Distributed Optimization in Time-Varying Environments

A thesis of two parts

MARIE MAROS

Doctoral Thesis
Stockholm, Sweden 2019
Akademisk avhandling som med tillstånd av Kungl Tekniska högskolan framlägges till offentlig granskning för avläggande av teknologie doktorsexamen i electro- och systemteknik Monday den 19 agusti 2019 klockan 10.00 i F3, Lindstedtsvägen 26, Stockholm.

© 2019 Marie Maros, unless otherwise noted.

Tryck: Universitetsservice US AB
Abstract

Solving optimization problems in a distributed manner is critical in many systems. Many relevant systems are distributed in nature in the sense that they consist of autonomous agents that are to come to a joint decision based on a certain metric. In many cases, these agents may collect information independently and would therefore have to centralize all the data. In applications were this is not a viable approach distributed solutions are desirable. In this thesis, we study distributed optimization methods in time-varying environments.

In the first part of the thesis, we consider optimization problems that evolve over time in a controlled manner. We propose the use of the Alternating Direction Method of Multipliers (ADMM) due to its flexibility in step-size selection. We establish ADMM’s ability to follow an optimal point as it moves over time. In our set-up, a distributed variant of ADMM is allowed to perform a single iteration per problem change. Under smoothness assumptions on the objective and constraint functions we establish that there exists a sufficiently small variation of the problem data for which we can guarantee that ADMM will be able to follow the optimal point in a decentralized manner. These conditions are less stringent than the conditions found in the literature. Later on, we introduce a stochastic model for the variation of the problem’s data. Under some assumptions, we establish that decentralized ADMM is capable of remaining in a bounded mean square neighbourhood of a primal-dual optimal point. Introducing a stochastic model allows to us relax many of the requirements found in the literature, while still providing some guarantees. We provide with application examples and simulations for both scenarios.

In the second part of the thesis we consider distributed optimization methods that converge over time-varying networks. We propose the first dual method to converge linearly on time-varying networks, in which we allow the networks to become disconnected. We establish that the method converges R-linearly and illustrate that under some circumstances it performs better than other state of the art methods, while, at the same time, cutting the required information exchanges in half. Since the proposed method is computationally quite expensive we propose a linearized and therefore computationally cheaper version of our method. Finally, we establish that the linearized version will also converge R-linearly on time-varying graphs and quantify the loss in convergence rate due to the approximation.
Att lösa optimeringsproblem på ett distribuerat tillvägagångssätt är kritiskt i många system. Många relevanta system är distribuerade till sin karaktär i den mening att de består av autonoma agenter som skall nå ett gemensamt beslut baserat på en viss metrik. I många fall kan dessa agenter samla in information självständigt och skulle därför behöva centralisera all data. För tillämpningar där detta inte är möjligt är distribuerade lösningar önskvärda. I denna avhandling studerar vi distribuerade optimeringsmetoder i tidsvariabla omgivningar.


I avhandlingens andra del studerar vi distribuerade optimeringsmetoder som konvergerar över tidsvariabla nätverk. Vi föreslår den första dual metoden som konvergerar linjärt över tidsvariabla nätverk där vi tillåter nätverken att bli frikopplade. Vi visar att metoden konvergerar R-linjärt och belyser att under vissa omständigheter fungerar den bättre än andra tidigare kända metoder medan den samtidigt halverar de erforderliga informationsutbyte. Eftersom den föreslagna metoden är ganska beräkningsintensiv föreslår vi en linjäriserad, och därför beräkningsmässigt enklare, variant av vår metod. Slutligen fastställer vi att den linjäriserade varianten också konvergerar R-linjärt på tidsvariabla grafer och kvantifierar förhusten i konvergenschastighet på grund av approximationen.
List of Papers

The thesis is based on the following papers:


Other works not included in this thesis:


Acknowledgements

First and foremost, I would like to thank my advisor, Prof. Joakim Jaldén for giving me the opportunity to embark in this journey. Thank you for your keen technical eye, your understanding, and patience, but especially, thank you for trusting me and my abilities.

Teaching would not have been half as fun without Prof. Tobias Oechtering. Thank you for all your time and advice, I am sure I am a better teacher because of it.

I am indebted to Prof. Angelia Nedić for hosting me during my stay in Arizona and for making time in her busy schedule to have discussions with me. I would like to extend my thanks to Prof. Anna Scaglione for her kindness during my stay in Arizona. Thanks to Wicak Ananduta, Giuseppe Belgioso, Ignacio Losada Carreno, Nurullah Karakoç, Rakshe Ramakrishna, Nikhil Ravi, Dr. Teklemariam Tesfay and Parth Thaker for making my stay in Arizona all the more fun. Special thanks to Dr. Eran Schweitzer for our morning coffee discussions and for hosting a wonderful thanksgiving dinner. I would like to particularly thank Kári Heinisson for our memorable (and sometimes scary) hikes and for his endless adventurous spirit.

I am also indebted to Dr. César Uribe Menses for inviting and hosting me during my short stay in Boston and giving me the opportunity to meet many inspiring researchers, himself included.

I am deeply grateful to Prof. Alex Olshevsky for acting as an opponent to the thesis, and to the members of the evaluation committee, Prof. Geert Leus, Prof. François Glineur, and Prof. Kostas Margellos. It is an honour to have my work assessed by such an accomplished group of people.

Special thanks to Dr. Klas Magnusson, Dr. Nima Moghadam, Dr. Ehsan Olfat, Dr. Arash Owrang, Dr. Martin Sundin, Dr. Alla Tarighati and Dr. Arun Venkitaraman for many laughs together during our “fikas”. I would also like to thank Dr. Ahti Aimonäe with whom I have enjoyed sharing course work with. I am indebted to Dr. Arun Venkitaraman for his friendship and sense of humour without which my journey would have been significantly blander. My deepest gratitude goes to Baptiste Cavarec for his friendship, for cutting through mountains, and for introducing me to climbing. Thanks to Dr. Germán Bassi for your friendship, your Argentine passion, and for your attention to detail while proof reading this thesis. Special thanks also to Dr. Photis Stavrou for meaningful discussions over coffee. I would also like to thank Dr. Zuxing Li with whom sharing teaching duties was always a great experience. Thanks to my former and present colleagues in the division with whom I have had the pleasure to share dinners, parties, escape room activities, skiing trips, etc.

While this thesis carries my name as its sole author, it would not have been possible without the support of my partner and eternal cheerleader Jonatan Asketorp. Thank you for all your support and understanding. I would like to thank you, Göran Asketorp, and Henrik Forssel for the Swedish translation of the abstract. Finally, my deepest gratitude goes to my parents, Ladawan
and Tamas Maros, for their unconditional support and for being a constant I can always rely on.

Marie Maros
Stockholm, August 2019
Contents

Abstract v
Sammanfattning vii
List of Papers ix
Acknowledgements xii
Contents xiii

I Introduction and Thesis Overview 1

1 Introduction 3
  1.1 A brief history of distributed optimization . . . . . . . . . . . . 5
    1.1.1 Time-Varying Problems . . . . . . . . . . . . . . . . . . 8
    1.1.2 Time-Varying Networks . . . . . . . . . . . . . . . . . . 10

2 Contributions 17
  2.1 Time-Varying Problems . . . . . . . . . . . . . . . . . . . . . . . 17
    2.1.1 ADMM for Distributed Dynamic Beamforming . . . . . . 17
    2.1.2 Dynamic Power Allocation for Smart Grids via ADMM . 21
    2.1.3 On Decentralized Tracking with ADMM for Problems with
         Time-Varying Curvature . . . . . . . . . . . . . . . . . . . . 22
  2.2 Time-Varying Networks . . . . . . . . . . . . . . . . . . . . . . . 26
    2.2.1 PANDA . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 26
    2.2.2 Eco-PANDA . . . . . . . . . . . . . . . . . . . . . . . . . . . 29

II Included Papers 31

A ADMM for Distributed Dynamic Beamforming 33
  A.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35
### Contents

E.3.1 Proof structure ........................................ 126
E.3.2 Establishing the circle of arrows ................. 128
E.3.3 Applying the small gain theorem ............... 137
E.4 Numerical Experiments ............................... 140
E.5 Conclusion ............................................. 141

F Eco-PANDA: A Computationally Economic, Geometrically Converging, Dual Optimization Method on Time-Varying Undirected Graphs 143
F.1 Introduction ............................................. 145
F.2 Problem Formulation and Algorithm .............. 146
F.3 Proof of convergence .................................. 149
F.4 Numerical Experiments .............................. 155
F.5 Conclusions ............................................ 156

Bibliography 157
Part I

Introduction and Thesis Overview
Introduction

Many problems in engineering [1] can be cast as optimization problems of the form

\[
\begin{align*}
\min_{\mathbf{x} \in \mathbb{R}^p} & \quad f_0(p_0, \mathbf{x}) \\
\text{subject to} & \quad f_i(p_i, \mathbf{x}) \leq 0, \ i = 1, \ldots, N_c,
\end{align*}
\] (1.0.1a)

where \( f_i, i = 0, \ldots, N_c, \) are convex functions in \( \mathbf{x} \), and \( p_i \in \mathbb{R}^{M_i}, i = 0, \ldots, N_c \) denote the problem parameters. Typically, the problem parameters \( p_i \) represent collected data or measurements. Classical optimization methods [2] have been successfully applied on problems stemming from wireless communications, signal processing [3], economics [4], and automatic control [5], among others. However, in current systems many of the applications found in the mentioned works [3–5] are distributed in nature. This implies that different subsets of the parameters \( p_0, \ldots, p_{N_c} \) are in possession of different agents or nodes connected via a network. For the sake of clarity, when we use the word network we refer to the set of links through which the agents are connected. On the other hand, when we use the word system we refer to the whole entity which consists of both the agents and the network. A naive solution has all agents transmit their data to a single agent. The receiving agent would then find the optimal solution to (1.0.1) and distribute the solution to the remaining agents. The disadvantages of this approach are twofold. First, it introduces a single point of failure in the system. If the agent that receives all the information is not operating correctly, or is malicious, the entire system is compromised. Second, depending on the problem characteristics and the network structure, the amount of communication required can be very large. This is especially critical in applications that require a quick response of the system. Examples in which this is the case will be presented later on in this thesis. Thus, an alternative solution is a distributed solution, i.e., an iterative procedure such that at each iteration each agent exchanges some information with her immediate neighbours.

