Distributed Optimization in Electric Power Systems: Partitioning, Communications, and Synchronization

Submitted in partial fulfillment of the requirements for

the degree of

Doctor of Philosophy

in

Electrical and Computer Engineering

Junyao Guo

B.S., Electronic Engineering, Tsinghua University

Carnegie Mellon University
Pittsburgh, PA

March 2018

Acknowledgements

I would like to express my appreciation to the many people who helped me during my Ph.D. study at Carnegie Mellon University. First and foremost, I would like to gratefully thank my wonderful advisors, Prof. Gabriela Hug and Prof. Ozan Tonguz. The research presented in this thesis would not have been possible without the many fruitful discussions with my advisors either before a white board or over the Skype, their carefully crafted comments on every draft article I wrote, or their high standards for my work, and myself. To Gabriela, for always being conscientious and responsible, for teaching me the importance of paying attention to details, for her permanent encouragement, support, and trust, and for being an inspiring role model for me of an extraordinary female researcher. To Ozan, for teaching me to look for the counterintuitive in scientific research, for continuously challenging me and pushing me towards the difficult path that turns out to be the most rewarding, and for encouraging me to always aim for excellence.

I would like to thank ABB, a pioneering technology leader in industrial digitalization, and the National Science Foundation (grant number: ECCS 1252944), who sponsored my Ph.D. research. In particular, I would like to thank Dr. Xiaoming Feng from ABB, who also serves as my committee member, for constantly providing invaluable inputs and bringing in industrial experience for my research over these years and for hosting my trip to ABB Research. I would also like to sincerely thank my other two committee members, Prof. Rohit Negi and Prof. Stephen Boyd, for their insight and expert opinions that contribute to perfecting this thesis.

I am grateful for the friendship of my colleagues at CMU, Rui, Dinghuan, Kyri, Javad, Andrew, Xiao, Hameed, Chenye, Amin, Dmitry, Jesse, Jiun-Ren, Hsu-Chieh, Rusheng, Chiyu, Xiaoqi, Yaoqing, Xia and Liangyan. The many discussions, meetings and rehearsals that we had together helped me get prepared for my qualification exam and academic presentations, come up with new research ideas, and most importantly become a better researcher. We also had lots of fun together going to conferences, eating out, and going on field trips. I would also like to thank other members in Gabriela's group at ETH, Switzerland, where I spent the summer of 2016, for hosting me and making my stay at ETH fun and enjoyable.

I also want to say thank you to my lovely friends and roommates who accompanied me

the most during my study in Pittsburgh. To Mengdi, for knowing each other for more than ten years and accompanying each other during our first year of exploration and adventure in Pittsburgh. To Jingyi, for being an amazing roommate and friend, and for those delightful moments we shared together cooking, shopping (mostly online) and traveling.

Finally, I owe the greatest gratitude to my family. To my parents, Shuxu Guo and Wei Zhao, for your unconditional love and support, and your devotion to education which has instilled in me a never-ending thirst for knowledge from a very young age. To my beloved husband, Jun Yao, for showing up in our shared B-200 office, for always being the first person I could talk to whether I had a bad day or a new research idea, for keeping me calm and carry on, and for your love, respect and trust no matter we live under the same roof or thousands of miles apart. I cannot imagine going through all these years without your support.

Abstract

To integrate large volumes of renewables and use electricity more efficiently, many industrial trials are on-going around the world that aim to realize decentralized or hierarchical control of renewable and distributed energy resources, flexible loads and monitoring devices. As the cost and complexity involved in the centralized communications and control infrastructure may be prohibitive in controlling millions of these distributed energy resources and devices, distributed optimization methods are expected to become much more prevalent in the operation of future electric power systems, as they have the potential to address this challenge and can be applied to various applications such as optimal power flow, state estimation, voltage control, and many others.

While many distributed optimization algorithms are developed mathematically, little effort has been reported so far on how these methods should actually be implemented in real-world large-scale systems. The challenges associated with this include identifying how to decompose the overall optimization problem, what communication infrastructures can support the information exchange among subproblems, and whether to coordinate the updates of the subproblems in a synchronous or asynchronous manner.

This research is dedicated to developing mathematical tools to address these issues, particularly for solving the non-convex optimal power flow problem. As the first part of this thesis, we develop a partitioning method that defines the boundaries of regions when applying distributed algorithms to a power system. This partitioning method quantifies the computational couplings among the buses and groups the buses with large couplings into one region. Through numerical experiments, we show that the developed spectral partitioning approach is the key to achieving fast convergence of distributed optimization algorithms on large-scale systems.

After the partitioning of the system is defined, one needs to determine whether the communications among neighboring regions are supported. Therefore, as the second part of this thesis, we propose models for centralized and distributed communications infrastructures and study the impact of communication delays on the efficiency of distributed optimization algorithms through network simulations. Our findings suggest that the centralized communications infrastructure can be prohibitive for distributed optimization and cost-effective

migration paths to a more distributed communications infrastructure are necessary.

As the sizes and complexities of subproblems and communication delays are generally heterogeneous, synchronous distributed algorithms can be inefficient as they require waiting for the slowest region in the system. Hence, as the third part of this thesis, we develop an asynchronous distributed optimization method and show its convergence for the considered optimal power flow problem. We further study the impact of parameter tuning, system partitioning and communication delays on the proposed asynchronous method and compare its practical performance with its synchronous counterpart. Simulation results indicate that the asynchronous approach can be more efficient with proper partitioning and parameter settings on large-scale systems.

