Thesis for The Degree of Licentiate of Engineering

Decentralized Constrained Optimization: a Novel Convergence Analysis

Firooz Shahriari-Mehr

Department of Computer Science and Engineering
Chalmers University of Technology | University of Gothenburg
Gothenburg, Sweden, 2023

Decentralized Constrained Optimization:
a Novel Convergence Analysis

Firooz Shahriari-Mehr

ISSN 1652-876X

Department of Computer Science and Engineering
Division of Data Science and AI
Chalmers University of Technology | University of Gothenburg
SE-412 96 Göteborg,
Sweden
Phone: +46(0)31 772 1000

Printed by Chalmers Digitaltryck,
Gothenburg, Sweden 2023.

To my Family and Friends.

Decentralized Constrained Optimization: a Novel Convergence Analysis

Firooz Shahriari-Mehr

Department of Computer Science and Engineering
Chalmers University of Technology | University of Gothenburg

Abstract

One reason for the spectacular success of machine learning models is the appearance of large datasets. These datasets are often generated by different computational units or agents and cannot be processed on a single machine due to memory and computing limitations. Moreover, the data may contain sensitive information and hence should not be shared among different machines Distributed systems can handle these problems by keeping the data locally and leveraging the cooperation of agents over a communication graph. This thesis is focused on a family of distributed systems, where the objective is to minimize a sum of locally held functions subject to local constraints, called the Decentralized Constrained Optimization Problem (DCOP). This problem is of significant importance as it arises in various real-world applications such as distributed sensor networks, decentralized control, and multi-agent systems. Our main concern is to develop efficient first-order decentralized optimization algorithms to solve the DCOP.

The first part of our contributions is the development of a generic algorithmic framework for the DCOP, which we refer to as *Double Averaging and Gradient Projection (DAGP)*, where each local function and local constraint is only accessible by a particular agent. This algorithm is presented in our first paper, and we both theoretically and numerically demonstrate its competitive convergence rate. Our work is the first to consider distributed constraints in the DCOP. In the second paper, we revealed the importance of the DCOP and the intuitions behind the update rules of the DAGP algorithm. Moreover, by building upon the initial work and addressing its limitations, we developed a more comprehensive and general methodology to provide a rate of convergence for optimization algorithms. This methodology is called "*Aggregate lower bounding,*" which foregoes the need for Lyapunov functions or decaying step-sizes. This novel convergence analysis methodology is our second contribution.

Keywords

Constrained optimization, Convergence analysis, Convex optimization, Distributed optimization, Decentralized optimal transport, Multi-agent systems

List of Publications

Appended publications

This thesis is based on the following publications:

[**Paper I**] **F. Shahriari-Mehr**, D. Bosch, and A. Panahi, *"Decentralized constrained optimization: Double averaging and gradient projection."* *IEEE Conference on Decision and Control (CDC), 2021, pp. 2400-2406..*

[**Paper II**] **F. Shahriari-Mehr** and Ashkan Panahi, *"Double Averaging and Gradient Projection: Convergence Guarantees for Decentralized Constrained Optimization."* *Submitted to IEEE Transactions on Automatic Control, under review.*

Acknowledgment

First of all, I would like to express my deepest gratitude to my main supervisor, Ashkan Panahi, for his invaluable guidance, support, and encouragement throughout my research journey. His expertise, insight, and patience have been critical to my success, and I am truly grateful for his great mentoring. I would also like to thank my co-supervisor Morteza Haghir Chehreghani for his fruitful input and encouragement. I greatly appreciate the valuable inputs provided by my examiner, Devdatt Dubhashi, during presentations and meetings, which have helped me improve my work and grow professionally.

I would like to express my appreciation to my colleagues in the DSAI division; David, Mehrdad, Arman, Peter, Linus, Adam, Emil, Solrun, Hanna, Niklas, Hampus, Lovisa, Emilio, Tobias K., Newton, Christopher, Anton, Alexander, Filip, Daniel, Mena, Tobias N., Markus, Denitsa, Lena, Juan, Vladimir, Milad, Shirin, and Mohammad. Their expertise and support have been invaluable, and I am proud to work alongside such talented individuals. I have gained valuable knowledge and insights from our conversations and presentations at Chalmers ML seminars. Thank you for making our division an inspiring and enjoyable place to work.

I would also like to thank my family, Kamran, Jaleh, Goshtasb, Kaveh and Ramina for their complete support and encouragement throughout my academic pursuits. Their love and understanding have been the foundation of my success, and I am forever grateful for their unwavering support.

This research has been supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation, project "Efficient Data Representation and Machine Learning over Next Generation Networks," grant number 37200114. So, thank you WASP for generously supporting my Ph.D. studies.

Firooz Shahriari-Mehr
Gothenburg, May 2023

Contents

Part I

Introductory chapters

Chapter 1

Introduction

Supervised learning is one of the core techniques used in machine learning and artificial intelligence applications. It provides models from labeled training data, which are to make accurate predictions on new, unseen data. Empirical Risk Minimization (ERM) is a highly popular framework for supervised learning, which relies on a finite-sum optimization problem, associated with the training data set, the set of candidate models, and the learning task. The success of the ERM framework generally hinges on the complexity of the candidate models and the size of the training dataset. The common wisdom in the field of machine learning is that the larger the dataset and the more complex the models are (large-scale supervised learning), the better performance on unseen data becomes. However, resource limitations and privacy concerns prevent processing large amounts of data on a single machine for large-scale problems, making distributed optimization methods essential in addressing these challenges.

In distributed optimization, two setups are considered based on the communication network. The first is the centralized setup, which includes a master node that coordinates all other nodes. In this setup, issues such as master bottleneck and master failure may arise since all nodes communicate with the master node. The second setup is known as the decentralized setup, in which there is no master node. In this configuration, connecting numerous computational units together increases the overall computing capacity, resulting in faster wall-clock convergence. In this thesis, our focus is on solving finite-sum optimization problems in a decentralized setup.

Convex optimization is a framework to find the minimum of a convex function over a convex set, which arises in various applications such as machine learning, signal processing, and control theory. To solve such optimization problems, a wide range of iterative algorithms have been proposed, each addressing different challenges in the problem, such as handling constraints or large-scale problems, improving convergence rate, and more. These methods can be classified as either first-order or second-order methods, depending on the information they use from the objective function. Moreover, they are dual-based optimization methods that rely on the dual problem and utilize

iterative techniques, which may take advantage of any underlying structure in the dual form, such as separability. In this thesis. we focus on convex finite-sum optimization problems. We consider the general case, where each term in the objective function is convex and the optimization is over a constraint set that is the intersection of several convex sets. We explore both algorithmic solutions based on first-order information and theoretical convergence guarantees in this thesis. In the following, we will provide an overview of the upcoming chapters.

Organization of the thesis

In Chapter 2, we will review the fundamentals of convex optimization problems. We will proceed by categorizing iterative optimization methods and briefly explaining each category. Lastly, we take a brief look at the Lyapunov-based analysis of optimization algorithms, the common practice of finding quadratic Lyapunov functions, and the underlying challenges, motivating alternative approaches. In Chapter 3, we will examine supervised learning and the empirical risk minimization problem. We demonstrate that ERM exhibits a finite-sum structure, and how a decentralized approach addresses issues related to the scale of the problem as well as the privacy issues. In Chapter 4, we will conduct an in-depth examination of decentralized optimization algorithms and delve into the current challenges faced in this area. The purpose of this section is to emphasize that there are still unexplored aspects and gaps that necessitate additional research efforts. In Chapter 5, we will provide an overview of the papers included in this thesis and outline our contributions to the field of decentralized optimization. Finally, in Chapter 6, we will suggest several potential future research directions for interested readers.

Chapter 2

Convex optimization

Convex optimization is a branch of mathematical optimization that deals with the minimization of convex functions over convex sets. It is used in a variety of fields, including machine learning, signal processing, finance, and control systems. This chapter provides an overview of the convex optimization problem formulation, various methods for solving this problem, and the convergence guarantees associated with these methods. A more comprehensive treatment of the subject can be found in [1]–[4].

2.1 Problem definition

We start by defining convex functions and convex sets.

Definition 1 (Convex set)**.** A set $S \subseteq \mathbb{R}^n$ is called convex if the line segment connecting any two points in the set is entirely contained within the set, i.e. for all $\mathbf{x}, \mathbf{y} \in S$ and any $\alpha \in [0, 1]$

$$\alpha\mathbf{x} + (1 - \alpha)\mathbf{y} \in S.$$

Definition 2 (Convex function)**.** A function $f : \mathbb{R}^n \to \mathbb{R}$ is called convex if the line segment connecting any two points on the graph of the function lies above or on the graph itself, i.e. for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and any $\alpha \in [0, 1]$

$$f(\alpha\mathbf{x} + (1 - \alpha)\mathbf{y}) \leq \alpha f(\mathbf{x}) + (1 - \alpha)f(\mathbf{y}).$$

A convex optimization problem can be mathematically represented by

$$\mathbf{x}^* = \arg\min_{\mathbf{x} \in S} \ f(\mathbf{x}), \tag{2.1}$$

where S is a convex set and f is a convex function. During this thesis, we assume the optimization problems are *solvable*, which means there exists at least one optimal solution $\mathbf{x}^* \in S$ such that the above optimization problem achieves its finite optimal value f^* at this optimal point $f^* = f(\mathbf{x}^*)$, that is for all $\mathbf{x} \in S$ we have $f(\mathbf{x}^*) \leq f(\mathbf{x})$.