Problems for which distributed solutions are desirable are common, for example, in estimation [6], wireless networks [7], and smart grids [8]. Most of the classical methods used to solve the problem (1.0.1) in a centralized manner have been ex-
Introduction

tended to work in a distributed set-up given that the original problem has a beneficial structure. In the centralized set-up, some methods are more desirable than others depending on the structure of the objective functions and constraints [9], [10], in the sense that they may provide a quicker convergence to an optimal solution, robustness to parameter choices, etc. Solving problems in a distributed manner inherits all the intricacies of solving them in a centralized manner but we now also need to consider how the information is spread, i.e., what information is known to which agent. In most cases, the information each individual agent has acquired is relevant to the entire system. This implicitly defines a notion of coupling between the different agents in the system. Typically, approaches to solve (1.0.1) in a distributed manner will attempt to represent all the coupling using a single constraint. In this way, the complicating factor introduced by the distributed nature of the problem is represented by a single entity within the optimization problem.

To make the exposition clearer let us introduce some notation. Let $n$ denote the number of agents in the network. Further let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ be the graph representing the system where the vertices $\mathcal{V} = \{1, \ldots, n\}$ denote the set of agents and $\mathcal{E}$ denote the set of edges representing the network links. We will assume for now that the graph is undirected and connected. This implies that, if agent $i$ can communicate with agent $j$, agent $j$ can also communicate with agent $i$, and that by using multiple hops all agents are capable of communicating with each other.

A sub-set of problems of the family of (1.0.1) that is common and amendable to distributed solutions can be described as

$$
\min_{\{x_i \in \mathbb{R}^p\}_{i=1}^n} \quad \sum_{i=1}^n f_i(x_i) \quad \text{(1.0.2a)}
$$

subject to

$$
g_i(x_i) \leq 0, \quad i = 1, \ldots, n \quad \text{(1.0.2b)}
$$

$$
h(x_1, \ldots, x_N) \leq 0, \quad \text{(1.0.2c)}
$$

where each agent $i$ has exclusive knowledge of their own convex objective $f_i$ and convex constraint function $g_i$. Note that both $f_i$ and $g_i$ decouple throughout the agents, since the only depend on their local variables $x_i$. Note that in (1.0.2), as opposed to in (1.0.1), the problem data is not made explicit.

The coupling in (1.0.2) is represented by the convex function $h$ which couples or relates all the variables $(x_1, \ldots, x_n)$. All the work included in this thesis concerns problems of the form (1.0.2). Note that this includes resource allocation type of problems, where the goal is to maximize the sum-utility of the system while using a shared resource represented by (1.0.2c). If the constraint (1.0.2c) is affine, cast with equality, and is further endowed with the structure of the graph $\mathcal{G}$, we can enforce the agents to agree on their individual values of $x_i$. For this, let us define a mixing matrix $W$. The mixing matrix has a network dependent structure which represents the coupling in the optimization problem.

Let $\tilde{x} = [x_1^T, \ldots, x_n^T]^T$. Under the assumption that the graph $\mathcal{G}$ is connected the
mixing matrix $W$ is such that the problem

$$\min_{x \in \mathbb{R}^{np}} \sum_{i=1}^{n} f_i(x_i) + I_{g_i \leq 0}(x_i)$$

subject to

$$W\bar{x} = 0,$$

where $I_{g_i \leq 0}$ is the indicator function of the set $\{x_i : g_i(x_i) \leq 0\}$, is equivalent to its centralized counterpart

$$\min_{x \in \mathbb{R}^p} \sum_{i=1}^{n} f_i(x) + I_{g_i \leq 0}(x).$$

Note that in (1.0.3) we have a copy, $x_i$, of the optimization variable in (1.0.4), $x$, for each agent $i$. The copies $x_i$ are all stored in $\bar{x} \in \mathbb{R}^{np}$. Introducing the copies allows each agent to have her own variable to update when the methods are designed. Many methods have been proposed in the literature to solve optimization problems of the form (1.0.3) under different assumptions on the smoothness and convexity properties of the objectives $f_i$ and on the presence or absence of the constraints $g_i$.

We review some of the works in the next section.

1.1 A brief history of distributed optimization

A history on distributed optimization must start with its distinction from parallel computing. The roots of parallel computing go back to the 1950’s; in the parallel computing set-up, a computational task is to be split and assigned to $n$ different agents under the coordination and control of a coordinating or master entity. Further, the architecture is typically pre-set or designed with specific computational tasks in mind. Distributed computing is different from parallel computing in some aspects. The computational task is also split and assigned to $n$ different agents but the master or coordinating agent is preferably, though not always, removed. Moreover, the network structure is typically enforced by external factors rather than designed for a specific set of computational tasks. Due to the exogenous nature of the network, the links that define it may induce delays or even not be present at all times. Within distributed computing, we focus on obtaining distributed solutions to optimization problems, i.e., the computational task at hand is to solve an optimization problem.

Distributed optimization has been an active research topic since the early 1980s, with much of the current literature building on the seminal work [11] (see Chapter 5 in particular). In [11] the goal was to distribute the computational effort of minimizing a smooth function $f$ among $n$ agents. The work in [11] set up the framework under which much of the modern literature is based upon. First and most generally, agents have finite memory and are not required to exchange information with all other agents at all iterations, allowing for communication in localized neighbourhoods and asynchronous exchanges. Second, and mostly relating to first order
Introduction

methods\(^1\), the iterates of the methods proposed in [11] can be interpreted as perturbed consensus steps in which the perturbation stirs the variables in a direction that corresponds to a descent direction in the objective function.

An independent line of research stemmed from incremental optimization methods [13]. In particular, the work in [6] focuses on optimization problems of the form

\[
\min_{x \in \mathbb{R}^p} \sum_{i=1}^{n} f_i(x)
\]

(1.1.1)

by observing that many estimation criteria possess that particular structure. The problem is solved in a distributed manner by updating the primal variable, \(x\), sequentially such that each node \(i\) awaits the reception of a message containing the latest update in \(x\) before performing any computations based on their objective \(f_i\).

Based on the use of dual methods, distributed optimization for decentralized resource allocation started gaining traction in the early 2000s. Depending on the structure of the resource allocation problem at hand, a master agent would be required or not. Notable works in this direction include [15] and [16], which laid down the foundation to parallel optimization via dual decomposition [7]. In particular, the work in [15] established linear convergence and the ideal network weight selection (via SDP) of distributed gradient descent when used upon resource allocation problems with a strongly convex objective with Lipschitz continuous gradients.

Building on [11] the acclaimed work in [14] formalized the notion of system and problem models that would become the standard in the current literature. The setup is similar to that in [11] in the sense that perfect communication amongst agents is not assumed. In particular, the system is modelled via a time-varying graph in which edges may be present or not at different time instances; within the graph each node is allowed to communicate solely with its immediate neighbours. The problem model dealt with in [14] is the same as in [6], i.e. (1.1.1). The method proposed is in the lines of that in [11] where a consensus-type iterate is perturbed by a local (sub)-gradient. This method is currently referred to as decentralized (sub)-gradient descent. Convergence and rate guarantees are also provided in [14]. Many works stemmed from the work in in [14] with improved convergence rates under more stringent assumptions on the objective functions, quantized and or asynchronous iterates, etc.

The work in [14] was extended to deal with constraints in [26] and even shown to converge on random networks [27]. Under the assumption that the objective functions in (1.1.1) have Lipschitz continuous gradients, the work in [17] establishes that the method proposed in [14] converges to a neighbourhood of the optimal point at rate \(O(\frac{1}{k})\), where \(k\) is the iteration number. If the objective functions are also strongly convex, then [17] establishes that the rate is linear \(O(\lambda^k), \lambda \in (0,1)\) to a neighbourhood of the optimal point. Hence concluding that “vanilla” decentralized gradient descent required a diminishing step-size to converge to the

\(^1\)First order methods do not require the computation or approximation of second order derivatives.
optimal point even when applied to the most forgiving class of problems. If the step-size were set to remain static the method would simply oscillate in a neighbourhood of the optimal point. This is a negative result as it indicates a need for additional coordination to select a sequence of diminishing step-sizes, while, at the same time, providing significantly slower rates than those corresponding to the centralized gradient descent for the same problem family [52]. Meanwhile, primal and primal-dual methods were being adapted to solve problems of the form of (1.0.3) in a distributed set-up. In particular, the work in [18] introduces a decentralized version of dual ascent (or dual sub-gradient) [123].