The outcome of this research provides important insights into how existing hardware and software solutions for Energy Management Systems in the power grid can be used or need to be extended for deploying distributed optimization methods, which establishes the interconnection between theoretical studies of distributed algorithms and their practical implementation. As the evolution towards a more distributed control architecture is already taking place in many utility networks, the approaches proposed in this thesis provide important tools and a methodology for adopting distributed optimization in power systems.

Contents

List of Tables

List of Figures

Chapter 1

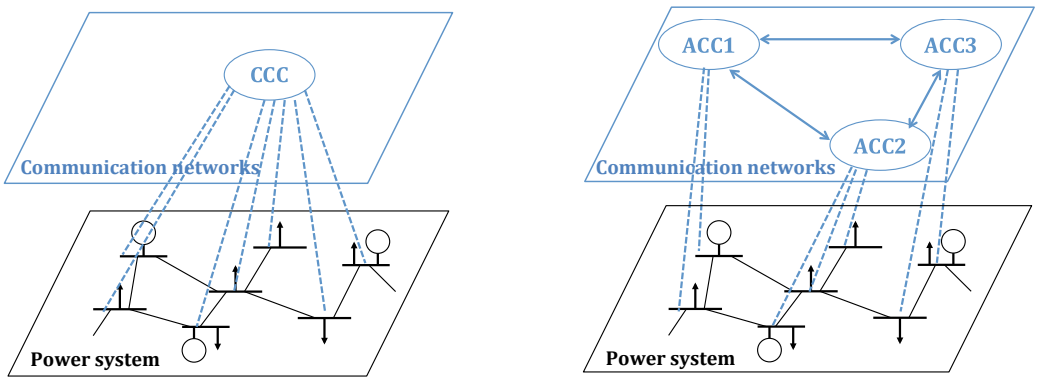
Introduction

1.1 Background and Motivation

Electric power systems are large-scale systems spanning countries and sometimes entire continents. Optimizing the operation of the grid is a challenging task due to the large number of variables and constraints and their inherently non-linear physical characteristics. Further challenges arise with the transition of the current grid to a smarter grid that integrates a large number of renewable energy resources, flexible loads, and storage devices, which entails an even larger increase in the size and complexity of the associated problems.

Traditionally, the optimal management of the grid in a large area has been and is still mostly solved as a single optimization problem at a Central Control Center (CCC). This limits the size of the problem that can be solved in a given amount of time. Moreover, as all the measurements need to be sent to one CCC, the communications load of the backbone network is high which may result in large communication delays or even failures of data delivery. To address these issues, there has been a growing interest in distributed optimization over the past two decades, where a large system is decomposed into smaller control areas, each having an Area Control Center (ACC) that makes local control decisions [1]. The ACC has similar functionalities as a Remote Terminal Unit (RTU) placed at the substation that can monitor the local system states, store measurement data, and communicate with other ACCs.

This paradigm shift from centralized to distributed control architecture is illustrated



(a) Centralized (b) Distributed

Figure 1.1: Conceptual illustration of centralized and distributed control architectures of power systems.

conceptually in Fig. 1.1. The blue dotted lines denote the communication links between a control center and the buses, while the blue double arrowed solid lines denote the communication links between pairs of neighboring control centers. The distributed optimization scheme studied in this thesis is based on geographical partition of the power system, where each ACC solves a subproblem locally. As the ACCs formulate their subproblems only with local information assuming some boundary conditions as fixed, they need to acquire information from neighbors to update these conditions periodically such that the overall optimality of the system can be achieved. However, the communications is only needed between directly connected neighboring areas while no centralized or master node is needed for the coordination of the ACCs. Also, the information to be exchanged are intermediate results of the subproblems, which usually include the variables at the boundary buses but not all local buses. Most studies on distributed algorithms assume that these algorithms are performed in an iterative and synchronized fashion; i.e., the communications start after all ACCs have solved their subproblems and the computation starts after all ACCs have finished the information exchange. The iterations proceed until some stopping criteria are fulfilled.

Figure 1.2 illustrates *where* distributed optimization takes place in power system operations. Given the current system state, distributed algorithms are run at ACCs. After the algorithm converges to a control solution, this solution is applied to change the control settings. Then the new system state is estimated, which enables the next round of distributed

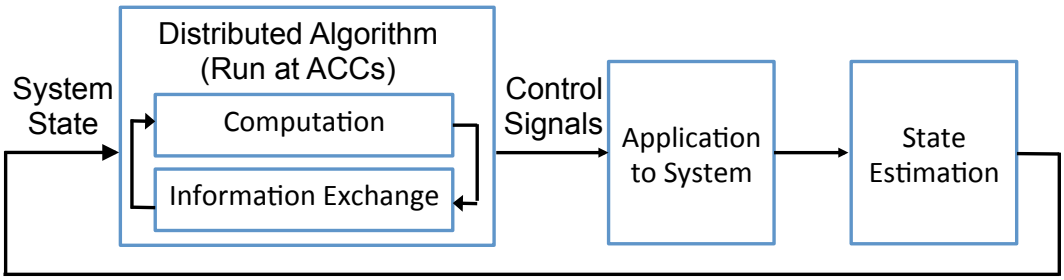


Figure 1.2: A sequence of power system operations.

optimization. The distributed optimization scheme could potentially alleviate the centralized computational and communications burdens, enhance the robustness and scalability of optimization in large-scale systems, as well as preserve the data privacy of the users in different parts of the grid.