The constraint set S in the optimization problem (2.1) is written as a general convex set showing all possible values the optimization variable can have. In standard optimization problems, the constraint set contains all points which are in the intersection of several equality and inequality constraints as

$$h^v(\mathbf{x}) = 0, \qquad v = 1, \dots, c_e$$

$$g^v(\mathbf{x}) \leq 0, \qquad v = 1, \dots, c_i$$

where c_e and c_i are the numbers of equality and inequality constraints.

The optimization problem in (2.1) is called *primal* problem. The *dual* problem is formulated by introducing Lagrange multipliers for the constraints, which are denoted as $\boldsymbol{\lambda}$ for inequality constraints and $\boldsymbol{\nu}$ for equality constraints. Then, the Lagrangian function is defined as

$$L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = f(\mathbf{x}) + \sum_{v=1}^{c_i} \lambda^v g^v(\mathbf{x}) + \sum_{v=1}^{c_e} \nu^v h^v(\mathbf{x}).$$

Given the Lagrangian, the dual optimization problem is defined as

$$\max_{\boldsymbol{\lambda} \geq \mathbf{0}, \boldsymbol{\nu}} \quad \min_{\mathbf{x}} \quad L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}). \tag{2.2}$$

The relations between the optimal values of the primal minimization problem in (2.1) and the dual maximization problem in (2.2) are governed by weak and strong duality theorems. Weak duality indicates that the optimal value of the dual problem serves as a lower bound for the optimal value of the primal problem. On the other hand, strong duality establishes the conditions under which the optimal values of both the primal and dual problems are equal.

Numerous iterative algorithms exist to determine the optimal solutions and values for optimization problems in either equation (2.1) or (2.2). These algorithms differ with respect to several factors, including the characteristics of the objective functions (smooth/non-smooth, convex/non-convex), the nature of the constraints (constrained/unconstrained, specific constraint structures), the problem type (primal or dual), the optimization setup (single-machine, centralized, or decentralized), and the rate at which they converge to the optimal solution (linear or sub-linear).

The terms linear and sublinear rates of convergence are frequently used throughout this thesis, making it beneficial to define them in Section 2.2 before discussing and comparing the convergence guarantees of various iterative algorithms in Section 2.4. Furthermore, it is essential to include the definitions of smooth and strongly convex functions in Section 2.3, as they are consistently mentioned in this thesis.

2.2 Convergence rates

Convergence rate is a performance measure that quantifies how fast a sequence $\{\mathbf{s}_k\}$ converges to $\mathbf{s}^*$. In the following, we will define different convergence rates in the ascending order [5].

- **sublinear rate of convergence** implies that the algorithm's progress slows down over time at a gradually pace. Therefore, the terms in the sequence does not change for large values of k. i.e.

$$\lim_{k\to\infty} \frac{\|\mathbf{s}_{k+1} - \mathbf{s}^*\|}{\|\mathbf{s}_k - \mathbf{s}^*\|} = 1,$$

- **linear rate of convergence:** If $\|\mathbf{s}_{k+1} - \mathbf{s}^*\| \leq \lambda \|\mathbf{s}_k - \mathbf{s}^*\|$, for $\lambda \in (0, 1)$, the error $\|\mathbf{s} - \mathbf{s}^*\|$ reduces at a constant proportion with each iteration, leading to a steady and consistent progress towards the optimal solution.

- **superlinear rate of convergence:** A sequence $\mathbf{s}_k$ convergences with a superlinear rate of convergence if

$$\lim_{k\to\infty} \frac{\|\mathbf{s}_{k+1} - \mathbf{s}^*\|}{\|\mathbf{s}_k - \mathbf{s}^*\|} = 0.$$

- **higher order convergence rates (quadratic, cubic, ...):** In the above definitions the order of $\|\mathbf{s}_k - \mathbf{s}^*\|$ is one. The $p-$order convergence rate is defined as

$$\lim_{k\to\infty} \frac{\|\mathbf{s}_{k+1} - \mathbf{s}^*\|}{\|\mathbf{s}_k - \mathbf{s}^*\|^p} \leq M,$$

  for a positive constant M. For $p = 2$, and $p = 3$, the rate of convergence is called Quadratic and Qubic, respectively.

In the realm of optimization algorithms, our goal is to demonstrate the convergence of multiple sequences and establish their respective rates of convergence. Firstly, we need to determine the rate of convergence for the objective error sequence, known as the *optimality gap* and denoted by $f(\mathbf{x}_k) - f(\mathbf{x}^*)$, as it converges to zero. When dealing with constrained algorithms, it is necessary to establish a convergence rate for the sequence that measures the distance between $\mathbf{x}_k$ and the constraint set, which also should converge to zero and is called *feasibility gap*. In cases where a unique global solution exists, such as when the objective function is strongly convex, we can also demonstrate the convergence of the optimization method's iterates, $\mathbf{x}_k$, to the optimal solution, $\mathbf{x}^*$. Lastly, for non-convex objective functions with multiple local minima, it is essential to prove the convergence of the objective function's gradient, denoted by $\nabla f(\mathbf{x}_k)$, to zero.

2.3 Smooth and strongly-convex functions

Smooth and strongly convex functions offer benefits in terms of faster convergence, unique global minimum, better convergence guarantees, and ease of implementation. They allow the derivation of tighter convergence bounds for optimization algorithms. These bounds provide insights into the algorithm's performance and help choose appropriate hyperparameters such as the step size, ensuring that the algorithm converges to the optimal solution with a specific convergence rate.

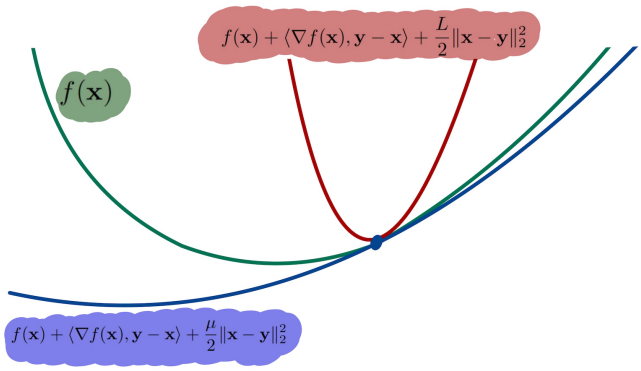

Figure 2.1: Smooth and strongly-convex functions.

Definition 3 ($L-$Smooth function)**.** A differentiable function f is $L-$smooth if its derivative is $L-$Lipschitz continuous. For a convex and $L-$smooth function f, for all $\mathbf{x}, \mathbf{y}$ in its domain, we have

$$f(\mathbf{y}) \leq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{L}{2}\|\mathbf{x} - \mathbf{y}\|_2^2.$$

Definition 4 ($\eta-$Strongly convex function)**.** A function f is $\eta-$strongly-convex, if for every $\mathbf{x}$ and $\mathbf{y}$ in its domain, we have

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{\mu}{2}\|\mathbf{x} - \mathbf{y}\|_2^2.$$

The condition of strong convexity does not need a function to be differentiable, and for non-smooth functions, the sub-gradient takes the place of the gradient.

To provide a more tangible understanding of the above definitions, let us examine Figure 2.1. This illustration demonstrates that when a function is $L-$smooth, it can be upper bounded by a quadratic function with a positive curvature L. In contrast, $\eta-$strong convexity indicates the existence of a quadratic lower bound with curvature η. As a result, it is straightforward to show that a strongly convex function over $\mathbb{R}^n$ has a unique global optimal point [3].

2.4 Iterative algorithms

Iterative methods are widely used to solve convex optimization problems. The most common iterative methods can be classified into three main categories based on the nature of the information from the objective function they use and whether they are solving the primal or dual problem: first-order methods, second-order methods, and dual-based methods. We will discuss each category in this section, including their definition, specific algorithms in that class, and their convergence properties.

2.4.1 First-order methods

First-order methods use the gradient (first-order information) of the objective function. These methods are particularly well-suited for large-scale or high-dimensional problems, as they generally have lower computational complexity per iteration compared to second-order methods. First-order methods typically exhibit a linear convergence rate for strongly convex problems and a sublinear convergence rate for general differentiable convex problems. *Gradient Descent (GD)* is a well-known instance of first-order methods, which updates the optimization variables as

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mu \nabla f(\mathbf{x}_k), \tag{2.3}$$

where μ is a design parameter known as step size. It should be selected appropriately to ensure convergence of the sequence $\{\mathbf{x}_k\}_{k\in\mathbb{N}}$ to the optimal solution $\mathbf{x}^*$. For $L-$smooth and $\eta-$strongly-convex functions, it has been shown that the optimality gap will decay linearly with the order of $\left(\frac{\kappa_f - 1}{\kappa_f + 1}\right)^k$, where $\kappa_f = L/\eta$ is the condition number of f, and the step size is chosen as $\mu = 1/L$. For general convex and smooth objective functions, this algorithm enjoys sublinear rate of convergence $\mathcal{O}(1/k)$.

There are numerous gradient descent-based optimization algorithms that can be broadly categorized based on stochastic approaches, accelerated techniques, and distributed setups. These categories include various techniques that address specific challenges or improve the convergence rate and efficiency of the optimization process. Below, we provide a brief description of each category with one example, and we refer the reader to [1], [4] for further details.

Stochastic optimization algorithms

Stochastic Gradient Descent (SGD) is a popular optimization algorithm in this category, which is widely used in practice, e.g. in training machine learning models. Stochastic methods are often used when the objective function f in (2.1) is the average of N terms, i.e. $f(\mathbf{x}) = \mathbb{E}_I[f_I(\mathbf{x})]$, where I is a uniform discrete random variable such that $\mathbb{P}(I = i) = 1/N$ for $i = 1, 2, \ldots, N$. Consequently, the optimization problem for this setup is called ***finite sum minimization***, and it is the case in machine learning applications where each term in the average is the loss function computed at a data point measuring the discrepancy between a model and the data point.