A widely used method in distributed optimization is the Alternating Direction Method of Multipliers (ADMM). By introducing auxiliary variables that live on the edges of the graph, the method provides a distributed solution and inherits most of the convergence properties of its centralized counterpart. In particular, ADMM has been shown to converge at rate $O(1/k)$ for convex constrained problems [19]. A decentralized and asynchronous variant of ADMM was presented for the first time in [20], establishing a convergence rate of $O(1/k)$ given that the objective functions are convex. Meanwhile, in [21] the authors established that decentralized ADMM converges linearly to the optimal point $x^*$ of (1.1.1) when applied to (1.1.1) given that the objective functions are strongly convex and have Lipschitz continuous gradients. Other works have further extended the convergence properties of ADMM as well as proposed linearized [22], asynchronous [23], censored [24]; and other variants [25]. Other type of primal-dual methods have been proposed for the distributed solution of optimization problems of the form (1.0.3). While ADMM and dual ascent are the most influential primal-dual and dual method, respectively, to the work in the thesis, other works worth mentioning are decentralized dual averaging [28] and the work in [29]. In [29] a primal dual method is proposed to solve constrained optimization problems of the form of (1.0.2).

Bridging the gap between primal methods and primal-dual methods while, at the same time, providing a technique to tweak decentralized gradient descent (so as to converge linearly with a constant step-size) is EXTRA [30]. Let $x^*_i$ denote a copy of the optimal point $x^*$ at agent $i$. The main issue with decentralized gradient descent when using a constant step-size is that the gradient of each privately owned objective function need not necessarily be zero, i.e., while the quantity $\sum_{i=1}^n \nabla f_i(x^*_i) = 0$ each individual $\nabla f_i(x^*_i)$ need not be equal to zero. This causes decentralized gradient descent to oscillate around the minimizer instead of converging to it if the step size is constant. The remedy proposed in [30], EXTRA, is to have the gradient at two consecutive iterates counter each other so that, at the limit, the perturbation incurred to the primal variables is actually zero. EXTRA requires two communication exchanges per iterate in which different mixing matrices may be used. A primal-dual interpretation of EXTRA was also provided in [31]. DIGing, the method proposed in [31], is a particular case of EXTRA where the two different mixing matrices are set to be the same. DIGing is shown to converge R-linearly on time-varying networks serving as the main inspiration for our work on time-varying networks. Further linking EXTRA to primal-dual methods, the work in [32] estab-
lishes that EXTRA can be shown to be equivalent to linearized generalized ADMM for particular choices of mixing matrices.

Recently, for unconstrained problems with strongly convex objective functions with Lipschitz continuous gradients, optimal convergence rates and algorithms that achieve them in distributed set-ups have been established and proposed in [33,34]. These are, in essence, dual ascent-type methods using Nesterov’s acceleration. The works in [33,34] shed some light in the loss of rate incurred due to the usage of a distributed method as opposed to its centralized counterpart. The optimality of dual-ascent in distributed set-ups also motivates part of our work on time-varying networks.

1.1.1 Time-Varying Problems

A model for time-varying optimization problems is given by

\[
\begin{align*}
&\min_{x \in \mathbb{R}^p} f_0(p_0^t, x) \\
&\text{subject to } f_i(p_i^t, x) \leq 0, \quad i = 1, \ldots, N_c,
\end{align*}
\]  

(1.1.2)

where the problem parameters \( p_i^t \) are allowed to vary with the time index \( t \). Depending on how quickly the problem evolves as a function of \( t \), the number of iterations required by an iterative scheme to solve (1.1.2) to a reasonable level of accuracy may be prohibitive. Further, in a distributed set-up, the convergence of the method is hindered not only by the machine’s ability to compute iterates before the problem has changed significantly but also by the network’s structure, as this will affect how quickly the agents can obtain information from the other side of the network. Problems of the type (1.1.2) in contexts where the problem changes quickly enough so that an iterative method does not have time to converge to a desirable level of accuracy include applications in control [38], robotics [39], smart-grids [40] and signal processing [41] among others [42], [43].

The first work to deal with time-varying optimization problems in a centralized manner, albeit without constraints, can be found in [35]. In [35] the author selects a problem family for which, in the static case, gradient descent converges linearly. Then, it is established performing a gradient step by problem change, given that the movement of the optimal point from one time instant to the next (primal drift) is uniformly bounded, one will always remain in a neighbourhood proportional to the upper bound on the primal drift. These type of statements are referred to as tracking statements in the literature. Tracking statements of the type of [35] have been extended to handle constraints [36] and even to stochastic variants [37].

These statements are fundamentally different from those performed within regret minimization contexts. Regret minimization deals with sequences of optimization problems described by their objective functions \( \{f_t\}_{t \geq 1} \). However, the goal is to choose, at each time \( t \), a solution that performs close to the best overall decision
1.1. A brief history of distributed optimization

in hindsight. More specifically, the goal is to have the quantity

\[ R(T) \triangleq \left\{ \sum_{t=1}^{T} f_t(x_t) - \min_{x \in \mathbb{R}^n} \sum_{t=1}^{T} f_t(x) \right\} \]  

(1.1.3)

grow sub-linearly with time. Regret minimization is commonly referred to as on-line optimization. Many different scenarios are considered, including the case in which the choice of functions is adversarial in nature [50]. Note that the difference lies in the fact that, when making a tracking statement, we are interested in remaining close to the \textit{current} minimizer, while in the on-line case, we want to perform close to the overall minimizer in hindsight. In the regret minimization case, under mild assumptions, this implies that the objective we are trying to minimize at each \( T \), \( \sum_{t=1}^{T} f_t(x_t) \), becomes closer to a static function as time goes by. A closely related quantity to that in (1.1.3) is the dynamic regret [51]

\[ R_d(T) \triangleq \left\{ \sum_{t=1}^{T} f_t(x_t) - \sum_{t=1}^{T} \min_{x_t \in \mathbb{R}^n} f_t(x_t) \right\} \]  

(1.1.4)

Note that the main difference with (1.1.3) is that, in (1.1.4), we are comparing to the aggregate cost of always making the optimal decision. In this case the difference between a tracking statement and minimizing the dynamic regret is more subtle. This approach shares with static regret minimization that the objective function is accumulative. However, the dynamic regret is minimized if the minimizer of each function \( f_t \) can be picked at each moment in time. As the goal is to minimize the quantity (1.1.4) a different effort can be exerted at different time instances \( t \) depending on the gain we predict. Minimizing the dynamic regret (1.1.4) relates closely to using prediction-correction type methods on problems of the type (1.1.2). We will, in the paragraph, give a small overview on this type of methods but we will not do this in much detail as this is not within the scope of the thesis.

The methods for which tracking guarantees exist are typically either first or second order methods. First order methods can be shown to be able to track while no active effort is made by considering the time-varying nature of the problem. However, second order methods, typically based on homotopy or continuation methods [44] take an active approach in attempting to track the optimal solution to the optimization problem. These type of works assume that the derivative of the gradient with respect to time, i.e. \( \frac{\partial}{\partial t} \nabla f_0(p_0, x) \), is known for all \( x \). By combining this with the computation of the Hessian matrix, a prediction regarding the new optimal point is made based on the previous estimate. After this, once the new set of parameters is obtained, the estimate is corrected using the new data [45,46]. Interior point methods have also been extended to work in the time-varying setup [47] by using the prediction-correction framework used in [46]. In particular the work in [46] has been extended in [48] so as to deal with non-smooth components in the objective via forward-backward splitting ideas. However, the time-varying component of the objective is still required to be several times differentiable both with
respect to the optimization variable and time. In relation to minimizing the dynamic regret (1.1.4) we can see that by using high order derivatives we can predict how much gain we may obtain from utilizing computational resources to further minimize the objective.

While the discussed methods offer very high accuracies it is not always reasonable to assume that the required derivatives can be quickly computed or are known. This motivates the use of first order methods even if the statements that can be made are weaker. The work in [35] for time-varying problems has close similarities to the analysis of convergence given erroneous computations of gradients or proximal steps [53]. This is because having the problem vary in time can be seen as computing the gradient or proximal steps of a static problem of interest inaccurately. Consequently, just like how much error an algorithm can tolerate depends on how quickly it converges in the error free case; how quickly, and in what size of a neighbourhood of the optimal point the iterates remain in, in the time-varying case, depends on the performance of the method in the static case.

Inspired by the work in [35], the work in [49] establishes the tracking ability of ADMM in distributed set-ups. While differentiability with respect to the optimization variable is still a requirement, higher order derivatives are not necessary. Only first order differentiability is required to guarantee linear convergence of decentralized ADMM in the static case. A particular charm of ADMM in the time-varying case is that due to its proximal nature, the method will remain convergent under any choice of step-size. This is especially relevant when the Lipschitz continuity or strong convexity constant are allowed to change over time as well. We will exploit this fact in one the works included in the thesis. A unifying view on first order methods for decentralized and time-varying optimization is given in [54]. As pointed out in [54], given that we have a static bound on the primal-dual drift\(^2\), linear convergence is required in order to guarantee that we asymptotically remain in a neighbourhood of the optimal point. In one of the works presented in this thesis, we challenge this notion for a specific class of problem.

