distributed Augmented Lagrangian Algorithms“. In: *IEEE Transactions on Automatic Control* 60.4 (2015), pp. 922–936 (cit. on p. 20).

[Joh+08] Bjorn Johansson, Tamás Keviczky, Mikael Johansson, and Karl Henrik Johansson. „Subgradient methods and consensus algorithms for solving convex optimization problems“. In: *47th IEEE Conference on Decision and Control*. 2008, pp. 4185–4190 (cit. on pp. 15, 25).

[JXM14] D. Jakovetic, J. Xavier, and J. M. F. Moura. „Fast Distributed Gradient Methods“. In: *IEEE Transactions on Automatic Control* 59.5 (2014), pp. 1131–1146 (cit. on p. 20).

[KJR15] A. Koppel, F. Y. Jakubiec, and A. Ribeiro. „A Saddle Point Algorithm for Networked Online Convex Optimization“. In: *IEEE Transactions on Signal Processing* 63.19 (2015), pp. 5149–5164 (cit. on p. 20).

[KQW16] K. Yuan, Q. Ling, and W. Yin. „On the Convergence of Decentralized Gradient Descent“. In: *SIAM J. on Optimization* 26.3 (2016), pp. 1835–1854 (cit. on p. 20).

[Kri09] Alex Krizhevsky. *Learning multiple layers of features from tiny images*. Tech. rep. 2009. Chap. 3 (cit. on p. 65).

[Lec+98] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. „Gradient-based learning applied to document recognition“. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324 (cit. on p. 65).

[Li+20] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. „Federated learning: Challenges, methods, and future directions“. In: *IEEE Signal Processing Magazine* 37.3 (2020), pp. 50–60 (cit. on pp. 9, 24, 30).

[Lia+17] Xiangru Lian, Ce Zhang, Huan Zhang, et al. „Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent“. In: *Advances in Neural Information Processing Systems*. 2017, pp. 5330–5340 (cit. on pp. 8, 9, 24).

[LNR17] S. Lee, A. Nedic, and M. Raginsky. „Stochastic Dual Averaging for Decentralized Online Optimization on Time-Varying Communication Graphs“. In: *IEEE Transactions on Automatic Control* 62.12 (2017), pp. 6407–6414 (cit. on p. 20).

[LV10] Z. Liu and L. Vandenberghe. „Interior-point method for nuclear norm approximation with application to system identification“. In: *SIAM J. Matrix Anal.* 31.3 (2010), pp. 1235–1256 (cit. on p. 7).

[LZ16] Guanghui Lan and Yi Zhou. „Conditional Gradient Sliding for Convex Optimization“. In: *SIAM Journal on Optimization* 26 (Jan. 2016), pp. 1379–1409 (cit. on p. 89).

[Ma+15] Chenxin Ma, Virginia Smith, Martin Jaggi, et al. „Adding vs. averaging in distributed primal-dual optimization“. In: *International Conference on Machine Learning*. 2015, pp. 1973–1982 (cit. on p. 20).

[Ma+17] Chenxin Ma, Jakub Konečnỳ, Martin Jaggi, et al. „Distributed optimization with arbitrary local solvers“. In: *Optimization Methods and Software* 32.4 (2017), pp. 813–848 (cit. on p. 20).

[McM+21] H Brendan McMahan et al. „Advances and Open Problems in Federated Learning“. In: *Foundations and Trends® in Machine Learning* 14.1 (2021) (cit. on pp. 9, 24, 30).

[MHK18] Aryan Mokhtari, Hamed Hassani, and Amin Karbasi. „Conditional Gradient Method for Stochastic Submodular Maximization: Closing the Gap“. In: *Conference on Artificial Intelligence and Statistics*. Vol. 84. 2018, pp. 1886–1895 (cit. on pp. 29, 88).

[Mit+22] Angan Mitra, Nguyen Kim Thang, Tuan-Anh Nguyen, Denis Trystram, and Paul Youssef. „Online Decentralized Frank-Wolfe: From theoretical bound to applications in smart-building“. In: *GloTS 2022 - Global IoT Conference*. Dublin, Ireland, June 2022, pp. 1–12 (cit. on p. 82).

[Ned+09] A. Nedic, A. Olshevsky, A. Ozdaglar, and J. N. Tsitsiklis. „On Distributed Averaging Algorithms and Quantization Effects“. In: *IEEE Trans. on Automatic Control* 54 (2009), pp. 2506–2517 (cit. on p. 20).

[NLR15] A. Nedic, S. Lee, and M. Raginsky. „Decentralized online optimization with global objectives and local communication“. In: *2015 American Control Conference (ACC)*. 2015, pp. 4497–4503 (cit. on p. 20).

[NOS17] Angelia Nedić, Alex Olshevsky, and Wei Shi. „Achieving Geometric Convergence for Distributed Optimization Over Time-Varying Graphs“. In: *SIAM Journal on Optimization* 27.4 (2017), pp. 2597–2633. eprint: `https://doi.org/10.1137/16M1084316` (cit. on p. 15).

[Pip+20] Manisa Pipattanasomporn, Gopal Chitalia, Jitkomut Songsiri, et al. „CU-BEMS, smart building electricity consumption and indoor environmental sensor datasets“. In: *Scientific Data* (2020) (cit. on p. 82).