In machine learning problems, computing the true/full gradient of the loss function requires computing the gradient over the entire dataset, which can be computationally inefficient when the dataset size, N, is large. In response, stochastic methods estimate the true gradient by randomly selecting a small group (minibatch) of terms from the average and updating the optimization variables as

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mu \mathbf{g}_k, \tag{2.4}$$

where $\mathbf{g}_k$ is the estimate of the true gradient based on the selected terms and varies among different stochastic optimization algorithms. In the original

SGD algorithm, one term is randomly selected and $\mathbf{g}_k = \nabla f_{i_k}(\mathbf{x}_k)$, where i_k is distributed according to the distribution I. This estimate is *unbiased*. i.e. $\mathbb{E}[g_k] = \nabla f(\mathbf{x}_k)$, but it has a variance $\mathbb{E}[\|\mathbf{g}_k - \nabla f(\mathbf{x}_k)\|_2^2]$, leading to the convergence of the algorithm to a neighborhood of the optimal solution with a constant step size. The size of the neighborhood depends on the variance of gradient estimates. In order to achieve convergence to the exact optimal solution, several variance reduction techniques have been proposed. These techniques introduce additional variables that act as memory to keep track of the true gradient and reduce the variance, resulting in the possibility of using fixed step sizes and linear convergence rates. Examples of these techniques include SAG [6], SAGA [7], and SVRG [8].

Accelerated methods using momentum

Acceleration methods improve the speed of convergence by incorporating the gradients calculated in past interactions. Nesterov's Accelerated Gradient Descent (AGD) is an example of an accelerated method that incorporates a momentum term into the update rule. This algorithm updates the optimization variables $\mathbf{x}_k$ by introducing intermediary variables $\mathbf{y}_k$, with initialization $\mathbf{x}_0 = \mathbf{y}_0 \in \mathbb{R}^n$, as follows

$$\begin{aligned} \mathbf{y}_{k+1} &= \mathbf{x}_k - \frac{1}{L}\nabla f(\mathbf{x}_k) \\ \mathbf{x}_{k+1} &= (1+\beta)\mathbf{y}_{k+1} - \beta \mathbf{y}_k \end{aligned} . \tag{2.5}$$

The first equation is the simple GD update with $\mu = 1/L$. The second equation is where the momentum introduced since $\mathbf{x}_{k+1}$ depends on the two past values of $\mathbf{x}_k$ and $\mathbf{x}_{k-1}$.

The additional term $\beta(\mathbf{y}_{k+1} - \mathbf{y}_k)$ leads to faster convergence in comparison to GD ($\beta = 0$) with the same assumptions on the objective function. To be more concrete, the AGD convergence rate for $L-$smooth and $\eta-$strongly-convex functions with $\beta = \frac{\sqrt{\kappa_f}-1}{\sqrt{\kappa_f}+1}$ matches the following lower bound up to constants [4]

$$\|\mathbf{x}_k - \mathbf{x}^*\|_2^2 \geq \left(\frac{\sqrt{\kappa_f}-1}{\sqrt{\kappa_f}+1}\right)^{2k} \|\mathbf{x}_0 - \mathbf{x}^*\|_2^2. \tag{2.6}$$

We observe that the momentum term improves the dependence of convergence rate on condition number of function f from κ_f for GD to $\sqrt{\kappa_f}$ for AGD. Consequently, for *ill-conditioned* problems with large κ_f, such as statistical learning problems with small regularization parameters, accelerated methods are appropriate choices for an optimization algorithm. On the other hand, for general convex and $L-$smooth functions, the convergence rate is $\mathcal{O}(1/k^2)$. Heavy ball acceleration [9] is another acceleration method with weaker convergence guarantees. As a final remark, the acceleration idea can be applied to many basic optimization algorithms for achieving faster convergence in different settings, for example, the setting with a composite objective function that includes a non-smooth term [10], or accelerated versions of stochastic methods [11], [12].

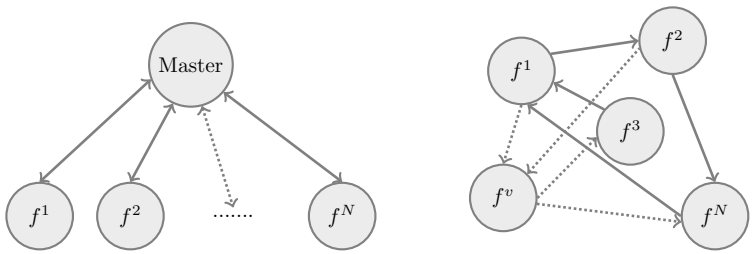

Figure 2.2: Distributed optimization setups: centralized versus decentralized.

Distributed optimization algorithms

All the above-mentioned algorithms run on a single computing machine. In some practical applications such as large-scale machine learning, sensor networks, and multi-agent systems, these *single-machine* methods become inefficient or infeasible due to privacy concerns or computational limitations. In response, distributed optimization algorithms are designed to solve optimization problems in distributed and parallel computing environments, where multiple agents or nodes work collaboratively to find a solution. For these algorithms, similar to the stochastic methods, the objective function has a finite sum structure, and the distributed unconstrained optimization problem can be written as

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \ \frac{1}{N} \sum_{v=1}^{N} f^v(\mathbf{x}). \tag{2.7}$$

Distributed optimization algorithms can be classified according to the communication network topology that connects the agents. In the following, we will present two common distributed setups, which are depicted in Figure 2.2.

- **Centralized or Master-Worker setup:** In this setup, there exists a master node that coordinates all the workers and updates the optimization variables. The communication network topology in this setup is a star-shaped network. Gradient descent can be adapted for this setup, resulting in an algorithm called *distributed gradient descent*. At iteration $k+1$ of this algorithm, the master node broadcasts the optimization variables $\mathbf{x}_k$ to the worker nodes. The worker nodes then compute their local gradients $\nabla f^v(\mathbf{x}_k)$ and send the local gradient information back to the master node. Finally, the master node updates the optimization variables by aggregating the local gradients received from the worker nodes as follows

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{\mu}{N} \sum_{v=0}^{N} \nabla f^v(\mathbf{x}_k). \tag{2.8}$$

  This algorithm distributes gradient computations across N worker nodes. Consequently, it achieves linear scaling of convergence time to a specified error. In a centralized setup, all nodes must communicate with the master node, which can become a bottleneck. Furthermore, if the master node

fails due to practical reasons such as system overheating, the algorithm cannot proceed to the next iteration.

- **Decentralized setup:** The decentralized setup addresses the issues of master bottleneck and failure. In this setup, no single master node is responsible for coordinating the worker nodes. Instead, nodes communicate directly with one another in a peer-to-peer manner over a general connected communication network. This approach eliminates the single point of failure and reduces communication bottlenecks. In decentralized optimization, each node maintains its own local copy of the optimization variables and updates them based on local computations and communications that depend on the specific optimization algorithms employed.

  Since each node has its own copy of the optimization variables, and the goal is to find the unique optimal solution of the distributed optimization problem in (2.7), the nodes must agree on one *consensus* solution at the end of the algorithm. In this regard, the optimization problem in (2.7) can be equivalently reformulated as the following *Decentralized optimization problem* if the communication network, represented by a graph, is strongly connected.

$$\begin{aligned} \{\mathbf{x}^{v,*}\} = \quad & \underset{\{\mathbf{x}^v\}_{v=1}^N}{\arg\min} \quad \frac{1}{N}\sum_{v=1}^{N} f^v(\mathbf{x}^v) \\ & \text{s.t.} \quad \mathbf{x}^v = \mathbf{x}^u \quad \forall (v,u) \in \mathcal{E} \end{aligned} \tag{2.9}$$

  where $\mathcal{E}$ is the set of all graph edges. *Decentralized Gradient Descent* [13] is the adapted version of gradient descent for the decentralized setup using the so-called gossip matrices [14]. In this algorithm, each node updates as

$$\mathbf{x}_{k+1}^v = \mathbf{x}_k^v - \sum_{u \in \mathcal{N}_{\text{in}}^v \cup \{v\}} w_{vu}\mathbf{x}_k^u - \mu \nabla f^v(\mathbf{x}_k^v), \tag{2.10}$$

  where w_{vu} is the weight associated to the communication link between nodes v and u. If the weights are designed so that their collection, called *Gossip matrix*, satisfies several assumptions, DGD will converge to the consensus optimal solution of (2.9) using diminishing step sizes. With constant step sizes, it will converge to a neighborhood of optimal solution, where the size of this neighborhood depends on the *distributed variance*

$$\frac{1}{N}\sum_{v=1}^{N} \|\nabla f^v(\mathbf{x}^*)\|_2^2.$$

  There are multiple challenges involved in the decentralized setup, and different algorithms are proposed in the literature to address these challenges. A more detailed discussion of decentralized optimization methods will be provided in Chapter 4.

2.4.2 Second-order methods

Second-order optimization methods are a class of algorithms that use second-order derivatives or their approximations to solve the optimization problem in (2.1). These methods include the Newton and Quasi-Newton methods [5]. Newton's method is known for its fast convergence, typically exhibiting quadratic convergence when the objective function is twice continuously differentiable and strongly convex. However, Newton's method requires computing and inverting the Hessian matrix, which can be computationally expensive and impractical for large-scale problems. Quasi-Newton methods are developed to address the computational challenges of Newton's method by approximating the Hessian matrix or its inverse. Broyden–Fletcher–Goldfarb-Shanno (BFGS) and Limited-memory BFGS algorithms [5] are popular Quasi-Newton methods.