1.1.2 Time-Varying Networks

Finding solutions to problems of the type (1.0.3) in a distributed fashion allows the use of scattered resources and data while providing robustness to the system. However, in many applications the network does not remain static throughout the entire process. This is especially true if the communication between agents is done via a wireless network. The work [14] discussed in Section 1.1 establishes the convergence of (sub)-gradient descent under a \(B\)-connectivity assumption. A sequence of graphs \(\{G(k)\}_{k \geq 0}\) is \(B\)-connected if accumulating the links in a window of length \(B\) gives a (strongly) connected graph. This allows for some instances of the network to be disconnected while, at the same time, guaranteeing that all nodes are

\(^2\)Primal-dual drift denotes the movement of the primal dual optimal point from one time instant to the next.
1.1. A brief history of distributed optimization

reached sufficiently often. Perhaps more dramatically than in the case of static networks, developments on distributed optimization on time-varying networks have been closely tied to those of average consensus. This makes intuitive sense as average consensus is a fundamental block of most distributed optimization methods. While all the work included in this thesis assumes undirected graphs, introducing the techniques used to balance directed graphs is of relevance to understand the difficulty of extending some of our results to directed graphs.

A directed graph creates an inherent imbalance in the network as more information may flow in than out in some of the agents. This effect has to be “balanced” for agents to eventually agree and for them to obtain the network average as opposed to a weighted average. Before going into more details regarding techniques that take care of this imbalance we will distinguish between column stochastic mixing matrices $C : 1_n^T C = 1_n^T$ as opposed to row stochastic mixing matrices $R : R 1_n = 1_n$, where $1_n$ denotes the vector of all ones of dimension $n$. Having column stochastic mixing matrices corresponds to a more restrictive set-up as compared to row stochastic matrices. The usage of column stochastic matrices implies that the minimum knowledge each agent has is knowledge of both out and in degree, while in the case of row stochastic matrices it is implied that each agent has knowledge of its in degree. While row stochastic matrices makes more practical sense, as they capture the essence of the agents simply broadcasting their updates, we will see that the use of row stochastic matrices comes with added difficulties when dealing with time-varying networks.

Balancing a directed graph is the goal of both push-sum and graph balancing. The type of “balancing” one has to do when dealing with a column as opposed to row stochastic matrix is slightly different. We illustrate this with the simple static average consensus case. Let $x(0) \triangleq [x_1(0), \ldots, x_n(0)]^T$ be a vector containing the private quantities $x_i(0)$ held by each agent $i = 1, \ldots, n$. The agents’ goal is to obtain the network average by exchanging information with their neighbours via a directed network, i.e. we want an interate and distributed procedure for which $x(\infty) = \bar{x} \triangleq \frac{1}{n} 1_n 1_n^T x(0)$.

Assume the matrix $C$ is column stochastic and the network is strongly connected. Consequently, the matrix has all eigenvalues strictly within the unit circle with the exception of one eigenvalue of value 1 with left eigenvector $d_c$ and right eigenvector $1_n$. By recursively applying the iterate $x(t) = C x(t-1)$ we have that

$$x(\infty) = d_c 1_n^T x(0),$$

(1.1.5)

where $d_c$ is a stochastic vector. Equation (1.1.5) conveys the message that while the network average has indeed been computed, each agent merely sees a scaled version of it. On the other hand, assume that the network is represented by a row stochastic matrix $R$ and the network is strongly connected. Then, the matrix $R$ has all its eigenvalues strictly within the unit circle with the exception of a single

---

3The in and out degree of an agent is how many agents are within her in and out neighbourhood
Algorithm 1.1.1 Push-sum method

1: Initialize $y(0) = 1_n$, $u(0) = x(0)$.
2: for $t = 0, 1, \ldots$ do
3: Each agent $i$ sends $c_{ji}u_i(t)$ and $c_{ji}y_i(t)$ to each $j \in N_i,_{out}$ and receives $c_{ij}u_j(t)$ and $c_{ij}y_j(t)$ from agents $j \in N_i,_{in}$:
   
   $$y(t) = C y(t - 1)$$  \hspace{1cm} (1.1.9)
   $$u(t) = C u(t - 1)$$  \hspace{1cm} (1.1.10)

4: Each agent $i$ computes $x_i(t) = y(t) - 1u_i(t)$, i.e.,
   
   $$x(t) = \text{diag}(y(t))^{-1}u(t)$$  \hspace{1cm} (1.1.11)
5: end for

Eigenvalue with value 1 with left eigenvector $1_n$ and right eigenvector $d_r$. Then, by recursively applying $x(t) = Rx(t - 1)$ we obtain

$$x(\infty) = 1_n^T d_c x(0)$$  \hspace{1cm} (1.1.6)

where $d_c$ is a stochastic vector. In this case, we see that the agents agree on the quantity but instead of having computed the network average they computed a weighted average with weights $d_r$. Equations (1.1.5) and (1.1.6) already illustrate that the difficulty of compensating for a column stochastic matrix will be lower than compensating for a row stochastic matrix. This is because for the column stochastic matrix the compensation can be done a posteriori, i.e., once an arbitrary number of averaging steps has already been performed while in the row stochastic case one must compensate at every step and before exchanging with the neighbours one more time. In other words, if the quantities $d_c$ and $d_r$ can be computed or estimated, we could reach true average consensus by

$$x(\infty) = \text{diag}(nd_c)^{-1}d_c 1_n^T x(0),$$  \hspace{1cm} (1.1.7)
$$x(\infty) = 1_n d_c \text{diag}(nd_r)^{-1}x(0).$$  \hspace{1cm} (1.1.8)

For the column stochastic case, a solution was proposed in [106]. The push-sum method [106] performs the iterates illustrated in Algorithm 1.1.1 where $N_i,_{out}$ denotes the out neighbourhood of agent $i$, and $N_i,_{in}$ denotes the in neighbourhood of agent $i$.

Note that, for push-sum, $y(\infty) = d_c 1_n^T 1_n = nd_c$, leading to

$$x(\infty) = \text{diag}(nd_c)^{-1}d_c 1_n^T x(0)$$

and implying that we have achieved average consensus. A similar, albeit more demanding method, can be used for row stochastic matrices. The first instance in
1.1. A brief history of distributed optimization

Algorithm 1.1.2 Balancing method proposed in [107]

1: Initialize $Y(0) = I_{n \times n}$, $u(0) = x(0)$.
2: for $t = 0, 1, \ldots$ do
3: Each agent $i$ sends $r_{ji}y_i(t)$ to each $j \in N_{i,\text{out}}$ and receives $r_{ij}y_j(t)$ from agents $j \in N_{i,\text{in}}$:
   \[
   Y(t) = RY(t - 1) \quad (1.1.12)
   \]
   \[
   (1.1.13)
   \]
4: Each agent sets $x_i(t - 1) = \frac{x_i(t - 1)}{y_i(t)}$
5: Each agent $i$ sends $r_{ji}x_i(t - 1)$ to each $j \in N_{i,\text{out}}$ and receives $r_{ij}x_j(t - 1)$ from agents $j \in N_{i,\text{in}}$:
   \[
   x(t) = Rx(t - 1) \quad (1.1.14)
   \]
6: end for

which such an approach has been used for decentralized optimization can be found in [107]. In [107] instead of having each agent $i$ exchange with its neighbours a scalar quantity $y_i(t)$ the agents will exchange an entire vector $y_i(t) \in \mathbb{R}^n$; the balancing method proposed in [107] is illustrated in Algorithm 1.1.2. For this method the agents must know how many agents in total are in the network so as to set the length of $y_i(0)$. Let us define a matrix $Y(t) \triangleq [y_1(t) \ldots y_n(t)]^T$ where each vector $y_i(t)$ is present at agent $i$.

The intuition for the convergence of Algorithm 1.1.2 is simple but establishing that this is the case is slightly more involved. The idea is that as $Y(t) = R^t$ and, therefore, in the limit the diagonal elements of $Y(\infty)$, $y_{ii}(\infty)$, will match the elements of the right eigenvector of the matrix $R$, $d_{ri}$, i.e. $y_{ii}(\infty) = d_{ri}$. Based on the iterates in Algorithm 1.1.1 and in Algorithm 1.1.2 we can illustrate further the complication of using Algorithm 1.1.2 on time-varying graphs.

Let us first take a closer look at Algorithm 1.1.1. It is clear that $x(0) = \bar{x}_i1_n + (I_n - \frac{1}{n}1_n1_n^T)x(0)$, where $1_n$ is the identity matrix of size $n \times n$ and $\bar{x}_i$ can be any of the elements of $\bar{x}$. Then, by element-wise division of the $x$ and $y$ iterates of the push-sum method we obtain
\[
\frac{x(k + 1)}{y(k + 1)} = \frac{\bar{x}_iC^{k}1_n + C^{k}(I_n - \frac{1}{n}1_n1_n^T)x(0)}{C^{k}1_n}, \quad (1.1.15)
\]
where the term with $(I_n - \frac{1}{n}1_n1_n^T)x(0)$ is not relevant as it will vanish. This is because the right eigenvector of $C$ is $1_n$ which corresponds to the only eigenvalue that is equal to 1. Since $1_n$ is in the null-space of $I_n - \frac{1}{n}1_n1_n^T$, the term $C^k(I_n - \frac{1}{n}1_n1_n^T)\}$
vanishes exponentially fast in $k$. The term which contains the mean value $\bar{x}_i$ has its multiplying term $C^k_1n$ completely compensated by the denominator $C^k_1n$ and therefore the effect of having a directed network is fully compensated at each iteration and not just asymptotically. In other words, we rely on the effect of the iterates in $y$ at each iteration $k$ rather than on its asymptotic properties. This will notion is critical when dealing with time-varying networks. Note that the same argument can not be made for Algorithm 1.1.2. This is because instead of compensating for the effect of the network at each iteration, we are compensating for the network asymptotically, i.e. we are relying on the asymptotic effect of the sequences $\{y_{ii}(t)\}_{t \geq 0}$ rather than on their instantaneous effect.