One key application considered for distributed optimization is the Optimal Power Flow (OPF) problem, which is at the heart of power systems operations and planning. The most common objective of OPF is to schedule the generation units optimally subject to the power flow balances and operational constraints such as transmission line capacities. OPF problems have been studied for over half a century, which is required to be solved by system operators every five minutes in less than one minute [2]. However, due to the non-convexity introduced by the AC power flow equations, OPF is known to be a difficult problem to solve even in a centralized manner for large systems [3]. Currently, most system operators only solve linear approximations of this problem to which the solution is only reasonably acceptable but not optimal, which may result in unnecessary cost of tens of billions of dollars in the US annually. Distributed optimization is considered as one promising approach to solve the OPF problem for large-scale systems more efficiently. However, difficulties arise in the search for a distributed solution since optimality or even convergence cannot be guaranteed for most distributed methods on non-convex problems [4][5].

Distributed methods based on various decomposition techniques have been proposed to solve the OPF problem, including Lagrangian Relaxation [6], Augmented Lagrangian Relaxation [7, 8, 9] or more specifically Alternating Direction Method of Multipliers [5, 10], Optimality Condition Decomposition [11, 12, 13], and Benders Decomposition [14]. An overview of these techniques can be found in [4, 15]. However, the focus of these studies

is purely on the development of a decomposition method to ensure the convergence to the overall optimally of the system. The experiments in most of these studies are conducted on IEEE standard test systems whose size and complexity are not comparable with those of real power systems. While the IEEE standard test systems serve as good platforms to showcase the functionality of new algorithms, researchers still face a practical yet challenging question: can one successfully apply distributed algorithms to real-world large-scale power systems problems, in particular to non-convex OPF problems? Ultimately, the value of distributed optimization can only be understood and its full potential can only be realized if it can be applied to large-scale real power systems.

The challenges associated with answering this question are tightly related to the three practical issues that one needs to consider for implementation of any distributed optimization and control methods, namely, system partitioning, communications, and synchronization [1]. Specifically, one needs to identify how to decompose the overall optimization problem, what communications infrastructures can support the information exchange among subproblems, and whether to solve the subproblems in a synchronous or an asynchronous manner. The motivation for investigation into these three issues are further elaborated as follows:

- *System Partitioning.* While there are many studies on the development of distributed optimization methods, the test system used in most studies are either partitioned in an arbitrary way or configured by connecting several IEEE standard test systems using a few tie lines where the system naturally partitions into subsystems. Consequently, the results either do not reflect the realistic performance for a truly connected system and/or the best performance achievable by the presented distributed method. Moreover, for large-scale real-world networks whose topologies are hard to visualize, it is even challenging to search for an arbitrary partition of the system. Therefore, a systematic tool for system partitioning is needed for the application of distributed optimization methods.

- *Communications.* The convergence of the distributed solution to an overall optimal point of the entire system is dependent on the periodic information exchange among the subproblems. Hence, compared with the centralized optimization approach, communication plays a much more important role as it determines whether timely decisions can

be made to ensure an optimal and efficient operation of the entire grid. Nevertheless, communications is often assumed to have zero delays and errors, and thus neglected in most studies on distributed algorithms. While the assumption of perfect communications simplifies the theoretical analysis of distributed algorithms, it barely holds in practice. Hence, it is necessary to evaluate the impact of the conceivable communication infrastructures and technologies on the performance of distributed methods before any of these methods can actually be implemented.

Particularly, the following two critical questions need to be answered. First, in terms of implementation, what communication infrastructures and technologies should be adopted to support the information exchange involved in the distributed optimization process? Second, how much will the performances of distributed algorithms degrade with communication delays or failures in terms of convergence speed, solution quality, robustness, and so on, compared with the theoretical results where communications are assumed to be perfect? To answer these questions, an evaluation of the capability of the existing communications infrastructure of the grid is needed and the performance of distributed algorithms needs to be investigated with the communications overhead taken into account.

- *Synchronization.* The majority of distributed algorithms are developed based on the premise that the workers that solve subproblems are synchronized. However, synchronization may not be easily obtained in a distributed system without centralized coordination, which to a certain degree defeats the purpose of the distributed algorithm. Moreover, the sizes and complexities of subproblems are usually dependent on the system's physical configuration, and therefore are heterogeneous and require different amounts of computation time. Therefore, even if synchronization is achievable, it may not be the most efficient way to implement distributed algorithms. Furthermore, the communication delays among workers are also heterogeneous which are determined by the communication infrastructures and technologies used. In a synchronous setting, all workers need to wait for the slowest worker to finish its computation or communications. This may lead to the under-utilization of both the computation and communication resources as some workers remain idle for most of the time. Therefore,

asynchronous distributed optimization schemes need to be developed to address the limitation of synchronous schemes.

This thesis is dedicated to developing mathematical tools to address the above practical issues and provide insights into an efficient implementation of distributed optimization methods for large-scale OPF problems. As the first part of this thesis, we investigate how partitioning affects the speed of the distributed optimization methods and develop a partitioning method based on spectral clustering that improves the performance of two distributed algorithms. As the second part, we evaluate centralized and distributed communications infrastructures for their capabilities to support distributed methods. Specifically, we demonstrate how the network topologies can be constructed in different communications infrastructures to enable the required information exchange and evaluate the impact of communications delays on the efficiency of the distributed methods. As the sizes of the subproblems and the locations of the control entities are dependent on the system partitioning and the communication infrastructures, it is highly likely that the computation time and communication delays for solving different subproblems are heterogeneous, consequently, synchronous updates of the subproblems might not be efficient. Hence, as the third part of this thesis, we develop an asynchronous distributed optimization method, compare its performance with synchronous methods in terms of efficiency and solution quality, and show how the performance of this asynchronous approach can be affected by parameter tuning, system partitioning, and communication delays.