[QL18] Guannan Qu and Na Li. „Harnessing Smoothness to Accelerate Distributed Optimization“. In: *IEEE Transactions on Control of Network Systems* 5.3 (Sept. 2018), pp. 1245–1260 (cit. on p. 15).

[QL20] G. Qu and N. Li. „Accelerated Distributed Nesterov Gradient Descent“. In: *IEEE Transactions on Automatic Control* 65.6 (2020), pp. 2566–2581 (cit. on p. 20).

[Rei+19] A. Reisizadeh, A. Mokhtari, H. Hassani, and R. Pedarsani. „An Exact Quantized Decentralized Gradient Descent Algorithm“. In: *IEEE Transactions on Signal Processing* 67.19 (2019), pp. 4934–4947 (cit. on p. 7).

[RM51] Herbert Robbins and Sutton Monro. „A Stochastic Approximation Method“. In: *The Annals of Mathematical Statistics* 22.3 (1951), pp. 400–407 (cit. on p. 4).

[RNB05] M. G. Rabbat, R. D. Nowak, and J. A. Bucklew. „Generalized consensus computation in networked systems with erasure links“. In: *IEEE Workshop on Signal Processing Advances in Wireless Communications*. 2005, pp. 1088–1092 (cit. on p. 20).

[Sca+19] K. Scaman, F. Bach, S. Bubeck, Y.T. Lee, and L. Massoulié. „Optimal Convergence Rates for Convex Distributed Optimization in Networks“. In: *Journal of Machine Learning Research* 20.159 (2019), pp. 1–31 (cit. on p. 20).

[SG08] Matthew Streeter and Daniel Golovin. „An Online Algorithm for Maximizing Submodular Functions“. In: *Advances in Neural Information Processing Systems*. Ed. by D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou. Vol. 21. Curran Associates, Inc., 2008 (cit. on p. 14).

[SJ16] Andrea Simonetto and Hadi Jamali-Rad. „Primal recovery from consensus-based dual decomposition for distributed convex optimization“. In: *Journal of Optimization Theory and Applications* 168.1 (2016), pp. 172–197 (cit. on pp. 15, 25).

[SJ18] S. Shahrampour and A. Jadbabaie. „Distributed Online Optimization in Dynamic Environments Using Mirror Descent“. In: *IEEE Transactions on Automatic Control* 63.3 (2018), pp. 714–725 (cit. on p. 20).

[SSY18] Tao Sun, Yuejiao Sun, and Wotao Yin. „On Markov Chain Gradient Descent“. In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS'18. Montréal, Canada: Curran Associates Inc., 2018, pp. 9918–9927 (cit. on pp. 17, 55, 57).

[TLR12] K. I. Tsianos, S. Lawlor, and M. G. Rabbat. „Consensus-based distributed optimization: Practical issues and applications in large-scale machine learning“. In: *50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. 2012, pp. 1543–1550 (cit. on p. 7).

[Vap98] V. Vapnik. *Statistical learning theory*. Wiley, 1998 (cit. on pp. 2, 7).

[Wai+17] H. Wai, J.Lafond, A. Scaglione, and E. Moulines. „Decentralized Frank–Wolfe algorithm for convex and nonconvex problems“. In: *IEEE Transactions on Automatic Control* 62.11 (2017), pp. 5522–5537 (cit. on pp. 14, 15, 20, 25, 31, 32).

[WAP18] W. Zheng, A. Bellet, and P. Gallinari. „A Distributed Frank—Wolfe Framework for Learning Low-Rank Matrices with the Trace Norm“. In: *Machine Learning* 107.810 (2018), pp. 1457–1475 (cit. on p. 7).

[Xie+20] Jiahao Xie, Zebang Shen, Chao Zhang, Boyu Wang, and Hui Qian. „Efficient Projection-Free Online Methods with Stochastic Recursive Gradient.“ In: *AAAI Conference on Artificial Intelligence*. 2020, pp. 6446–6453 (cit. on pp. 18, 29, 65, 67, 73).

[Yan+13] F. Yan, S. Sundaram, S. V. N. Vishwanathan, and Y. Qi. „Distributed Autonomous Online Learning: Regrets and Intrinsic Privacy-Preserving Properties“. In: *IEEE Transactions on Knowledge and Data Engineering* 25.11 (2013), pp. 2483–2493 (cit. on p. 20).

[Zha+17] W. Zhang, P. Zhao, W. Zhu, S.C.V. Hoi, and T. Zhang. „Projection-Free Distributed Online Learning in Networks“. In: *Proceedings of the 34th International Conference on Machine Learning*. 2017, pp. 4054–4062 (cit. on pp. 17–20, 24, 65, 67, 73).

[Zhu+21] Junlong Zhu, Qingtao Wu, Mingchuan Zhang, Ruijuan Zheng, and Keqin Li. „Projection-free Decentralized Online Learning for Submodular Maximization over Time-Varying Networks“. In: *Journal of Machine Learning Research* 22.51 (2021), pp. 1–42 (cit. on p. 88).

[Zin03] Martin Zinkevich. „Online Convex Programming and Generalized Infinitesimal Gradient Ascent“. In: 2 (Apr. 2003) (cit. on pp. 6, 19).

List of Figures

List of Tables