While other techniques such as graph balancing [104] exist and have been successfully used in distributed optimization over static directed graphs [105] we will not discuss this as its relevance to the work presented in the thesis is low. However, we point out that the distributed sub-gradient method proposed in [105] uses graph balancing to deal with a graph that is represented by a column stochastic mixing matrix. Graph balancing requires more communication overhead than push-sum and the resulting method does not seem to have any performance gain over the usage of sub-gradient descent with push-sum as proposed in [108]. The method proposed in [108] is shown to convergence over $B$-strongly connected networks with column stochastic mixing matrices. Aside from the work proposed in [108] other methods have been proposed that guarantee convergence on $B$-strongly connected networks under different circumstances. NEXT [109] (In network Non Convex Optimization) proposes a method that guarantees convergence to a first order stationary point over directed graphs in the non-convex case. NEXT is later shown to be a particular case of SONATA [110] which will also converge on $B$-strongly connected time-varying directed graphs. The method proposed in [31] was the first method to be shown to converge linearly on $B$-connected time-varying directed graphs. The method proposed in [31] uses a gradient tracking scheme akin to that proposed in [30] but requires two information exchanges per iteration in order to guarantee convergence. This is because both the average of the primal variable and the moving average of the gradients is to be computed throughout the network. In order to track the network average gradients [31] draws inspiration from [111]. In [111] the authors propose a method to track the network average of time-varying quantities. The convergence of the method depends on how much the quantity one is tracking changes from one iteration to the next and this plays a key role in the convergence proofs. The method proposed in [31] is shown to converge both on $B$-connected time-varying undirected networks and on $B$-strongly connected time-varying directed networks. For the extension to directed networks the scheme used is push-sum [106]. Methods similar to [31] have been also proposed and established to converge even with uncoordinated step-sizes [129].

Dual-ascent type of methods have been shown to yield optimal convergence rates in the distributed set-up. However, no linearly converging extension of dual-ascent type methods over time-varying or directed graphs existed in the literature before PANDA, which is included in this thesis. Following similar lines, the work in [112]
establishes that accelerated dual ascent achieves optimal convergence rates in a
decentralized set-up if the undirected graphs are always connected and only a finite
number of changes are allowed. As opposed to the work in [112] our work allows for
the networks to even be disconnected at all times as long as $B$—connectivity holds.
Further, since PANDA will require solving an optimization problem at each iteration
we propose Eco-PANDA which linearizes the iterates of PANDA and therefore
reduces the methods’ computational burden by sacrificing some speed but keeping
all of PANDA’s advantages.
Contributions

In this chapter we introduce the contributions of this thesis to the literature of distributed optimization. We separate the contributions in two parts. In the first part, we introduce our works on decentralized optimization methods for time-varying problems. In the second part, we introduce our work on time-varying networks. When dealing with time-varying problems we assume that the problem parameters are changing with time while the constraint that enforces consensus remains static. On the other hand, when we deal with time-varying networks the only quantity changing with time is the network connecting the agents and as a consequence, the consensus constraint. For each of the included papers we provide a summary, an explanation of the main statement, and a sketch on how the statement is proven in the papers. Finally, we provide some discussion and possible extensions to each of the works. Note that the notation in each of the sections below is kept consistent with each of the papers the corresponding sections describe.

2.1 Time-Varying Problems

2.1.1 ADMM for Distributed Dynamic Beamforming

In this work we establish the ability of ADMM of tracking, in a decentralized manner, the minimum-power beamformers required to fulfill certain quality of service (QoS) constraints in a wireless Multiple Input Single Output (MISO) network. While in the entire paper the proofs are tailored to the beamforming problem we will argue in this section that a more general formulation of the result is possible.

Problem and Network Assumptions

The optimization problem at hand consists of finding the minimum power that $B$ base stations need to serve a total of $K$ mobile users while providing a certain QoS. Each user $k$ is served by a specific base station $b$. Let $b(k)$ denote the base station serving user $k$. Analogously let $U(b)$ denote the set of users that base station $b$ is serving. We assume that each of the base stations has $N_T$ antennas while each
of the mobile users has a single antenna. This implies that the wireless channel between each base station and mobile user can be represented by a vector \( h_{b_k} \in \mathbb{C}^{N_T} \). Further, we use the Signal To Interference and Noise Ratio (SINR) metric to represent the quality of service. The SINR is a ratio between the power the user receives, corresponding to the signal of interest, against interfering plus noise power. Hence, in a single base station and single user system, the strategy requiring the least power while guaranteeing a certain SINR would be for the beamformer \( \mathbf{w}_1 \propto h_{11} \). This maximizes the received power \( |h_{11}^H \mathbf{w}_1|^2 \) with the minimum required transmit power \( \| \mathbf{w}_1 \|^2 \). When more users are present an alternative strategy is zero forcing. Assume that we are in a scenario with a single base station and \( N_U \) total users. Then, we have the corresponding channels \( \{ h_{11}, \ldots, h_{1N_U} \} \). Given sufficient degrees of freedom, a possible strategy is to select each beam-former \( \mathbf{w}_i \) such that

\[
|h_{b_i}^H \mathbf{w}_i|^2 > 0 \quad \text{and} \quad h_{b_j}^H \mathbf{w}_i = 0 \quad \text{for} \quad j \neq i.
\]

This will counter interference completely, making the noise the only term that negatively impacts the SINR. This approach is good when the power of the noise \( \sigma_i^2 \) is very low, but is sub-optimal in general. This is especially clear in the case in which different mobile users suffer from different noise variances, i.e. no different power allocation is done to take into account the different noise variances. In order to deal with the power allocation optimally we formulate the following general problem with \( B \) base stations

\[
\begin{align*}
& \min_{\{ \mathbf{w}_k \in \mathbb{R}^{N_T} \}_{k=1}^{N_U}} \sum_{b=1}^{B} \sum_{k \in U(b)} \| \mathbf{w}_k \|^2 \\
\text{subject to} \quad & \frac{|h_{b(k)}^H \mathbf{w}_k|^2}{\sum_{i \in U(b(k)) \setminus k} |h_{b(k)}^H \mathbf{w}_i|^2 + \sum_{m \neq b(k)} \sum_{i \in U(m)} |h_{m(k)}^H \mathbf{w}_i|^2 + \sigma_k^2} \geq \gamma_k, \\
& \quad \text{SINR}_k(\mathbf{W}, \mathbf{H})
\end{align*}
\]

for \( k = 1, \ldots, N_U \)

where \( \mathbf{H} \) and \( \mathbf{W} \) are matrices containing all the channels and beam-former variables, respectively. Note that the inequality constraints are the SINR constraints. In the numerator we see the power of the signal of interest, i.e., the power received by user \( k \) from the base station that is serving \( k, b(k) \). In the denominator we find from left to right, the power of the interference caused by the serving of other users \( U(b(k)) \) by the same base station as \( k \), the power of the interference generated by other base stations serving their own users, and the power of the noise. The constraints (2.1.1b) represent the SINR we wish to guarantee for each user \( k \).

We are now to discuss which channels are known at which part of the system. In the single base station scenario we assumed that the base station is aware of the channel between itself and each of its users. The same is true for each the base stations. However, each base station is unaware of the channels between other base stations and the users, including its own. Further, as the users physically move, the channels \( \mathbf{H} \) will change.
2.1. Time-Varying Problems

In relation to the distributed optimization set-up, each of the agents of the system corresponds to a base station. Paper A contains a detailed explanation on how to re-write the optimization problem in (2.1.1) to one of the form of (1.0.3). This is done by adding additional variables that allow the decomposition of the SINR constraints in a way that each base station can perform updates even without the knowledge of channels involving other base stations. In particular, we write the problem in a way that is solvable via distributed ADMM. Due to the additional variables the objective function cannot be regarded as strongly convex. Further the constraints which appear are second order cones. This implies that none of the results in the literature which guarantee linear convergence of ADMM apply. Yet we are capable of providing a tracking statement under the assumption that the change in the channels from one time instance to the next is bounded by a sufficiently small quantity.

The base-stations are assumed to be connected by a reliable backhaul network. However, each base station is allowed to communicate exclusively with its neighbouring base stations defined by one hop communication over the backhaul network. Via this network, each base-station will communicate the parameters corresponding to a single iteration of distributed ADMM per channel change.

Main statement

Let $H^{[i]}$ be a matrix containing all channels of the system at time $i$. Further, let $W^{[i]}$ be a matrix containing all the beam-formers computed by the system, via distributed ADMM, at time $i$, and $W^{[i]}\star$ denote the matrix containing the optimal beam-formers at time $i$. We now introduce the main statement provided in the paper.

**Theorem 2.1.1.** Let $\{H^{[i]}\}_{i=0}^{\infty}$ be a sequence of channels that lie within a compact set $H$. Assume that the set $H$ is such that $\forall H \in H$ the problem (2.1.1) is feasible.

Given arbitrary positive constants $\epsilon_1 > 0$ and $\epsilon_2 > 0$, there is some $\delta > 0$ for which ADMM generates a sequence of beamformers $\{W^{[i]}\}_{i=0}^{\infty}$ for which the distance to the optimal beam-formers is guaranteed to fulfil

$$\limsup_{i \to \infty} \|W^{[i]} - W^{[i]}\star\|_F \leq \epsilon_1,$$

(2.1.2)

where $\| \cdot \|_F$ denotes the Frobenius norm. Further, the SINR of each user $k$ is guaranteed to fulfil

$$\liminf_{i \to \infty} \text{SINR}_k(W^{[i]}, H^{[i]}) \geq \gamma_k - \epsilon_2,$$

(2.1.3)

whenever $\|H^{[i]} - H^{[i-1]}\| \leq \delta$ for all $i \geq 1$.