1.2 Literature Review

1.2.1 Distributed Large-Scale OPF

While various distributed optimization techniques have been proposed and applied to the OPF problem [6, 7, 8, 9, 5, 10, 11, 12, 13], only few studies demonstrate the scalability of the proposed distributed methods. In [16], two decomposition algorithms for solving the Security-Constrained OPF problem are investigated and applied to the Polish 3012-bus system. However, the OPF problem is decomposed according to contingency scenarios and not geographical regions, which is much harder. In [17], a method based on ADMM and

proximal message passing is proposed for solving a dynamic OPF problem, and is tested on networks with thousands of nodes. However, both the nodes and the branches are generated randomly, which could result in a network that is very different from real-world systems. Different from the aforementioned studies, this thesis contributes to the geographical decomposition of large-scale real transmission networks for implementing distributed methods.

To overcome some of the difficulties in solving non-convex OPF in a distributed manner, one approach proposes convexifying the OPF problem before applying a distributed algorithm [3, 18, 19, 20]. A common approach for convexification is based on semidefinite relaxation [21, 22]. However, due to lack of adherence to the original non-convex problem, the optimality achieved by this approach can only be ensured for some simple networks such as the IEEE benchmark systems and acyclic radial networks [21, 22]. Moreover, semidefinite programming solvers generally result in large computational efforts, hence, they do not scale well to problems in large transmission networks, which typically have thousands of buses and meshed topologies [23]. As an alternative to semidefinite programming with lower complexity, LinDistFlow [24] provides a convex approximation of power flow equations, which, however, is only shown to work for distribution networks. For optimizing distribution networks, the authors in [25] propose using second-order cone relaxation for convexifying the OPF problem before applying a distributed algorithm, and test this approach on a real-world 2065-bus distribution system. As transmission networks are meshed networks, it is not clear whether the results obtained in [25] still hold for these networks.

1.2.2 System Partitioning

Power system partitioning for purposes and applications other than distributed optimization are extensively studied. One group of studies aims to divide the overall problem into equally sized subproblems to enable the utilization of parallel processors using conventional optimization methods such as dynamic programming [26], contour tableau [27], simulated annealing [28], genetic algorithm [29], and Tabu search [30] to partition the problem. Power system partitioning with the objective to minimize the coupling between regions are studied in [31, 32, 33]. In [31], harmony search is proposed for network partitioning which aims to minimize the number of tie lines between clusters. In [32], a slow coherency based networking

partitioning method is proposed for network reduction by grouping elements into areas according to the strength of their coupling resulting in multiple areas which are weakly coupled among each other. In [33], an electrical distance based method combined with evolutionary algorithm is introduced to evaluate partitioning solutions using multi-metrics. Graph partitioning theory has also been applied to power system clustering which aims to minimize the power imbalance between generation and loads within islands to avoid cascading events [34, 35]. However, the main focus of these studies is on network reduction or islanding, which is very different from distributed optimization.

Other related work includes [36, 37], which focus on clustering for efficient computations. In [36], for instance, an approach to balance the size of subproblems and the coordination effort of these subproblems is proposed. In [37], a similar objective of computational load balancing as in [36] is imposed on the application of contingency screening to solve the DC OPF problem. However, no specific distributed optimization method is used in these studies and there is no quantification of the improvement in computational efficiency when using the optimal partition in terms of the number of iterations and convergence time. In this thesis, we propose a system partitioning approach that integrates the computational coupling between buses which has a strong impact on the performance of distributed optimization methods and quantify the efficiency gain of distributed algorithms when using the proposed partitioning approach.

1.2.3 Communications

While communications infrastructures in smart grids have been extensively investigated in recent years, most of the existing work focuses on applications in distribution networks. In [38], an overview over the communications challenges involved in energy management at the demand side using smart meters is presented. In [39][40], radial, meshed and hierarchical communications infrastructures are studied where their capability to deliver sensing and control data for autonomous networks management is evaluated. For Wide Area Measurement System (WAMS) applications, optimal designs of the communications infrastructures are proposed in [41, 42, 43], and their performances are evaluated using the OPNET modeler in [44, 45, 46, 47]. To explore the impact of communications networks on power system

operations, some co-simulation platforms have been developed to integrate the communications network simulation with the physical control of power system components and their dynamics [48, 49, 50]. Main applications of those co-simulation platforms include testing protection schemes and investigating cyber-security issues. However, the applications considered in all of the above studies are different from distributed optimization which has different communications requirements. The communications plane considered in this thesis is the Wide-Area-Network (WAN) that connects multiple control centers, whereas the aforementioned studies focus on the communications in power systems networks that connect control centers with end devices.

Compared with the above applications, the research on communications required by distributed algorithms is rather limited where most studies consider consensus-based algorithms. For example, in [51], the impact of different communications network topologies on the convergence rate of the Incremental Cost Consensus algorithm applied to the economic dispatch problem is investigated. In [52, 53, 54, 55], various communication delay models are proposed and integrated into the linear update equations of the nodes, and the effect of delays or uncertainty on the convergence of consensus algorithms is evaluated. However, for non-convex optimization problems, it is difficult to model the communications involved mathematically due to the fact that closed-form solutions do not exist in such situations. Therefore, this thesis presents a simulation-based approach that models the communications plane for distributed optimization and evaluates the impact of communication delays on distributed methods.