Despite the convergence advantages of second-order methods, they have some limitations, such as the need to compute or approximate the Hessian matrix, which is a computationally expensive task. These methods may not be suitable for problems with non-smooth or non-convex objective functions. On the other hand, first-order methods only require gradient information, which is generally easier to compute. These methods are more widely applicable and can handle larger-scale problems, differentiable non-smooth functions, and distributed settings. Although first-order methods typically have slower convergence rates compared to second-order methods, their simplicity and computational efficiency make them the preferred choice in many applications.

2.4.3 Dual-based methods

The aforementioned algorithms address the optimization problem within the primal domain. However, alternative approaches exist that focus on the dual Lagrangian form of the problem in (2.2). Various dual-based optimization techniques are developed by employing existing methods for primal optimization and applying them to the dual maximization problem. For example, by applying gradient ascent or proximal point algorithm to the dual problem [2]. Similar to their primal counterparts, dual-based optimization algorithms can be classified as first-order or second-order, depending on the specific algorithm and the information used from the Lagrangian dual function.

Dual-based techniques are primarily employed to achieve computational benefits. For instance, by expressing the primal optimization problem in its Lagrangian form, the dual problem becomes constraint-free and smooth. Additionally, specific structures within the dual problem, such as separable optimization variables, can sometimes be exploited. However, dual-based methods necessitate first-order or second-order information from the Lagrangian dual function, which includes the conjugate of the objective function. Computing this information may be infeasible in certain applications. To conclude, deciding between primal-based or dual-based methods depends on the characteristics of the problem, but primal methods are more simple to understand and more robust to hyperparameters [15].

2.5 Lyapunov-based convergence analysis

In this section, we derive the convergence rate of the GD algorithm for L-smooth objective functions by employing the classical Lyapunov-based convergence analysis technique. From convexity and $L-$smoothness, we have

$$f(\mathbf{x}_k) \geq f(\mathbf{x}^*), \tag{2.11}$$

$$f(\mathbf{x}^*) \geq f(\mathbf{x}_k) + \langle \nabla f(\mathbf{x}_k), \mathbf{x}^* - \mathbf{x}_k \rangle, \tag{2.12}$$

$$f(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_k) + \langle \nabla f(\mathbf{x}_k), \mathbf{x}_{k+1} - \mathbf{x}_k \rangle + \frac{L}{2}\|\mathbf{x}_{k+1} - \mathbf{x}_k\|_2^2. \tag{2.13}$$

By combining the above inequalities and considering the GD update dynamics presented in Equation (2.3), we have

$$f(\mathbf{x}_{k+1}) - f(\mathbf{x}^*) + \frac{1}{\mu}\langle \mathbf{x}_{k+1} - \mathbf{x}_k, \mathbf{x}_{k+1} - \mathbf{x}^* \rangle - \frac{L}{2}\|\mathbf{x}_{k+1} - \mathbf{x}_k\|_2^2 \leq 0. \tag{2.14}$$

Then, using algebraic equations and completing the squares, we can simplify the above equation to

$$f(\mathbf{x}_{k+1}) - f(\mathbf{x}^*) + \frac{1}{2\mu}\left[\|\mathbf{x}_{k+1} - \mathbf{x}^*\|_2^2 - \|\mathbf{x}_k - \mathbf{x}^*\|_2^2\right] + \left(\frac{1}{2\mu} - \frac{L}{2}\right)\|\mathbf{x}_{k+1} - \mathbf{x}_k\|_2^2 \leq 0. \tag{2.15}$$

By choosing the appropriate step size $\mu \leq 1/L$, the last term is always positive; hence it can be removed from the inequality. For the GD algorithm, we can prove that the first term $f(\mathbf{x}_{k+1}) - f(\mathbf{x}^*)$ is a positive decreasing sequence, which shows the convergence of $f(\mathbf{x}_k)$ to the optimal value $f(\mathbf{x}^*)$ if the sequence converges to zero. To show convergence to the optimal value, we can sum the inequality in (2.15) over iterations from $k = 0$ to $K - 1$. This sum is valuable as it allows us to leverage the telescoping property of the difference between the Lyapunov function $L(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}^*\|_2^2$ computed at two successive iterations. By defining $\bar{\mathbf{y}} = \frac{1}{K}\sum_{k=0}^{K-1}\mathbf{x}_k$, and using Jenson's inequality, we have

$$f(\bar{\mathbf{y}}) - f(\mathbf{x}^*) \leq \frac{1}{K}\sum_{k=0}^{K-1} f(\mathbf{x}_k) - f(\mathbf{x}^*) \leq \frac{\|\mathbf{x}_0 - \mathbf{x}^*\|_2^2}{2\mu K},$$

which is the standard $\mathcal{O}(1/k)$ rate of convergence. This analysis is based on simplifying the given inequalities from assumptions and algorithm dynamics to the point that only two composite terms remain in the inequality. The first term is a positive-definite function Φ, which usually contains several positive terms showing the convergence of the algorithm to the optimal solution. The second composite term is in the form of the difference between a Lyapunov function computed at two successive iterations. To make it more clear, in this analysis, one seeks the functions Φ and L satisfying the following inequality

$$L(\mathbf{\Psi}_{k+1}) - L(\mathbf{\Psi}_k) + \Phi(\mathbf{\Psi}_k) \leq 0, \tag{2.16}$$

where $\mathbf{\Psi}$ represents the state vectors (optimization variables and possible auxiliary variables) describing the dynamics of the algorithm. Then, by summing over all iterations up to $K-1$, and using Jenson's inequality, the $\mathcal{O}(1/k)$ rate of convergence will be achieved. Finding the Lyapunov function L, and simplification of the inequalities to (2.16) is not a straightforward task for many optimization algorithms. In this thesis, our second contribution is the introduction of a new convergence analysis technique referred to as the "Aggregate Lower Bounding" methodology. With this approach, the reliance on a Lyapunov function in the analysis is removed.

2.6 Summary

This section provided an overview of optimization algorithms and classified them based on their different characteristics. We clarified why we are interested in the decentralized first-order optimization algorithms. In this chapter, we showed finding Lyapunov functions is not straightforward for all optimization algorithms and new methodologies will be of interest, which is the second part of our contributions presented in Paper II. In the subsequent chapters, first, we will introduce supervised learning as one of the most practical and significant problems solved using decentralized first-order methods. Then, in Chapter 4, we will review decentralized first-order optimization algorithms in detail.

Chapter 3

Supervised learning

In this chapter, we briefly introduce supervised learning and discuss classification and regression problems. Then, we review the concept of empirical risk minimization, which demonstrates the role of optimization in machine learning. For more details, we refer the reader to [16], [17].

3.1 Problem definition

Supervised learning is the procedure of training a predictive model, represented by its parameters, using a labeled dataset, i.e. a set of input-output pairs. After the training procedure, the trained model will be used for the prediction of unseen data. In the following, the main aspects of supervised learning are reviewed.

training set: A collection of N data points, where sample i is represented by a feature vector $\mathbf{x}_i$ and its corresponding target value or label y_i. Accordingly, the set $\{\mathbf{x}_i, y_i\}_{i=1}^{N_s}$ indicates the training set.

Classification or Regression: Classification deals with discrete possible values for y_i called labels or categories. Prediction of a handwritten digit is one example of classification. On the other hand, regression deals with continuous target values y_i. House price prediction is one example of a regression problem.

Model: Model $g(\mathbf{x}, \mathbf{w})$ is a function that maps input features $\mathbf{x}$ to outputs, and its parameters $\mathbf{w}$ are learned during the training process. Neural networks, logistic regression, linear regression, decision trees, and support vector machines are examples of machine learning models. These models can be adapted to either classification or regression tasks by modifying the output layer, loss function, or aggregation method, as needed.

loss function: Loss functions $l(g(\mathbf{x}_i, \mathbf{w}), y_i)$ quantify the difference between the predicted outputs of a model and the actual target values. By appropriately selecting the loss function based on either classification or regression tasks and minimizing the average loss computed for all training samples, the model parameters can be computed by solving the so-called Empirical Risk Minimization

(ERM) optimization problem. Binary cross-entropy, categorical cross-entropy, and hinge loss are examples of loss functions commonly used in classification tasks, whereas mean squared error, mean absolute error, and Huber loss are frequently employed in regression tasks [16].

3.2 Empirical Risk Minimization

ERM is the problem of minimizing the average loss on the training dataset, with the hope that the model will likely generalize well to unseen data. However, it is essential to be cautious about overfitting, which occurs when the model becomes too specialized to the training data and fails to perform well on new data. To address overfitting, regularization techniques are often used in conjunction with ERM, adding a penalty term to the loss function to encourage simpler models that generalize better. Therefore, the general ERM with regularization term $r(\mathbf{w})$ can be written as

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \quad \frac{1}{N}\sum_{i=1}^{N} l(g(\mathbf{x}_i, \mathbf{w}), y_i) + \lambda r(\mathbf{w}), \tag{3.1}$$

where λ is the regularization parameter. In this thesis, our focus is not on statistical machine learning problems and the design of the model, loss function, or dataset. For instance, we are not concerned with determining the best model and loss function for a specific dataset. We assume that the training dataset, model, and loss function are already given. Our objective is to find the optimal model parameters by solving the ERM problem. This problem has a finite-sum structure, allowing us to leverage efficient, scalable stochastic, and distributed optimization algorithms to find the optimal parameters effectively.