As will be argued with more detail within the paper, the statement above guarantees the existence of a bound on sub-optimality and infeasibility that vanishes as $\delta \to 0$ making the statement non-trivial. However, the converse of the theorem is
not necessarily true in our case, i.e., we cannot guarantee that for any arbitrarily large $\delta > 0$ there exist non trivial bounds on sub-optimality and infeasibility. This is a weaker statement than those provided in, for instance, [35] where linear convergence of the method in the static cases can be established. While the result may not guarantee bounds for arbitrarily large $\delta$, they are guaranteed for sufficiently small $\delta > 0$. This implies that there will always exist a sufficiently large, yet bounded number of iterations to perform between channels $H^{[i]}$ and $H^{[i-1]}$ under which we can still provide tracking guarantees. This is established in a Corollary in the paper and allows us to claim that, if we understand the channel as a continuously varying quantity, there exists a sufficient increase of sampling frequency that allows us to track the optimal set of beam-formers.

**Proof Structure**

The proof relies on the convergence proof of ADMM provided in [73]. We define a Lyapunov function $V$ measuring the distance between the ADMM iterates and the primal-dual optimal point. This is well defined for the problem at hand because the primal-dual optimal point is unique for each $H^{[i]}$. We then assume some smoothness conditions on the primal-dual optimal points with respect to the channels, and smoothness conditions on the primal-dual iterates with respect to previous iterates and the channels. By using these smoothness conditions we establish that if decentralized ADMM is initialized such that the Lyapunov function is upper bounded by a certain quantity $\mu_u$ there exists a sufficiently small variation in the channel so as to guarantee that for all primal-dual points leading to Lyapunov values upper and lower bounded by $\mu_u$ and $\mu_l$ respectively, decentralized ADMM always manages to compensate for the variation in the channel. This implies that there exist a sufficiently small $\delta$ such that as $i \to \infty$ the iterates are such that the Lyapunov function yields values smaller or equal to $\mu_l$. We then relate the Lyapunov function to each of the quantities appearing in the main theorem providing the bounds. Finally, we establish that all the smoothness conditions we assumed are fulfilled for the beam-forming problem.

**Discussion and Possible Extensions**

In this paper we considered a very specific application. However, the proof technique can be generalized to any optimization problem for which the smoothness conditions can be established. In particular the problem should be of the form

$$\min_{\{x_i\}_{i=1}^n, \{y_i\}_{i=1}^n} \sum_{i=1}^n f_i(x_i)$$

subject to

$$f_i(x_i, y_i, p^T_i) \leq 0$$

$$Ey = 0,$$

where $y \triangleq [y^T_1, \ldots, y^T_n]^T$, $E$ is the mixing matrix and $t$ denotes the time index.
2.1.2 Dynamic Power Allocation for Smart Grids via ADMM

In this work we consider an economic dispatch problem on a micro-grid. In the micro-grid set-up, a small portion of the grid is connected to the main grid but is managed by an independent entity or coordinator. The goal of the coordinator is to allocate power consumption and production while respecting a set of constraints at any given time. These constraints will represent the power balance in the micro-grid and upper and lower bounds on the capacity of generators and consumers. These quantities may vary over time depending on the demand/supply, especially if renewable energies are present in the system.

Previous works on dynamic economic dispatch such as [83] do not consider upper and lower bounds on the consumed or generated power, we extend the work in [83] so as to be able to consider box constraints on the consumed and produced power.

Problem and Network Assumptions

We consider the dynamic economic dispatch problem, with \( n \) agents in which \( m \) are producers and the remaining are consumers, given by

\[
\min_{\{p_i\}_{i=1}^{n}} \quad \sum_{i=1}^{m} C_i(p_i, k) - \sum_{i=m+1}^{n} U_i(p_i, k) \tag{2.1.7a}
\]

subject to

\[
p_i[k] \leq p_i \leq p_i[k], \quad i = 1, \ldots, n \tag{2.1.7b}
\]

\[
\sum_{i=1}^{n} p_i = P[k], \tag{2.1.7c}
\]

where all the quantities parametrized by \( k \) are those that are allowed to vary with time. Note that the indexing and the quantity is irrelevant. In fact, the analysis still holds even if \( m \) is allowed to change with time. Each of the functions \( C_i(p_i, k) \) is convex in \( p_i \) and represents the cost of producing the power \( p_i \). Each of the functions \( U_i(p_i, k) \) is concave in \( p_i \) and represents the utility each consumer \( i \) obtains by consuming the power \( p_i \). The quantity \( P[k] \) represents the amount of power that must be produced in excess or the amount of power provided externally to the micro-grid. The inequality constraints represent constraints on how much power a consumer or producer is allowed to consume or produce. Note that in this case the bounds on power are allowed to change which may represent different consumer preferences and different producing capacities at different times. Finally, the cost of production and the utility a user gets may also be time varying.

In order to make the tracking statement we assume that each of the functions \( C_i(\cdot, k) \) and \( -U_i(\cdot, k) \) is at least \( \sigma \)-strongly convex and have at most \( L \)-Lipschitz gradients. We also assume that the primal-dual drift is bounded. Finally, we assume that all nodes can reach the system operator.
Proof Structure and Main Statement

Under the assumptions discussed in the previous section, we establish that for each $k$ the primal-dual optimal point is unique therefore making a primal-dual tracking statement well defined. We then propose a master-slave variant of ADMM tailored to the economic dispatch problem. We establish that given a single ADMM step we can provide a feasible set of powers $p^{[k]}$ which fulfil

$$\limsup_{k \to \infty} \|p^{[k]} - p^{[k]}\| \leq c,$$  \hspace{1cm} (2.1.8)

where $c$ is a constant proportional to the primal-dual optimality drift and $k$ denotes the time and iteration index.

2.1.3 On Decentralized Tracking with ADMM for Problems with Time-Varying Curvature

The work in [49] establishes that decentralized ADMM is capable of tracking the solution of a sequence of optimization problems with the structure

$$\min_{\{x_i\}_{i=1}^n, z} \sum_{i=1}^n f_i(x_i, k)$$ \hspace{1cm} (2.1.9a)

subject to

$$Ax + Bz = 0,$$ \hspace{1cm} (2.1.9b)

where each function $f_i(\cdot, k)$ is privately known by agent $i$ and is at least $\mu$–strongly convex and has at most $L$–Lipschitz continuous gradients at each time instance $k$. The equality constraint (2.1.9b) enforces all agents to agree in the value of $x_i$. The additional variable $z$ has a component per graph edge and is required to solve the problem in a decentralized manner using ADMM. The assumptions do not allow for the bounds on the curvature to become arbitrarily bad. Further, the authors require the primal-dual optimal drift to be uniformly bounded. In this work we extend the work in [49] to allow for arbitrarily badly conditioned functions and relax the bounds on the primal-dual optimal drift.

Problem and Network Assumptions

We model the sequence of optimization problem as follows

$$\min_{\{x_i\}_{i=1}^n, z} \sum_{i=1}^n f_i(x_i, \Theta_k)$$ \hspace{1cm} (2.1.10a)

subject to

$$Ax + Bz = 0,$$ \hspace{1cm} (2.1.10b)

where $\{\Theta_k\}_{k \geq 0}$ is a Markov process in a generalized state space $S$. Note that this implies that the Lipschitz continuity constant of the gradients $L(\Theta_k)$ and the strong convexity constants $\mu(\Theta_k)$ of the objective functions are dependent on $\Theta_k$. 
We assume that the Markov process has a unique probability distribution $\pi$, and is geometrically ergodic. The reason for this is intuitively explained in the following subsection. We assume that there exists a set $C \subset S$ with strictly positive probability $\pi(C) > 0$ such that for all $\theta \in C$ the objective functions $f_i$ are at least $\mu-$strongly convex and have at most $L-$Lipschitz continuous gradients, i.e., $\forall \theta \in C$, $\kappa(\theta) \triangleq \frac{L(\theta)}{\mu(\theta)} \leq \frac{L}{\mu} < \infty$. We also assume that the primal-dual optimal drift has finite fourth order moments, i.e., we replace the deterministic bound by a stochastic one. As a consequence, our tracking statement is also stochastic. Finally, we assume that the network connecting the agents in bidirectional and connected.

Main Statement

Let $u(k)$ be a vector containing the primal-dual iterates of ADMM at time $k$. Further, let $u^*(\Theta_k)$ denote a primal-dual optimal solution to the problem (2.1.10). Note that, in this case, these may no be unique as we allow for strong convexity to be lost. A detailed discussion can be found in the paper. Then, given the optimization problems (2.1.10) and under the assumptions mentioned in the previous subsection we have that

$$\limsup_{k \to \infty} \mathbb{E}[\|u(k) - u^*(\Theta_k)\|^2] \leq C,$$

(2.1.11)

where $C > 0$ is a polynomial in the fourth order moments of the primal-dual optimal drifts. Further, the value of the polynomial vanishes as the fourth order moments vanish. An exact formulation of this polynomial is provided in the paper.