1.2.4 Synchronization

The synchronization issue has been systematically studied in the research fields of distributed computing with seminal works [56, 57]. While the concept of asynchronous computing is not new, it remains an open question whether those methods can be applied to solving non-convex problems. Most of the asynchronous computing methods proposed can only be applied to convex optimization problems [56, 58]. These methods therefore can only solve convex approximations of the non-convex problems which may not be exact for all types of systems [59, 60, 61, 62].

Recently, there are a few asynchronous algorithms proposed that tackle problems with some level of non-convexity. Among these algorithms, the Alternating Direction Method of Multipliers (ADMM) has been found to work particularly well on non-convex problems when extended into an asynchronous setting. The studies of distributed ADMM method can be categorized according to whether they tackle convex or non-convex problems and whether the implementation is synchronous or asynchronous. For convex optimization and synchronous implementation, the linear convergence rate of distributed ADMM has been established under different settings [63, 64, 65, 66]. It has also been shown in [67] that ADMM could converge faster under additional regularity assumptions on the objective function such as strong convexity and smoothness. Researchers have also discovered that the convergence rate of ADMM strongly depends on the choice of the algorithm parameter and optimal parameter selection methods have been proposed in [68, 64, 66]. Beyond choice of algorithm parameter and properties of objective function, the convergence rate of ADMM on consensus problems is also influenced by the network topology [64]. For asynchronous ADMM applied to solve convex problems, its convergence rate has been quantified for specific problem structures. Linear convergence rate of asynchronous ADMM has been established in [69, 70] when applied to an unconstrained convex consensus problem and shown to be related with the number of workers, the amount of information collected by each worker and the delay of the worker. It is also shown in [71] that asynchronous ADMM can achieve linear convergence when a set of constraints and variables are randomly activated at each ADMM iteration.

For non-convex optimization with synchronous implementation, recent works have shown that under certain assumptions such as the regularity of objective functions and constraints, boundedness of variables, identifiable local minimum of subproblems or sufficiently large penalty parameter, synchronous ADMM is guaranteed to converge to a limit point that satisfies the KKT conditions of the problem with non-convex objectives [72] or non-convex constraints [72, 73, 74, 10]. However, regarding applying asynchronous ADMM to non-convex optimization, the studies on its convergence behavior is rather limited. Asynchronous ADMM is shown to converge for non-convex problems with the additional assumption of bounded delay of all workers in the consensus problem [75, 76]. In [77], an asynchronous incremental ADMM method is shown to converge with sublinear rate on a problem with

non-convex objectives under certain bounds imposed on the level of asynchrony.

Apart from ADMM, there are other asynchronous methods proposed to tackle non-convex problems. For example, [78] proposes an asynchronous probabilistic approach for non-convex distributed optimization and establishes an upper bound on the number of iterations required to achieve a certain level of feasibility. However, as the authors stated in [78], their proposed bound is mainly of theoretical interest as the choice of parameters used for their analysis is not likely to be set in the same way in practice, which shows the importance of empirical studies conducted for specific applications.

The focuses of the aforementioned studies are different from the problem considered in this thesis in the following aspects. First, the problem formulations studied are different from the OPF problem which has non-convex constraints that include variables from multiple regions. Second, the underlying communication graph is different. The problem studied in most research is the consensus problem where the workers are homogeneous and they aim to find a common system parameter. The underlying network topology is either a full mesh network where any pair of nodes can communicate [57] or a star topology with a centralized coordinator [69, 75]. Different from the consensus problem, this thesis considers partition-based optimization where the workers represent regions or subnetworks with different sizes. Furthermore, each worker only communicates with its physically connected neighbors and the information to be exchanged only contains the boundary conditions but not all local information. Thereby, the workers are heterogeneous and the communication topology is a partial mesh network with no centralized/master node needed.

Moreover, for partition-based distributed optimization problem with non-convex constraints, there lack both empirical and theoretical results on how fast asynchronous method converges and what factors affect its convergence rate, which are generally application dependent. Therefore, we conduct empirical studies in this thesis which could provide insights on how to better utilize distributed asynchronous ADMM for the application to the OPF problem.

1.3 Contributions

The main contributions of this thesis are outlined as follows:

- System Partitioning:
  - A system partitioning technique based on spectral clustering is proposed using an affinity metric based on the Hessian matrix of the considered optimization problem which measures the computational coupling among the buses while deploying distributed optimization algorithms. The proposed spectral partitioning technique is shown to improve the convergence performance of two distributed optimization algorithms, namely, Optimality Condition Decomposition (OCD) and Alternating Direction Method of Multipliers (ADMM), which provides a tool to determine how the problem should be optimally decomposed to obtain the largest benefit from distributed optimization.
  - With the proposed spectral partitioning technique, ADMM is shown to converge for large-scale AC OPF problems on realistic transmission networks. This result is a milestone as it shows for the first time that distributed optimization is a practically viable approach to key optimization problems in large-scale real power systems with no pre-defined partitions.
  - The proposed spectral partitioning technique is also applied to a multi-step OPF problem based on Model Predictive Control (MPC), and is shown to significantly improve the efficiency of distributed MPC. This finding shows that the same partition can be used for multiple time steps and that the proposed partitioning method is indeed practical as the partitions found do not need to be changed frequently.
- Communications:
  - A modeling and simulation method is proposed using the OPNET network modeler to model the communications plane and emulate the communications required in distributed optimization.