The optimization framework in Equation (3.1) encompasses different types of optimization problems by various choices of the objective function. Binary classification with logistic regression loss and ℓ_2- regularization, i.e.

$$l(g(\mathbf{x}_i, \mathbf{w}), y_i) = \log\left(1 + \exp^{-y_i \mathbf{x}_i^T \mathbf{w}}\right)$$

$$r(\mathbf{w}) = \|\mathbf{w}\|_2^2,$$

is $2\lambda-$strongly convex and $L-$smooth, where

$$L = 2\lambda + \frac{1}{4N}\lambda_{\max}\left(\sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^T\right).$$

Support vector machines with the following hinge loss are convex but non-smooth [18].

$$l(g(\mathbf{x}_i, \mathbf{w}), y_i) = \max\left\{0, 1 - y_i \mathbf{x}_i^T \mathbf{w}\right\}$$

$$r(\mathbf{w}) = 0$$

ℓ_1-regularized least squares regression, called *Lasso*, is smooth but not strongly convex with the following loss and regularization functions.

$$l(g(\mathbf{x}_i, \mathbf{w}), y_i) = (\mathbf{x}_i^T \mathbf{w} - y_i)^2$$

$$r(\mathbf{w}) = ||\mathbf{w}||_1$$

Finally, when choosing neural networks as the training model, the optimization problem becomes non-convex due to the compositional nature of the neural networks and using non-linear activation functions.

3.3 Summary

In this section, we examined the supervised learning problem and investigated the ERM as a central optimization problem for determining the optimal model parameters. We noted that ERM possesses a finite-sum structure. Therefore, for large-scale problems when the size of the training set or optimization variables is large, stochastic optimization methods can be employed for faster wall-clock convergence. Moreover, in situations where data samples are distributed or generated across a network of nodes and raw data sharing is not allowed due to privacy (e.g. private local training data) or resource constraints (e.g. memory limitations because all training samples cannot fit into one big data center), distributed optimization algorithms can be employed. The following chapter will offer an overview of distributed optimization algorithms, focusing specifically on decentralized approaches.

Chapter 4

Decentralized optimization

Decentralized optimization algorithms are those that solve the finite-sum minimization problem in a decentralized manner without relying on a central authority. In this chapter, we will provide an overview of decentralized optimization algorithms. We will discuss the challenges in decentralized optimization and how different optimization algorithms address these challenges.

4.1 Decentralized problem

Nowadays, the training data is generated on edge devices, for example, in federated learning applications such as next word prediction [19]. When local data is private or cannot be stored on a single machine due to communication or computation limitations, the problem can be addressed by solving ERM with a decentralized approach. ERM is an instance of the general Decentralized Constrained Optimization Problem (DCOP), which can be mathematically represented by

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \quad \frac{1}{M}\sum_{v=1}^{M} f^v(\mathbf{x}) \qquad \text{s.t.} \quad \mathbf{x} \in \bigcap_{v=1}^{N} S^v. \tag{4.1}$$

In this problem, each agent or node has its own local objective function $f^v(\mathbf{x})$ and local constraint set S^v. Equation (4.1) presents the general constrained problem, which is the case in applications such as the decentralized version of support vector machines [20], [21].

To solve DCOP in a decentralized manner, the agents communicate along a strongly-connected communication network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. This thesis focuses on first-order decentralized methods with directed communication networks such that each agent v can send its local solution and gradient information to its out-neighbors $\mathcal{N}_{\text{out}}^v$, the set of all nodes receiving information from node v. Moreover, it can receive local solutions and gradient information from all its in-neighbors $\mathcal{N}_{\text{in}}^v$ and compute their weighted average. The weights serve as design parameters, and the collection of these weights in a single matrix forms the so-called *gossip* matrix. To clarify, assume there exists a communication

link between node u and node v, indicated by the directed edge $(v, u) \in \mathcal{E}$. A weight, w_{vu}, is assigned to this communication link (the process for designing these weights will be discussed in the next paragraph). The gossip matrix $\mathbf{W}$ comprises all of these assigned weights and is expressed as $\mathbf{W} = [w_{vu}]$. Hence, the non-zero elements of a gossip matrix exhibit the same sparsity pattern as the adjacency matrix of the communication network. Using the graph Laplacian matrix, a gossip matrix $\mathbf{W}$ can possess either a zero row sum structure $\mathbf{W1} = \mathbf{0}$, a zero column sum structure $\mathbf{1}^T\mathbf{W} = \mathbf{0}^T$, or both zero row sum and zero column sum structures.

Gossip matrices are designed such that the algorithm converges to a *consensus* solution, that is $\mathbf{x}^v = \mathbf{x}^u = \mathbf{x}$, for all $v, u \in \mathcal{V}$. Moreover, this consensus point should be the *optimal* solution, i.e. $\mathbf{x} = \mathbf{x}^*$, and satisfy the optimality condition:

$$\mathbf{0} \in \sum_{v=1}^{N} \left(\partial I_{S^v}(\mathbf{x}^*) + \nabla f^v(\mathbf{x}^*)\right), \tag{4.2}$$

where $\partial I_{S^v}(\mathbf{x}^*)$ represents the normal cone to the set S^v at the point $\mathbf{x}^*$, which is defined as the subdifferential of the indicator function of S^v at $\mathbf{x}^*$.[1] In summary, the design of a decentralized optimization algorithm necessitates the selection of suitable hyperparameters and gossip matrices to ensure that all nodes converge toward a consensus and optimal solution.

4.2 Decentralized algorithms

Decentralized optimization algorithms can be classified based on various characteristics. One way to categorize them is by determining whether they solve the general constrained optimization problem, as in Equation (4.1), or its unconstrained counterpart, as in Equation (2.7). This feature is not specific to decentralized setup, and it is general. Another distinguishing feature of these algorithms is the communication links between nodes, which can be either uni-directional or bi-directional. Uni-directional links are represented by directed graphs, while bi-directional communication links are depicted using undirected graphs. Given that some common optimization methods solving a constrained problem involve an extra projection step onto the constraint set, and that the challenges between constrained and unconstrained optimization problems are fairly alike, we will first explore algorithms that do not address constraints. Afterward, we will link these insights to the constrained problem.

4.2.1 Decentralized Unconstrained Optimization methods

In this subsection, we will initially examine the decentralized unconstrained optimization algorithms over undirected graphs[2] . The discussion will include

[1] The indicator function takes the value of 0 for elements that belong to the constraint set S and $+\infty$ for elements that do not belong to S. The subdifferential is the set of all subgradients.

[2] When dealing with an undirected communication graph, it is relatively easy to construct gossip matrices that fulfill both the zero row sum and zero column sum requirements by

what we refer to as the *distributed variance* challenge and its resolution through the gradient tracking approach. Following this argument, we will explore the challenges associated with directed communication networks and the methods employed to address them.

Decentralized Gradient Descent (DGD) [13]

DGD is the simplest possible decentralized algorithm in which the local optimization variables in the kth iteration are updated as

$$\mathbf{x}_{k+1}^v = \mathbf{x}_k^v - \sum_{u \in \mathcal{N}_{\text{in}}^v \cup \{v\}} w_{vu} \mathbf{x}_k^u - \underbrace{\mu \nabla f^v(\mathbf{x}_k^v)}_{\text{greedy}}, \tag{4.3}$$

where w_{vu} are the elements of the so-called gossip matrix $\mathbf{W}$. The fixed-point iterations of this algorithm, taking into account convergence to a consensus solution, and assuming the gossip matrix $\mathbf{W}$ possesses a zero row sum structure, can be simplified to

$$\nabla f^v(\mathbf{x}) = \mathbf{0} \qquad \forall v \in \mathcal{V}.$$

This condition shows that the algorithm will converge to optimal solution $\mathbf{x}^*$ only if $\mathbf{x}^*$ is the minimizer of all local objective functions. In heterogeneous settings, this is not possible, and the algorithm converges to a neighborhood of the optimal solution, where the size of the neighborhood depends on the *distributed variance* calculated as [22]

$$\frac{1}{N} \sum_{v=1}^{N} \|\nabla f^v(\mathbf{x}^*)\|_2^2.$$

This justifies referring to the last term in Equation (4.3) as greedy alignment because each node is approximately minimizing its own local objective function by aligning the average solution in the direction $-\nabla f^v(\mathbf{x}_k^v)$. Therefore, to ensure the convergence of this algorithm to the optimal solution, a decaying step size is necessary [13], [23].

Decaying step size consistently results in slower convergence rates. For instance, in the context of DGD, the achievable rate concerning the optimality gap, or objective error, is $\mathcal{O}(1/\sqrt{K})$ for general convex and smooth functions [13]. When it comes to smooth and strongly convex functions, the rate is $\mathcal{O}(1/K)$. As such, employing a fixed step size is typically more appealing since it enhances the convergence rate from sublinear to linear. Thus, in order to employ a fixed step size to achieve faster convergence rates, we need to first understand the primary cause of the distributed variance that results in the utilization of decaying step sizes. Then, we can investigate potential solutions to tackle this issue.

By examining the update rule in (4.3) and considering fixed point iteration again, one can observe that if we replace $\nabla f^v(\mathbf{x}_k^v)$ by its average over all local

using the graph Laplacian matrix. If a gossip matrix satisfies both structures, the analysis becomes more straightforward in comparison to a gossip matrix with either zero row sum or zero column sum structure.

objective functions $\sum_{v=1}^{N} \nabla f^v(\mathbf{x}_k^v)$, the resulting algorithm will converge to an exact optimizer of DCOP using an appropriate fixed step size. For this replacement, all nodes must have access to the gradient of all local objective functions, which is not possible in decentralized setups. In response, several decentralized optimization algorithms are proposed in the literature, tracking the average of local gradients in a decentralized way. Many methods are based on Dynamic Average Consensus (DAC) protocol [24], which tracks the average of time-varying signals (local gradients in our case) over a communication network.