Proof Structure

The proof relies on the convergence proofs of ADMM in both [49] and [19]. These are used to claim that $\|u(k) - u^*(\Theta_k)\| \leq q(\Theta_k)\|u(k - 1) - u^*(\Theta_k)\|$ for some factor $q(\Theta_k) \in (0, 1]$, where $\Theta_k$ is an outcome of $\Theta_k$. Note that $q = 1$ corresponds to the case where linear convergence can not be guaranteed because the assumptions on the objective are not stringent enough. This happens whenever the condition number $\kappa(\Theta_k) \triangleq \frac{L(\Theta_k)}{\mu(\Theta_k)}$ is not bounded.

The proof of the main statement is based on establishing that a set $C$, in which the properties of the function are good, and, therefore, $q(\Theta) \leq q_C < 1 \forall \theta \in C$, is visited often enough to guarantee linear convergence in the mean square deviation sense. Since the process $(\Theta)_{k \geq 0}$ is not memoryless, dependence of $q(\Theta_i)$ with $q(\Theta_i)$ for any $i$ causes problems in the analysis. Because of this, we require that the Markov process is such that it “forgets” often enough. Note that if we were dealing with a Markov process defined on a countable space we would require only that the process is irreducible and aperiodic to define a renewal time with a geometric tail. This would guarantee the existence of a renewal process such that, at every time a specific state is entered, the chain can be considered to be started anew. Such states are termed atoms. Unfortunately, in a general state space no atoms are
generally present. Part of our proof relies on constructing a larger process \( \{ \Phi_k \}_{k \geq 0} \) in which the original process \( \{ \Theta_k \}_{k \geq 0} \) is embedded. This larger process can then be shown to have properties that allow us to claim some “forgetting” properties.

To build the process \( \{ \Phi_k \}_{k \geq 0} \) with the appropriate properties the process \( \{ \Theta_k \}_{k \geq 0} \) must be irreducible and aperiodic. Also there must exist a small set \( \mathcal{C} \), constants \( b < \infty \), \( \beta > 0 \), and a function \( V \geq 1 \) finite for some \( \Theta \in \mathcal{S} \) satisfying

\[
\Delta V(\Theta) \leq -\beta V(\Theta) + b \mathbb{1}_C(\Theta),
\]

where \( \mathbb{1}_C \) is the characteristic function of the set \( C \), and \( \Delta V(\Theta) = \mathbb{E}_{\Theta_k}[V(\Theta_{k+1}) - V(\Theta)] \) where the expectation is taken with respect to the transition probability density. The drift condition \((2.1.12)\) may seem restrictive in the sense that the requirement seems to be very specific for the set \( C \). However, under mild conditions it can be shown that the entire state space \( \mathcal{S} \) can be covered by a countable number of compact small sets. It suffices then that one of these sets is such that the condition number of the functions parametrized by the values of \( \Theta \) it includes is bounded. If the drift condition in \((2.1.12)\) holds, a Markov process is geometrically ergodic (the converse is also true), i.e., the \( m \)-hop transition probability converges exponentially fast in \( m \) to the unique stationary distribution. We now remind the reader the definition of a small set and then illustrate how to construct the process \( \{ \Phi_k \}_{k \geq 0} \). Let \( P^m(\Theta, B) \) denote the \( m \)-hop transition probability from state \( x \) to the measurable set \( B \). Then, a set \( C \) is \( (\nu, m) \)-small if there exits \( \beta > 0 \), an integer \( m \geq 1 \), and a measure \( \nu \) such that

\[
P^m(\Theta, B) \geq \beta \nu(B), \quad \forall B \in \mathcal{B}(\mathcal{S}), \forall \Theta \in C,
\]

where \( \mathcal{B}(\mathcal{S}) \) denotes the Borel algebra on \( \mathcal{S} \). Both here and in the paper we will operate under the assumption that this holds for \( m = 1 \). The extension for larger \( m \) is straightforward and is therefore skipped.

In order to construct the split chain \( \{ \Phi_k \}_{k \geq 0} \), we have to fist of all split the state space and the measures defined on \( \mathcal{B}(\mathcal{S}) \). We first split the spaces by defining \( \mathcal{S}_0 \triangleq \mathcal{S} \times \{0\} \) and \( \mathcal{S}_1 \triangleq \mathcal{S} \times \{1\} \). Both copies are equipped with copies \( \mathcal{B}(\mathcal{S}_0) \) and \( \mathcal{B}(\mathcal{S}_1) \) of the \( \sigma \)-field \( \mathcal{B}(\mathcal{S}) \). Let \( \mathcal{S} = \mathcal{S} \times \{0, 1\} \). Any sub-set \( A \subset \mathcal{S} \) will have its analogous copy \( A_0 \subset \mathcal{S}_0 \) and \( A_1 \subset \mathcal{S}_1 \). If \( \lambda \) is any measure on \( \mathcal{B}(\mathcal{S}) \), then we split the measure into two measures on \( \mathcal{S}_0 \) and \( \mathcal{S}_1 \). In particular, let

\[
\lambda^\ast(A_0) \triangleq (1 - \beta)\lambda(A \cap \mathcal{C}) + \lambda(A \cap \mathcal{C}^c)
\]

\[
\lambda^\ast(A_1) \triangleq \beta \lambda(A \cap \mathcal{C})
\]

where the constant \( \beta \) and \( \mathcal{C} \subset \mathcal{S} \) are those of the small set condition. Note that \( \lambda \) is the marginal measure induced by \( \lambda^\ast \), i.e.

\[
\lambda^\ast(A_0 \cup A_1) = \lambda(A).
\]

The final step in the construction of \( \{ \Phi_k \}_{k \geq 0} \) is to split the transition probabilities. Let us denote by \( \theta_0 \triangleq [\theta, 0] \) and \( \theta_1 \triangleq [\theta, 1] \). In particular, we define the split
2.1. Time-Varying Problems

Figure 2.1: Construction of the split chain by adding an additional binary random variable. This allows for the construction of the atom $C_1$.

probabilities for $x_i \in \hat{S}$ as

\[ \hat{P}(\theta_0, \cdot) = P^*(\theta, \cdot), \quad \theta_0 \in S_0 \setminus C_0 \] (2.1.17)

\[ \hat{P}(\theta_0, \cdot) = (1 - \delta)^{-1}(P^*(\theta, \cdot) - \delta \nu^*(\cdot)), \quad \theta_0 \in C_0, \] (2.1.18)

\[ \hat{P}(\theta_1, \cdot) = \nu^*(\cdot), \quad \theta_1 \in S_1, \] (2.1.19)

where the sets $C_0 \triangleq C \times \{0\}$ and $C_1 \triangleq C \times \{1\}$. The transition probabilities described above imply that the process $\{\Phi_k\}_{k \geq 0}$ behaves just like $\{\Theta_k\}_{k \geq 0}$ on the top half $S_0$ (See Figure 2.1). Once the process reaches the set $C_0$ it will, with probability $1 - \beta$, remain in $C_0$ and with probability $\beta$ drop to $C_1$. Note that the shaded part of the figure is non-accessible and $C_1$ becomes now an atom. This is because the transition from any $\theta_1 \in C_1$ is independent of where in $C_1$ we start from, and, therefore, when the process leaves $C_1$ we can think of it as leaving a single state with strictly positive probability $\pi^*(C_1) = \beta \pi(C) > 0$.

Since all the measures on $B(S)$ can be seen as marginal measures of those defined on $B(\hat{S})$ we replace the expectations with respect to $\{\Theta_k\}_{k \geq 0}$ with expectation with respect to $\{\Phi_k\}_{k \geq 0}$. Once we have done this, we can see that after we leave the set $C_1$ the behaviour of the process is independent to whatever had happened before. This implies that the behavior of the chain in between visits to $C_1$ is independent and identically distributed.

The next step is to establish that the visits to the set $C_1$ happen frequently enough. In order to do this we establish that under appropriate conditions on the Markov process $\{\Theta_k\}_{k \geq 0}$ the random variable dictating how much time passes between visits to $C_1$ has an exponentially decaying tail.

By combining the two statements described above we are capable of establishing the main statement of the paper.

Discussion and Possible Extensions

The result presented in the paper does not allow for the use of constraints. The extension should be straightforward enough in the case where the constraints are
fulfilled strictly within a small set \( C \). They would then be allowed to be fulfilled tightly elsewhere.

A further extension would be to allow for the objective function to become non-differentiable in some instances. This would work as long as the sub-gradients are assumed to be bounded. If the sub-gradients are not bounded, we would have issues bounding the optimal dual multipliers associated with the consensus constraint.

Due to space restrictions no proven examples of processes fulfilling the required assumptions were introduced in the paper submitted to CDC. However, it can be shown that an order 1 Autoregressive Process (AR) fulfils the assumptions. This is left to be included in the extended version of the paper.

### 2.2 Time-Varying Networks

Here we introduce our work on time-varying networks. Mainly we introduce PANDA and its linearized version, Eco-PANDA. Since the main statement in both cases holds under the same conditions, we introduce the problem and network assumptions here. We deal with optimization problems of the form

\[
\min_{\{x_i \in \mathbb{R}^p\}_{i=1}^n} \sum_{i=1}^n f_i(x_i) \tag{2.2.1}
\]

subject to

\[
(W(k) \otimes I_d)x = 0 \tag{2.2.2}
\]

where \( W(k) \in \mathbb{R}^{n \times n} \) is a mixing matrix representing the network connecting the agents at time \( k \). The entire system is represented by a sequence of time-varying graphs \( \{G(k)\}_{k \geq 0} \) with \( G(k) = \{V, E(k)\} \). We assume throughout the paper that the graphs are undirected. At each time instant, each agent \( i \) is allowed to exchange variables with their immediate neighbours at that time, \( N_i(k) \). We further assume that the sequence of graphs \( \{G(k)\}_{k \geq 0} \) is \( B \)-connected for \( B < \infty \).