  - The impact of communication delays on the efficiency distributed algorithms is evaluated using the proposed models in the context of possible communications infrastructures and technologies.
  - It is shown that the centralized communications infrastructure can be prohibitive for distributed optimization in large-scale systems and that the performance of distributed algorithms can be significantly improved by using a more distributed communications infrastructure; suggesting cost-effective migration paths to a more distributed communications infrastructure.
  - The strong relationship between distributed optimization and power system communications is investigated, therefore bridging the gap that exists between the communications research community and the distributed optimization research community in how these two research communities see the problem of deploying distributed methods in power systems.

- Synchronization:
  - A distributed asynchronous optimization method is proposed based on ADMM and its convergence for non-convex optimization problems is established under the assumption of bounded communications delay and some other mild conditions on the objective function and constraints. To the best of our knowledge, this is the first time that ADMM is shown to be convergent under an asynchronous setting for a partition-based distributed optimization problem with non-convex constraints that contain variables from different subproblems.
  - Empirical studies are carried out on multiple large-scale systems to better characterize the practical performance of asynchronous ADMM in terms of parameter tuning, system partitioning and communications delays, which show that asynchronous ADMM can be more efficient than its synchronous counterpart with proper settings on large-scale systems.

With the contributions outlined above, the ultimate goal of this thesis is to develop tools and a methodology that enable the actual implementation of distributed optimization algorithms and to provide empirical evidence which suggests that distributed optimization

can be a viable approach to tackle large-scale optimization problems in power systems if implemented with proper system partitioning, parameter tuning and communications technologies. The outcome of this thesis establishes the interconnection between theoretical studies of distributed algorithms and their practical implementation in power systems, and provides important insights into how existing solutions in power system operations can be used or need to be extended for deploying distributed optimization methods.

1.4 Thesis Outline

The chapters of this thesis are outlined as follows:

- **Chapter 2: Problem Statement and Methods** formulates the AC Optimal Power Flow (OPF) problem and gives an overview of the Interior Point Method and two distributed optimization methods utilized and built upon in this thesis, namely, Optimality Condition Decomposition (OCD) and Alternating Direction Method of Multipliers (ADMM).

- **Chapter 3: Spectral Partitioning** proposes a power system partitioning approach based on spectral clustering that speeds up the convergence of the OCD method. This partitioning approach defines an affinity metric that measures the computational coupling between pairs of buses and groups the more coupled buses into one region. Simulation results on IEEE test systems demonstrate the effectiveness of the proposed partitioning approach.

- **Chapter 4: Applications of Spectral Partitioning** applies the proposed spectral partitioning approach in conjunction with the ADMM method and to a multi-step OPF problem. Simulation results on several large-scale real-world transmission networks show that ADMM with spectral partitioning is a viable approach to tackle non-convex OPF problems in large systems. Experiments on the multi-step OPF problem with wind and storage integration show that the partitions found by spectral partitioning at one operating point is applicable to a wide range of operating points.

- **Chapter 5: Role of Communications Plane** proposes a modeling and simulation approach using the OPNET modeler to emulate the communications process required by distributed optimization algorithms. Communications technologies and schemes that support distributed optimization are first reviewed, and two communications infrastructures, namely, the distributed and centralized infrastructures are then evaluated. The efficiencies of the distributed optimization methods, OCD and ADMM, are compared with communication delays taken into account and the impact of delays on distributed optimization is discussed.

- **Chapter 6: Asynchronous Distributed ADMM** proposes an asynchronous distributed algorithm based on ADMM. The proposed method allows local updates with partial information from neighbors. Sufficient assumptions are made and the convergence of asynchronous ADMM is established. Then empirical studies are carried out that evaluate the impact of parameter tuning, system partitioning and communication delays on the proposed asynchronous ADMM method.

- **Chapter 7: Closure** concludes the thesis and discusses potential directions for future research in this area.

Chapter 2

Problem Statement and Methods

We deploy two distributed optimization algorithms in this research, namely, Optimality Condition Decomposition (OCD) [11] and Alternating Direction Method of Multipliers (ADMM) [10], which have been applied in multiple studies to solve the OPF problem in a distributed manner. In this chapter, we outline the distributed AC OPF formulations and the mechanisms of these two methods. We also review the Interior Point approach which is used to solve the OPF problem in a centralized manner in this thesis.

2.1 AC Optimal Power Flow Problem

Throughout this thesis, we consider the non-convex AC OPF problem, which is at the heart of power systems operations and planning. The most common objective of OPF is to schedule the generation units optimally subject to the power flow balances and operational constraints such as transmission line capacities. While there are multiple formulations for the OPF problem, we consider one standard formulation which is also used in the MATPOWER optimal power flow solver [79] as follows:

$$\underset{V,P,Q}{\text{minimize}} \quad f(P) = \sum_{i=1}^{n_b} \left(a_i P_i^2 + b_i P_i + c_i\right) \tag{2.1a}$$

$$\text{subject to} \quad P_i + jQ_i - P_i^d - jQ_i^d = V_i \sum_{j \in \Omega_i} Y_{ij}^* V_j^* \tag{2.1b}$$

$$P_i^{\min} \leq P_i \leq P_i^{\max} \tag{2.1c}$$

$$Q_i^{\min} \leq Q_i \leq Q_i^{\max} \tag{2.1d}$$

$$\theta_i^{\min} \leq \theta_i \leq \theta_i^{\max} \tag{2.1e}$$

$$V_i^{\min} \leq |V_i| \leq V_i^{\max} \tag{2.1f}$$

$$(|V_i||V_j||Y_{ij}|\cos(\theta_i - \theta_j - \delta_{ij}))^2 + (|V_i||V_j||Y_{ij}|\sin(\theta_i - \theta_j - \delta_{ij}))^2 \leq (S_{ij}^{\max})^2, \tag{2.1g}$$