DIGing [25], EXTRA [23], and NEXT [26]

These methods rely on gradient tracking techniques, which allow them to employ fixed step sizes and attain linear convergence to the precise optimal solution of the decentralized optimization problem. As an example, the DIGing algorithm integrates the DAC protocol with DGD dynamics for the purpose of tracking the average gradient. This algorithm updates the optimization variables by

$$\begin{aligned} \mathbf{x}_{k+1}^v &= \mathbf{x}_k^v - \sum_{u \in \mathcal{N}_{\text{in}}^v \cup \{v\}} w_{vu} \mathbf{x}_k^u - \mu \mathbf{d}_k^v, \\ \mathbf{d}_{k+1}^v &= \mathbf{d}_k^v - \sum_{u \in \mathcal{N}_{\text{in}}^v \cup \{v\}} w_{vu} \mathbf{d}_k^u + \nabla f^v(\mathbf{x}_{k+1}^v) - \nabla f^v(\mathbf{x}_k^v), \end{aligned} \tag{4.4}$$

where $\mathbf{d}^v$ variables estimate the global average gradient, i.e.

$$\lim_{k \to \infty} \|\mathbf{d}_k^v - \frac{1}{N} \sum_{v=1}^{N} \nabla f^v(\mathbf{x}_k^v)\|_2^2 = 0, \tag{4.5}$$

if

$$\lim_{k \to \infty} \|\nabla f^v(\mathbf{x}_{k+1}^v) - \nabla f^v(\mathbf{x}_k^v)\|_2^2 = 0,$$

which demonstrates that the change in the gradient of the local objective functions should be small after a large number of iterations. This algorithm is "approximately" equal to the single-machine GD in 2.3, which means the trajectories for these two algorithms are close to each other. To show this, by defining $\bar{\mathbf{x}}_k = \frac{1}{N} \sum_{v=1}^{N} \mathbf{x}_k^v$, summing (4.4) over all nodes, considering (4.5), adding and removing $\frac{1}{N} \sum_{v=1}^{N} \nabla f^v(\bar{\mathbf{x}}_k)$, we have

$$\bar{\mathbf{x}}_{k+1} = \bar{\mathbf{x}}_k - \frac{1}{N} \sum_{v=1}^{N} \nabla f^v(\bar{\mathbf{x}}_k) \underbrace{+ \tfrac{1}{N} \sum_{v=1}^{N} (\nabla f^v(\bar{\mathbf{x}}_k) - \nabla f^v(\mathbf{x}_k^v))}_{\text{error}},$$

where the last term indicates the difference between the single-machine GD and DIGing algorithms and makes it clear why DIGing is approximately equal to the single-machine GD. If the algorithm converges to a consensus solution, which happens due to the first weighted averaging, the error term will go to zero [27].

Push-sum DGD [28], [29]

In the aforementioned algorithms, we assumed an undirected communication graph and a gossip matrix $\mathbf{W}$ with both zero row sum and zero column sum structures. For directed communication networks, it is only possible to construct gossip matrices with either a zero row sum structure, represented by $\mathbf{W}$, or a zero column sum structure, represented by $\mathbf{Q}$, using input or output Laplacian matrices. The challenge with directed communication graphs is the convergence point of the gossip averaging iteration. To illustrate the challenge, let us analyze the gossip averaging iterations, where the objective is to find the average of initial values at each node by executing the following algorithm

$$\mathbf{x}_{k+1}^v = \mathbf{x}_k^v - \sum_{u \in \mathcal{N}_{\text{in}}^v \cup \{v\}} q_{vu} \mathbf{x}_k^u.$$

In this algorithm, considering $\mathbf{Q}$ with zero column sum structure, the node values will converge to the initial sum $\sum_{v=1}^N \mathbf{x}^0$ multiplied by a constant π^v. In [27], by using the Perron-Frobenius theorem [30], it is shown that these constants differ among the nodes and correspond to the elements of the so-called Perron vector $\boldsymbol{\pi}$ of the gossip matrix $\mathbf{Q}\boldsymbol{\pi} = \boldsymbol{\pi}$. The push-sum idea [31] involves iteratively estimating these constants by introducing auxiliary variables $\mathbf{y}$, which are updated similarly to the $\mathbf{x}$ variables and initialized by $\mathbf{1}$.

$$y_{k+1}^v = y_k^v - \sum_{u \in \mathcal{N}_{\text{in}}^v \cup \{v\}} q_{vu} y_k^u.$$

Then, by using the Perron-Frobenius theorem [30] and considering one-dimensional optimization variables $x^v \in \mathbb{R}^1$ for simplicity of equations, we can write

$$\begin{aligned}
\lim_{k\to\infty} \mathbf{x}_k &= \lim_{k\to\infty} (\mathbf{I} - \mathbf{Q})^k \mathbf{x}_0 = \boldsymbol{\pi}\mathbf{1}^T\mathbf{x}_0 = \mathbf{1}^T\mathbf{x}_0\boldsymbol{\pi} \\
\lim_{k\to\infty} \mathbf{y}_k &= \lim_{k\to\infty} (\mathbf{I} - \mathbf{Q})^k \mathbf{y}_0 = \boldsymbol{\pi}\mathbf{1}^T\mathbf{y}_0 = \mathbf{1}^T\mathbf{y}_0\boldsymbol{\pi} \\
\lim_{k\to\infty} z^v &= \frac{\left(\mathbf{1}^T\mathbf{x}_0\right)\pi_v}{\left(\mathbf{1}^T\mathbf{y}_0\right)\pi_v} = \frac{1}{N}\sum_{v=1}^N x_0^v,
\end{aligned}$$

where $\mathbf{x}_k$ and $\mathbf{y}_k$ are the stack of x^v and y^v variables, for all v, in one vector. The above equations show the convergence of the ratio $z^v = x^v/y^v$ to the average of initial values by estimating the Perron vector elements π^v through the updates of $\mathbf{y}$ variables. Now, the push-sum protocol can be applied to the DGD algorithm for solving the general decentralized optimization over directed graphs:

$$\begin{aligned}
\mathbf{u}_{k+1}^v &= \mathbf{u}_k^v - \sum_{u \in \mathcal{N}_{\text{in}}^v \cup \{v\}} w_{vu} \mathbf{x}_k^u, \\
y_{k+1}^v &= y_k^v - \sum_{u \in \mathcal{N}_{\text{in}}^v \cup \{v\}} w_{vu} y_k^u, \\
\mathbf{z}_{k+1}^v &= \mathbf{x}_k^v / y_k^v \ , \\
\mathbf{x}_{k+1}^v &= \mathbf{u}_{k+1}^v \ - \mu_k \nabla f^v(\mathbf{z}_{k+1}^v),
\end{aligned}$$

where μ_k is a diminishing sequence of step sizes. The necessity for a diminishing step size arises due to the algorithm's similarity to the DGD algorithm, as it lacks access to gradient information from other nodes. Consequently, to address this issue, the ratio variables $\mathbf{z}^v$ will converge to a neighborhood of the optimal solution. Gradient tracking approaches have been introduced.

Push-DIGing [25], DEXTRA [32], and SONATA [33]

Without tracking the average gradient, it is not possible to use a fixed step size and converge to the optimal solution of the finite sum minimization problem in (2.7). Since the graph is directed, the push-sum protocol can be incorporated into previously mentioned algorithms employing gradient tracking over undirected graphs. Push-DIGing [25], DEXTRA [32], and SONATA [33] have been developed in this vein, which converges to the precise solution of (2.7) over directed graphs. To gain a better understanding, we examine the variable updates equations for the Push-DIGing algorithm:

$$
\begin{aligned}
\mathbf{u}_{k+1}^v &= \mathbf{u}_k^v - \sum_{u \in \mathcal{N}_{\text{in}}^v \cup \{v\}} q_{vu} \left(\mathbf{u}_k^u - \mu \mathbf{g}_k^u\right), \\
y_{k+1}^v &= y_k^v - \sum_{u \in \mathcal{N}_{\text{in}}^v \cup \{v\}} q_{vu} y_k^u, \\
\mathbf{x}_{k+1}^v &= \mathbf{u}_{k+1}^v / y^v, \\
\mathbf{g}_{k+1}^v &= \mathbf{g}_k^v - \sum_{u \in \mathcal{N}_{\text{in}}^v \cup \{v\}} q_{vu} \mathbf{g}_k^u + \nabla f^v(\mathbf{x}_{k+1}^v) - \nabla f^v(\mathbf{x}_k^v).
\end{aligned}
$$

The algorithm encompasses all the components mentioned earlier. It exclusively employs the $\mathbf{Q}$ gossip matrix with a zero column sum structure, making it suitable for directed graphs. It includes $\mathbf{g}^v$ variables that are updated using the DAC protocol to track the average of all nodes' gradients. Furthermore, it incorporates the y^v variable at each node, which updates in a manner similar to the $\mathbf{u}^v$ variables. Therefore, based on the push-sum protocol, the ratio between $\mathbf{u}^v$ and y^v, defined as $\mathbf{x}^v$ in this algorithm, converges to a consensus solution.

Push-Pull [34], [35]

This algorithm employs both zero row sum and zero column sum gossip matrices, denoted respectively by the $\mathbf{W}$ and $\mathbf{Q}$ matrices. This method leads to a more straightforward implementation and more relaxed assumptions for communication graphs. It has been demonstrated that for this algorithm, a critical node is necessary, for which there should be a directed path from this node to all other nodes and, conversely, from all other nodes to this critical node [34]. The algorithm updates the optimization variables by incorporating an additional

term, $\mathbf{g}^v$, which tracks the average gradient as follows:

$$\mathbf{x}_{k+1}^v = \mathbf{x}_k^v - \sum_{u \in \mathcal{N}_{\text{in}}^v \cup \{v\}} w_{vu} \mathbf{x}_k^u - \mu \mathbf{g}_k^v,$$
$$\mathbf{g}_{k+1}^v = \mathbf{g}_k^v - \sum_{u \in \mathcal{N}_{\text{in}}^v \cup \{v\}} q_{vu} \mathbf{g}_k^u + \nabla f^v(\mathbf{x}_{k+1}^v) - \nabla f^v(\mathbf{x}_k^v).$$

To the best of our knowledge, this algorithm represents the state-of-the-art in decentralized unconstrained optimization algorithms over directed graphs, achieving a linear rate of convergence for smooth and strongly-convex functions [34].