We assume that each of the objective functions is at least \( \mu \)-strongly convex with \( \mu > 0 \) and has at most \( L \)-Lipschitz continuous gradients \( L < \infty \).

#### 2.2.1 PANDA

PANDA is the first dual method with linear convergence guarantees on time-varying networks. PANDA was first proposed in Paper D and with the proofs and further details provided in Paper E. As opposed to the work in [31], we require a single communication exchange per iteration. Further, as opposed to [34], we allow for the network to become disconnected as we only require \( B \)-connectivity.

**Main Statement**

We provide \( R \)-linear convergence guarantees for PANDA. We provide an upper bound for the convergence rate which depends on the properties of the sequence...
2.2. Time-Varying Networks

Figure 2.2: Sequences involved in establishing linear convergence of PANDA via the small gain theorem.

\( \{G(k)\}_{k \geq 0} \), the condition number of the objective function, and the selected step-size. Note that the bounds in papers D and E are slightly different. This is because the bound in E is tighter than the one found in paper D. In both cases the worst case convergence rate is obtained via the small gain theorem. In particular, we are to select a rate and step-size pair that make a specific gain strictly smaller than 1. The resulting function of rate and step-size is complicated and consequently, in order to obtain a closed for the rate, further bounding is used. When writing paper E we found tighter bounds on the gain with respect to the rate.

The iterates of the method are provided now to make the coming discussion more understandable.

\[
\begin{align*}
    x(k+1) & \triangleq \arg \min_x f(x) - x^T y(k) \\
    z(k+1) & \triangleq (W(k) \otimes I_p)z(k) + (x(k+1) - x(k)) \\
    y(k+1) & \triangleq y(k) - \alpha (x(k+1) - z(k+1))
\end{align*}
\] (2.2.3)

Proof Structure

The proof of convergence borrows much of the ingredients from the proof of convergence of DIGing [31]. As the authors of [31], we use the small gain theorem on a specific choice of sequences so as to establish that they all converge at a linear rate. The small gain theorem ties together a set of sequences. This is illustrated in Figure 2.2 where each node of the graph corresponds to a sequence. Each arrow in the graph implies a bound, i.e., \( r \rightarrow x^\perp \) implies that \( x^\perp \) can be bounded by \( r \), where sequence index is omitted for simplicity. The gain from one node to itself, when moving clockwise in the graph in Figure 2.2 must be strictly smaller than one in order to guarantee convergence.
We now discuss what quantity each of the sequences represent. The quantity \( r(k) \triangleq y(k) - y^* \), where \( y(k) \) corresponds to the dual update in PANDA and \( y^* \) corresponds to the dual optimal point, is the dual residual. We start the proof by establishing that a bound on the dual residual implies a bound on the quantity \( x^+(k) \triangleq (I_{np} - \frac{1}{n} I_n^T \otimes I_p)x(k) \), which denotes the network disagreement on the average of \( x(k) \). This is done by using the fact that the objective is strongly convex.

The next step is to establish that a bound on \( x^+(k) \) implies a bound on \( \Delta_{xz}^+ \). Let \( \Delta_{xz}^+(k) \triangleq (I_{np} - \frac{1}{n} I_n^T \otimes I_p)(x(k) - z(k)) \) indicates how far away the quantity \( x(k) - z(k) \) is from its own average. Within \( \Delta_{xz}^+(k) \), the only part we need to make an effort to bound by \( x^+(k) \) is \( z^+(k) \triangleq (I_{np} - \frac{1}{n} I_n^T \otimes I_p)z(k) \). Note that by construction, \( z(k) \) is attempting to track the average of \( x(k) \). A bound on \( x^+(k) \) implies that \( x(k) \) is close to being an average vector making the job on an average tracking scheme easier.

A bound on \( \Delta_{xz}^+(k) \) clearly implies a bound on the quantity \( \Delta y(k) \triangleq y(k) - y(k-1) \). This can be seen by looking at the dual update in PANDA (2.2.5). This step is essential for the entire convergence proof as it incorporates the step-size in the cycle which allows us to adjust the gain by making the step-size sufficiently small.

Next, a bound on the change in the dual variables implies a bound on the change of the primal variables, \( \Delta x(k) \triangleq x(k) - x(k-1) \). If we can bound how much \( x \) moves, we have that an average tracking scheme performs better, as the quantity it is attempting to compute the average of is moving less. Therefore, we can bound \( z^+(k) \triangleq (I_{np} - \frac{1}{n} I_n^T \otimes I_p)z(k) \).

Note that the iterate in \( y \) can be seen as a projected gradient descent step in which an error in the gradient is being performed. This error is given by the quantity \( z^+(k) \). Consequently, we use the results in [121] for inexact oracles to establish the last arrow in the cycle.

We then finally establish that there exists a range of rates and step-sizes for which the gain is smaller than one.

**Discussion and Possible Extensions**

PANDA requires doubly stochastic matrices and not symmetric matrices; this may be an indication that adapting PANDA to directed graphs should be easier than adapting other dual methods where the mixing matrices are required to be symmetric. However, the extension to directed graphs is still not trivial. A possible way of extending PANDA to directed graphs could be by applying push-sum as is done for DIGing [31], leading to the following method

\[
x(k+1) \triangleq \arg \min_x f(x) - x^T y(k) \quad (2.2.6a)
\]

\[
v(k+1) \triangleq (C(k) \otimes I_p)v(k) \quad (2.2.6b)
\]

\[
z(k+1) \triangleq \text{diag}(v(k+1))^{-1}(C(k) \otimes I_p)z(k) + (x(k+1) - x(k)) \quad (2.2.6c)
\]

\[
y(k+1) \triangleq y(k) - \alpha (I_{np} - C(k) \otimes I_p)(z(k+1) - x(k+1)) \quad (2.2.6d)
\]
When using a doubly stochastic matrix, a single information exchange per iteration is sufficient. This is because the same matrix averages out the iterates via the product $\left( W(k) \otimes I_p \right) z(k)$, while at the same time performing an “anti-averaging” operation via the product $\left( I_{np} - W(k) \otimes I_p \right) x(k)$ in the sense that

$$1^T_n \left( I_n - W(k) \right) = 0^T_n.$$ 

In order to be able to perform the same type of operations on directed graphs, we require: a row stochastic matrix for the averaging operations, i.e., $\left( R(k) \otimes I_p \right) z(k)$, which we can emulate via push-sum, and, an anti-averaging operation which we perform with the left multiplication by the matrix $I_{np} - C(k) \otimes I_p$. Note that if we attempt to establish the convergence of the iterates in (2.2.6) in the same way as with PANDA, for the second to last step in the cycle in Figure 2.2 we would have an additional error quantity $\left( (C(k) - \frac{1}{2} I_n I_n^T) \otimes I_p \right) x(k+1)$ which can not be bounded by the quantity $y(k) - y(k-1)$. Consequently, establishing the convergence of Push-PANDA is left for future work.

Another possible extension would be to have PANDA work in a broadcast setup. Note that based on the discussion in Section 1.1.2 it is not straightforward to extend PANDA to work on row stochastic matrices. This is because, as opposed to when using column stochastic matrices, we do not compensate for the unbalanced graph at each iteration but we do this asymptotically. Since the networks should be allowed to vary, we have that the right eigenvalue to which the balancing method is supposed to converge changes over time and does not settle. However, an extension to the broadcast case on static row stochastic graphs may be possible.

PANDA can be accelerated via Nesterov’s acceleration scheme and can be empirically seen to converge. A convergence proof of this is non-trivial and also left for future work. Dual ascent, just as PANDA relies on the strong convexity of the objectives as well as on the Lipschitz continuity of the objectives’ gradients. However, when using dual ascent the network at a specific time $k$ plays a role in determining the strong convexity of the Fenchel conjugate. This implies that it may be the case that the Fenchel conjugate is never strongly convex.

### 2.2.2 Eco-PANDA

Eco-PANDA (Economic PANDA) is a linearized version of PANDA that works under the same assumptions as PANDA. However, PANDA requires solving an optimization problem at each iteration. Very few useful problems are known to have easily computable Fenchel conjugates. Problems for which the Fenchel conjugate is easy to compute are referred to as dual friendly [34].

In Eco-PANDA the objective function is linearized at every iteration so as to be able to provide a close-form solution for the iterates in $x$. This makes the algorithm computationally cheaper while keeping all of its desirable properties. The convergence rate suffers from the approximation while still remaining R-linear over time-varying graphs.
Figure 2.3: Sequences involved in establishing linear convergence of Eco-PANDA via the small gain theorem. The differences with PANDA are outlined in red.

Main Statement

The main statement on Eco-PANDA holds under the same conditions as the main statement on PANDA. We establish the R-linear convergence of Eco-PANDA and provide an upper bound on its convergence rate. The convergence rate is dependent on the conditioning of the objective function, the properties of the time varying sequence \( \{G(k)\}_{k \geq 0} \), and the selection on step sizes.

Proof Structure

We establish the convergence of Eco-PANDA in a similar way to that of PANDA. However, more statements need to be made within the cycle of arrows to account for the additional error included in the linearizing of the function. These are singled out in red in Figure 2.3 and defined within the paper. While in the case of PANDA the error in the dual gradient is given by \( z^\perp(k) \), the error in Eco-PANDA is characterized by both \( z^\perp(k) \) and the error introduced by the linearization of the objective. The rest of the mechanics are the same.
Part II

Included Papers
Bibliography