for $i = 1, \ldots, n_b$ with n_b denoting the total number of buses in the system. The variables to be optimized in problem (2.1) are (V_i, P_i, Q_i), which stand for the complex voltage, the active power output and the reactive power output of generator at bus i, respectively. A quadratic cost model for each generator is used and (a_i, b_i, c_i) are the cost parameters of generator at bus i. These parameters are set to zero if the bus is not connected with any generator. (P_i^d, Q_i^d) are the active and reactive load at bus i, Y_{ij} is the ij-th entry of the line admittance matrix, and Ω_i is the set of buses connected to bus i. $(|V_i|, \theta_i)$ and $(|Y_{ij}|, \delta_{ij})$ are the magnitudes and angles of V_i and Y_{ij}, respectively, and $S_{ij}^{\max}$ is the maximum transmission capacity of line ij. Constraints (2.1b) denote the power flow balances at each bus, (2.1c) to (2.1f) denote the bounds on the variables, and (2.1g) denote the thermal line constraints. Setting the reference angle of the slack bus to zero is taken care of in (2.1e). For other nodes, $\theta_i^{\min}$ and $\theta_i^{\max}$ are set to $-\pi$ and π, respectively. Problem (2.1) is non-convex due to the non-convexity of the AC power flow equations (2.1b) and line limit constraints (2.1g).

To implement distributed optimization methods, the centralized OPF problem (2.1) needs to be decomposed into subproblems. Geographical decomposition of the system is considered where the transmission network is partitioned into a number of subnetworks each assigned to an ACC for solving a local subproblem; i.e., ACC k only optimizes the variables associated with the buses within region k. The connectivity of the areas is based on the network topology; i.e., we say two regions are neighbors if their associated subnetworks are physically connected and some variables of their variable sets appear in the same constraints. We use K to denote the total number of regions and $\mathcal{R}_k, k = 1, ..., K$, to denote the set of buses assigned to region k with $\mathcal{R}_k \cap \mathcal{R}_l = \emptyset, \forall l \neq k$. The transmission lines that connect the buses in two neighboring regions are referred to as tie lines in the rest of this thesis. We introduce $x_k = \{(V_i, P_i, Q_i) \mid i \in \mathcal{R}_k\}$ and $f_k(x_k)$ to denote the variables and the local objective function in region k, respectively. Then problem (2.1) can be formulated in terms

of sets of variables, i.e.,

$$\underset{x}{\text{minimize}} \quad \sum_{k=1}^{K} f_k(x_k) \tag{2.2a}$$

$$\text{subject to} \quad c(x_1, \ldots, x_k, \ldots, x_K) \leq 0 \tag{2.2b}$$

$$s_k(x_k) \leq 0; k = 1, \ldots, K, \tag{2.2c}$$

Due to the fact that the objective function (2.1a) is separable in terms of generators, it can be easily transformed into the form of sum of local objective functions (2.2a). Constraints (2.2b) are defined as coupling constraints as they contain variables from multiple regions, and thus couple the updates of variables in these regions. Note that we include the variable sets of all regions in the expression of constraints (2.2b) just for the simplicity of notation, where each coupling constraint actually includes the variable sets of neighboring regions but not all sets of variables. The coupling constraints include the power flow balance (2.1b) at the buses connected to the tie lines, i.e., buses at the boundaries between regions, and the thermal line limits (2.1g) of tie lines. Constraints (2.2c) are defined as the non-coupling constraints as they only contain variables from one region. Note that (2.2b) and (2.2c) include both equality and inequality constraints, where the inequality constraints can be handled with an Interior Point approach.

2.2 Interior Point Method

Various methods can be used to handle the inequality constraints in an optimization problem, where this thesis uses an Interior Point approach. Consider a general optimization problem

$$\underset{x}{\text{minimize}} \quad f(x) \tag{2.3a}$$

$$\text{subject to} \quad g(x) = 0 \tag{2.3b}$$

$$h(x) \leq 0. \tag{2.3c}$$

The Interior Point method transforms all inequality constraints into equality constraints by adding non-negative slack variables z to the left-hand-side of (2.3b). A barrier function is also added to the objective function that enforces z to be greater than zero; i.e., problem

(2.3a) is transformed into the following form:

$$\underset{x}{\text{minimize}} \quad f(x) - \zeta \sum_{i=1}^{p} \ln(z_i) \tag{2.4a}$$

$$\text{subject to} \quad g(x) = 0 \tag{2.4b}$$

$$h(x) + z = 0 \tag{2.4c}$$

$$z > 0, \tag{2.4d}$$

where p denotes the number of inequality constraints and $\zeta > 0$ is the barrier parameter which has to become (almost) zero in the optimum. The Lagrangian function of problem (2.4) is given by:

$$L = f(x) - \zeta \sum_{i=1}^{p} \ln(z_i) + \lambda^T g(x) + \mu^T (h(x) + z). \tag{2.5}$$