4.2.2 Decentralized Constrained Optimization methods

In this section, we will review the decentralized optimization methods addressing constraints and solving the problem presented in Equation (4.1). There are not many papers addressing this problem. However, they can be categorized not only based on whether they have considered undirected or directed communication graphs but also on whether they have taken distributed constraints into account or not. By distributed constraints, we refer to the situation in which each node has access to its own local private constraint set, which is not known to other agents.

As a note, our goal is to solve the constrained problem using first-order information, specifically by calculating the gradient and applying the projection operator. We prefer projection-based algorithms over dual-based or penalty-based methods because the latter requires computing gradient of Lagrangian or Augmented Lagrangian functions, which could be computationally expensive depending on the objective function and constraints. Additionally, the Lagrangian function might become non-smooth due to constraints, which means there is no guaranteed convergence rate when using first-order methods to solve the dual problem. This situation can occur, for instance, in the case of empirical risk minimization with ℓ_1-regularization.

Distributed Projected Sub-gradient algorithm [36]

This paper considers undirected and distributed algorithms. In the proposed algorithm, each node does one step of DGD, followed by projection onto its constraint set because of the possibility of going outside the *feasible reagion* after the first step. This algorithm can be written as

$$\mathbf{z}_{k+1}^v = \mathbf{x}_k^v - \sum_{u \in \mathcal{N}_{\text{in}}^v \cup \{v\}} w_{vu} \mathbf{x}_k^u - \mu \nabla f^v(\mathbf{x}_k^v),$$
$$\mathbf{x}_{k+1}^v = \mathrm{P}_{S^v}(\mathbf{z}_{k+1}^v),$$

where P_S indicates the projection operator. By following the same approach as the DGD algorithm in Section 4.2.1, we can write the fixed-point iteration of this algorithm and observe that

$$\mathbf{0} \in \partial I_{S^v}(\mathbf{x}^*) + \nabla f^v(\mathbf{x}^*). \qquad \forall v$$

Therefore, by using a fixed step size, the algorithm is not guaranteed to converge to the exact optimal solution because the objective functions and constraints differ among nodes. By using a fixed step size, the algorithm converges to a neighborhood of the optimal solution and the size of this neighborhood is equal to

$$\frac{1}{N}\sum_{v=1}^{N}\|\partial I_{S^v}(\mathbf{x}^*)+\nabla f^v(\mathbf{x}^*)\|_2^2.$$

Therefore, a decaying step size or a tracking approach is needed for convergence to the exact solution. To the best of our knowledge, there is not any paper that tracks the average of gradients and feasible directions.

As a final note for this algorithm, the convergence rate presented in that paper is established under restrictive assumptions regarding the communication graphs (fully connected graph) or constraints (identical constraints). Consequently, a general convergence rate for this algorithm has not been provided.

Directed-Distributed Projected Sub-gradient (DDPS) [37]

This paper has considered directed communication graphs but identical constraints among the nodes. The optimization variables are updated through the following equations in this algorithm

$$\mathbf{z}_{k+1}^v=\mathbf{x}_k^v-\sum_{u\in\mathcal{N}_{\text{in}}^v\cup\{v\}}w_{vu}\mathbf{x}_k^u+\epsilon\mathbf{y}_k^v-\mu_k\nabla f^v(\mathbf{x}_k^v), \tag{4.6}$$

$$\mathbf{x}_{k+1}^v=\mathrm{P}_S(\mathbf{z}_{k+1}^v), \tag{4.7}$$

$$\mathbf{y}_{k+1}^v=(1-\epsilon)\mathbf{y}_k^v-\sum_{u\in\mathcal{N}_{\text{in}}^v\cup\{v\}}q_{vu}\mathbf{y}_k^v+\sum_{u\in\mathcal{N}_{\text{in}}^v\cup\{v\}}w_{vu}\mathbf{x}_k^u, \tag{4.8}$$

where ϵ is the algorithm design parameter, which is generally a small positive value. As seen, this algorithm uses two row-stochastic and column-stochastic matrices, but it needs a diminishing step size.

This algorithm is the only one designed for directed graphs, addressing the setup considered in this thesis. However, it is tailored for identical constraints among the nodes. To enable a fair comparison, we attempted to modify this algorithm to accommodate distributed constraints. Accordingly, in the second step of this algorithm, we adapted the DDPS algorithm so that each node projects onto its individual constraint set in each iteration. It is important to note that the modified algorithm has not been examined in any published work, and thus, there are no guarantees regarding its convergence properties.

4.3 Summary

In this section, we provided an overview of decentralized optimization algorithms for both unconstrained and constrained decentralized optimization problems. We observed that there are not many papers addressing constraints, and among the few that do exist, none offer a working algorithm for distributed constraints,

even with undirected graphs. Therefore, in this thesis, we focused on distributed constraints and the more general setup of directed graphs. In the following chapter, we will briefly discuss the papers included in this thesis, along with an overview of our contributions.

Chapter 5

Summary of Included Papers

This thesis is written in the compilation format. In this chapter, we will provide an overview of the papers included. Each paper will be described individually, and we will provide a brief summary of the contributions and results. Accordingly, a clear understanding of the scope and significance of each paper and the relation between these papers is provided.

5.1 Paper I

In previous chapters, we discussed the importance of decentralized constrained optimization with distributed constraints. Furthermore, in Chapter 4, we identified gaps in the literature regarding constraints, specifically distributed constraints. In this thesis, we address this problem by proposing a novel decentralized optimization algorithm that takes into account general directed graphs and distributed constraints.

In Paper I, we introduce the Double Averaging and Gradient Projection (DAGP) algorithm. This algorithm employs two gossip matrices, one with a zero row-sum and another with a zero column sum structure, akin to the Push-Pull algorithm proposed for the unconstrained setup, making it suitable for directed graphs. DAGP implements a tracking approach, enabling the use of a fixed step size and fast convergence to the exact optimal solution. To the best of our knowledge, this is the first instance of addressing the problem of distributed constraints over a general directed communication network, along with basic assumptions on the objective function to provide a convergence rate. In this paper, we present the fundamentals of a novel convergence analysis methodology called "aggregate lower bounding," which is one of the main contributions of this thesis, further detailed in Paper II.

We conducted experiments on constrained decentralized optimization problems involving directed graphs, where DAGP demonstrates superior performance compared to the modified version of the existing algorithm DDPS. Ad-

ditionally, we carried out experiments on unconstrained problems, in which DAGP exhibits comparable performance to state-of-the-art decentralized optimization algorithms, namely Push-Pull and ADDOPT. This highlights the versatility and effectiveness of DAGP in handling various types of optimization problems in a decentralized setting.

5.2 Paper II

This paper builds upon Paper I, addressing the same setup and problem, but delving deeper into the ideas and intuitions behind the DAGP algorithm. For instance, it explains how the average of gradients and feasible directions are tracked, and how the optimality condition is satisfied using the innovative "distributed null projection" concept. Furthermore, the paper explores the importance of the constrained setup more extensively in the literature. We demonstrate that any unconstrained optimization, even with non-smooth objective functions, can be reformulated into an equivalent problem with a linear objective function and epigraphs as constraints. We also introduce a fast algorithm based on proximal backtracking for calculating projection onto the epigraph of a function, thereby enhancing the efficacy of DAGP in solving decentralized constrained optimization problems with non-smooth objectives.

In this paper, which contains the second main contribution of this thesis, we introduce a novel, general methodology for the convergence analysis of optimization methods. This methodology addresses the well-known challenges associated with Lyapunov functions mentioned in Section 2.5. Utilizing this framework, we demonstrate that the feasibility gap of the solution diminishes at a rate of $\mathcal{O}(1/K)$, and the optimality gap decreases at a rate of $\mathcal{O}(1/\sqrt{K})$, where K represents the total number of iterations. Our framework allows us to bypass restrictive assumptions, such as identical local constraints, a decaying step size, and strong-convexity. To further highlight the efficacy of this novel methodology, we provide an alternative convergence analysis of (non-distributed) gradient descent within this new context.

Finally, we expand our experimental results to include the practical optimal transport for the domain adaptation problem [38]. First, we reformulate the optimal transport problem within the context of decentralized constrained optimization with distributed constraints. Then, we provide numerical evidence of how the exact solutions for optimal transport can be computed efficiently, even for large problems, which is the case for the domain adaptation problem.

Chapter 6

Discussion and Future Work

In recent years, federated learning has attracted significant attention. There is a substantial overlap between federated learning and distributed optimization studied in this thesis. Consequently, numerous challenges already discussed in the literature of decentralized optimization or federated learning can be regarded as potential future directions of this thesis. To elaborate, the following are some possible directions:

- **Asynchronous setup:** There are some papers, such as [39], that have addressed communication delays and asynchronous iterations. We can explore this setup and adapt the DAGP algorithm and our convergence analysis methodology to accommodate asynchrony.

- **Decentralized stochastic optimization methods:** There are several papers, such as [40]–[42], that have considered local objective functions with finite-sum structure and proposed that in each iteration of the decentralized algorithm, each node utilizes stochastic gradients for updates. We can explore this setup, adapt the DAGP algorithm, and employ our aggregate lower bounding methodology. By doing so, we may introduce more efficient algorithms for large-scale machine learning problems.

- **communication efficiency:** Communication among agents can be costly. To reduce the number of transmitted bits, message compression can be employed, and the performance of decentralized algorithms can be analyzed in conjunction with the error introduced by message compression. It will be worthwhile to investigate the DAGP algorithm from this point of view.

- **communication network:** Another unexplored research direction, to the best of our knowledge, involves the design of communication networks. The central question to address is: What is the optimal fixed or dynamic communication network to achieve the fastest convergence rate? The

importance of answering this question lies in the fact that by optimizing the communication network, for instance, by reducing the number of edges while maintaining reasonable connectivity, we can achieve an $\epsilon-$optimal solution with fewer communication rounds among the nodes. This would result in a significant improvement concerning communication costs. Another reason for studying communication networks is that in certain environments, communication over specific links might be significantly more expensive. Designing a dynamic communication graph takes these costs into account and minimizes communication over these costly links. As a result, more informative data can be transmitted within the network.

In conclusion, it is worth noting that decentralized optimization, particularly with constraints, will gain increasing attention in the coming years due to the growing number of large datasets and machine learning models, as well as the demand for greater computational resources. As demonstrated, there are numerous challenges to be addressed in the future, and this thesis will contribute to addressing them.

Bibliography

[1] A. Beck, *First-order methods in optimization*. SIAM, 2017 (cit. on pp. 5, 9).

[2] S. Boyd, S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004 (cit. on pp. 5, 13).

[3] S. Bubeck *et al.*, "Convex optimization: Algorithms and complexity," *Foundations and Trends® in Machine Learning*, vol. 8, no. 3-4, pp. 231–357, 2015 (cit. on pp. 5, 8).

[4] Y. Nesterov *et al.*, *Lectures on convex optimization*. Springer, 2018, vol. 137 (cit. on pp. 5, 9, 10).

[5] J. Nocedal and S. J. Wright, *Numerical optimization*. Springer, 1999 (cit. on pp. 6, 13).

[6] M. Schmidt, N. Le Roux and F. Bach, "Minimizing finite sums with the stochastic average gradient," *Mathematical Programming*, vol. 162, pp. 83–112, 2017 (cit. on p. 10).

[7] A. Defazio, F. Bach and S. Lacoste-Julien, "Saga: A fast incremental gradient method with support for non-strongly convex composite objectives," *Advances in neural information processing systems*, vol. 27, 2014 (cit. on p. 10).

[8] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," *Advances in neural information processing systems*, vol. 26, 2013 (cit. on p. 10).

[9] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *Ussr computational mathematics and mathematical physics*, vol. 4, no. 5, pp. 1–17, 1964 (cit. on p. 10).

[10] A. Chambolle and C. Dossal, "On the convergence of the iterates of the "fast iterative shrinkage/thresholding algorithm"," *Journal of Optimization theory and Applications*, vol. 166, pp. 968–982, 2015 (cit. on p. 10).

[11] Y. Nesterov and S. U. Stich, "Efficiency of the accelerated coordinate descent method on structured optimization problems," *SIAM Journal on Optimization*, vol. 27, no. 1, pp. 110–123, 2017 (cit. on p. 10).

[12] Z. Allen-Zhu, Z. Qu, P. Richtárik and Y. Yuan, "Even faster accelerated coordinate descent using non-uniform sampling," in *International Conference on Machine Learning*, PMLR, 2016, pp. 1110–1119 (cit. on p. 10).

[13] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009 (cit. on pp. 12, 23).

[14] S. Boyd, A. Ghosh, B. Prabhakar and D. Shah, "Randomized gossip algorithms," *IEEE transactions on information theory*, vol. 52, no. 6, pp. 2508–2530, 2006 (cit. on p. 12).

[15] A. Chambolle and T. Pock, "A first-order primal-dual algorithm for convex problems with applications to imaging," *Journal of mathematical imaging and vision*, vol. 40, pp. 120–145, 2011 (cit. on p. 13).

[16] I. Goodfellow, Y. Bengio and A. Courville, *Deep learning*. MIT press, 2016 (cit. on pp. 17, 18).

[17] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006 (cit. on p. 17).

[18] L. Bottou, F. E. Curtis and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM review*, vol. 60, no. 2, pp. 223–311, 2018 (cit. on p. 18).

[19] P. Kairouz, H. B. McMahan, B. Avent *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021 (cit. on p. 21).

[20] S. Lee and A. Nedic, "Distributed random projection algorithm for convex optimization," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 2, pp. 221–229, 2013 (cit. on p. 21).

[21] P. A. Forero, A. Cano and G. B. Giannakis, "Consensus-based distributed support vector machines.," *Journal of Machine Learning Research*, vol. 11, no. 5, 2010 (cit. on p. 21).

[22] K. Yuan, Q. Ling and W. Yin, "On the convergence of decentralized gradient descent," *SIAM Journal on Optimization*, vol. 26, no. 3, pp. 1835–1854, 2016 (cit. on p. 23).

[23] W. Shi, Q. Ling, G. Wu and W. Yin, "Extra: An exact first-order algorithm for decentralized consensus optimization," *SIAM Journal on Optimization*, vol. 25, no. 2, pp. 944–966, 2015 (cit. on pp. 23, 24).

[24] M. Zhu and S. Martínez, "Discrete-time dynamic average consensus," *Automatica*, vol. 46, no. 2, pp. 322–329, 2010 (cit. on p. 24).

[25] A. Nedic, A. Olshevsky and W. Shi, "Achieving geometric convergence for distributed optimization over time-varying graphs," *SIAM Journal on Optimization*, vol. 27, no. 4, pp. 2597–2633, 2017 (cit. on pp. 24, 26).

[26] P. D. Lorenzo and G. Scutari, "Next: In-network nonconvex optimization," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, pp. 120–136, 2016 (cit. on p. 24).

[27] R. Xin, S. Pu, A. Nedić and U. A. Khan, "A general framework for decentralized optimization with first-order methods," *arXiv*, vol. 108, no. 11, 2020, ISSN: 23318422 (cit. on pp. 24, 25).

[28] K. I. Tsianos, S. Lawlor and M. G. Rabbat, "Push-sum distributed dual averaging for convex optimization," in *2012 ieee 51st ieee conference on decision and control (cdc)*, IEEE, 2012, pp. 5453–5458 (cit. on p. 25).

[29] A. Nedić and A. Olshevsky, "Distributed optimization over time-varying directed graphs," *IEEE Transactions on Automatic Control*, vol. 60, no. 3, pp. 601–615, 2014 (cit. on p. 25).

[30] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge university press, 2012 (cit. on p. 25).

[31] D. Kempe, A. Dobra and J. Gehrke, "Gossip-based computation of aggregate information," in *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, IEEE, 2003, pp. 482–491 (cit. on p. 25).

[32] C. Xi and U. A. Khan, "Dextra: A fast algorithm for optimization over directed graphs," *IEEE Transactions on Automatic Control*, vol. 62, no. 10, pp. 4980–4993, 2017. DOI: 10.1109/TAC.2017.2672698 (cit. on p. 26).

[33] G. Scutari and Y. Sun, "Distributed nonconvex constrained optimization over time-varying digraphs," *Mathematical Programming*, vol. 176, no. 1, pp. 497–544, 2019 (cit. on p. 26).

[34] S. Pu, W. Shi, J. Xu and A. Nedic, "Push-Pull Gradient Methods for Distributed Optimization in Networks," *IEEE Transactions on Automatic Control*, vol. 66, no. 1, pp. 1–16, 2021, ISSN: 15582523. DOI: 10.1109/TAC.2020.2972824. arXiv: 1810.06653 (cit. on pp. 26, 27).

[35] R. Xin and U. A. Khan, "A Linear Algorithm for Optimization over Directed Graphs with Geometric Convergence," *IEEE Control Systems Letters*, vol. 2, no. 3, pp. 313–318, 2018, ISSN: 24751456. DOI: 10.1109/LCSYS.2018.2834316. arXiv: 1803.02503 (cit. on p. 26).

[36] A. Nedic, A. Ozdaglar and P. A. Parrilo, "Constrained consensus and optimization in multi-agent networks," *IEEE Transactions on Automatic Control*, vol. 55, no. 4, pp. 922–938, 2010 (cit. on p. 27).

[37] C. Xi and U. A. Khan, "Distributed subgradient projection algorithm over directed graphs," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3986–3992, 2016 (cit. on p. 28).

[38] N. Courty, R. Flamary, D. Tuia and A. Rakotomamonjy, "Optimal Transport for Domain Adaptation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 9, pp. 1853–1865, 2017, ISSN: 01628828. DOI: 10.1109/TPAMI.2016.2615921. arXiv: 1507.00504 (cit. on p. 32).

[39] D. Wang, Z. Wang, M. Chen and W. Wang, "Distributed optimization for multi-agent systems with constraints set and communication time-delay over a directed graph," *Information Sciences*, vol. 438, pp. 1–14, 2018 (cit. on p. 33).

[40] A. Mokhtari and A. Ribeiro, "DSA: Decentralized double stochastic averaging gradient algorithm," *Journal of Machine Learning Research*, vol. 17, pp. 1–35, 2016, ISSN: 15337928 (cit. on p. 33).

[41] H. Hendrikx, F. Bach and L. Massoulié, "Dual-free stochastic decentralized optimization with variance reduction," *Advances in Neural Information Processing Systems*, vol. 33, pp. 19 455–19 466, 2020 (cit. on p. 33).

[42] R. Xin, S. Kar and U. A. Khan, "Decentralized stochastic optimization and machine learning: A unified variance-reduction framework for robust performance and fast convergence," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 102–113, 2020 (cit. on p. 33).