The first-order necessary conditions, i.e., the Karush-Kuhn-Tucker (KKT) conditions for an optimal solution of problem (2.4) are formulated as

$$\frac{\partial L}{\partial x} = \frac{\partial f(x)}{\partial x} + \left(\frac{\partial g(x)}{\partial x}\right)^T \cdot \lambda + \left(\frac{\partial h(x)}{\partial x}\right)^T \cdot \mu = 0 \tag{2.6a}$$

$$\frac{\partial L}{\partial \lambda} = g(x) = 0 \tag{2.6b}$$

$$\frac{\partial L}{\partial \mu} = h(x) + z = 0 \tag{2.6c}$$

$$\frac{\partial L}{\partial z} = \mu - \zeta \cdot diag\left(\frac{1}{z}\right) = 0 \tag{2.6d}$$

$$\mu \geq 0 \tag{2.6e}$$

$$z > 0. \tag{2.6f}$$

To solve problem (2.3), one can search for the solution to the KKT conditions (2.6) by using the Newton-Raphson approach to iteratively update the variables. Specifically, at each iteratoion, the changes in the variables can be calculated by:

$$\begin{pmatrix} \Delta x \\ \Delta \lambda \\ \Delta \mu \\ \Delta z \end{pmatrix} = -H^{-1} \cdot \begin{pmatrix} \frac{\partial L}{\partial x} \\ \frac{\partial L}{\partial \lambda} \\ \frac{\partial L}{\partial \mu} \\ \frac{\partial L}{\partial z} \end{pmatrix} = - \begin{pmatrix} \nabla L & \nabla g^T & \nabla h^T & 0 \\ \nabla g & 0 & 0 & 0 \\ \nabla h & 0 & 0 & diag(1) \\ 0 & 0 & diag(z) & diag(\mu) \end{pmatrix}^{-1} \cdot \begin{pmatrix} \nabla L \\ g \\ h + z \\ \mu z - diag(\zeta) \end{pmatrix}, \tag{2.7}$$

where H is the Jacobian of the KKT conditions which is also the Hessian of the Lagrangian function (2.5). Note that in (2.6d), we multiply both sides of the equation by z. The update of variables in the Interior Point approach at the ν-th iteration are given as follows:

$$x^{\nu+1} = x^{\nu} + \alpha_z \Delta x^{\nu} \tag{2.8a}$$

$$\lambda^{\nu+1} = \lambda^{\nu} + \alpha_\mu \Delta \lambda^{\nu} \tag{2.8b}$$

$$\mu^{\nu+1} = \mu^{\nu} + \alpha_\mu \Delta \mu^{\nu} \tag{2.8c}$$

$$z^{\nu+1} = z^{\nu} + \alpha_z \Delta z^{\nu}, \tag{2.8d}$$

with

$$\alpha_\mu = \min\{1,\ 0.9995\alpha'_\mu\}, \quad \alpha'_\mu = \min_{\Delta\mu_i^\nu<0}\{\frac{-\mu_i^\nu}{\Delta\mu_i^\nu}\} \tag{2.9a}$$

$$\alpha_z = \min\{1,\ 0.9995\alpha'_z\}, \quad \alpha'_z = \min_{\Delta z_i^\nu<0}\{\frac{-z_i^\nu}{\Delta z_i^\nu}\}. \tag{2.9b}$$

Updating the variables this way guarantees that $\mu^{\nu+1} \geq 0$ and $z^{\nu+1} \geq 0$. Then ζ is also updated by

$$\zeta^{\nu+1} = \beta \frac{\mu^{\nu+1^T} \cdot z^{\nu+1}}{p}, \tag{2.10}$$

where β is set between 0.1 to 0.5 depending on the problem.

2.3 Optimality Condition Decomposition

2.3.1 Basic Algorithm

OCD is an extension to the Lagrangian Relaxation method which has shown improved convergence properties over Lagrangian and Augmented Lagrangian methods [12]. For the implementation of OCD, each coupling constraint (2.2b) in problem (2.2) is assigned to one subproblem, which results in the following formulation of the subproblem associated with any region k:

$$\underset{x_k}{\text{minimize}} \quad f(x_k) + \sum_{l=1,l\neq k}^{K} \overline{\lambda}_l^T c_l(\overline{x}_1, ..., x_k, ..., \overline{x}_K) \tag{2.11a}$$

$$\text{subject to} \quad c_k(\overline{x}_1, ..., x_k, ..., \overline{x}_K) \leq 0 \tag{2.11b}$$

$$s_k(x_k) \leq 0 \tag{2.11c}$$

where $\lambda_l, l \neq k$ denote the Lagrange multipliers of the coupling constraints $c_l(\cdot)$ assigned to regions other than k but included in the objective function of region k. The variables with a bar indicate that their values are fixed using the last updated values acquired from other regions. Each of these subproblems (2.11a)-(2.11c) is solved by deriving the first order optimality conditions and then applying Newton-Raphson to compute search directions Δx_k and $\Delta\lambda_k$. After each area carries out one iteration of the Newton-Raphson approach, it updates its variables $x_k^{\nu+1} \leftarrow x_k^{\nu} + \Delta x_k$ and multipliers $\lambda_k^{\nu+1} \leftarrow \lambda_k^{\nu} + \Delta\lambda_k$. Here, we also use ν to denote the iteration counter. Then the updated values of variables are exchanged among areas to facilitate the next iteration. Such process continues until the overall optimality of the system is reached.

The above described procedure of OCD is indeed equivalent to decomposing the Hessian matrix used in the centralized optimization approach. To solve problem (2.2) in a centralized manner, one can solve for its KKT conditions using the Interior Point approach. After assigning the coupling constraints (2.2b) to different regions, the Lagrangian function of problem (2.2) can be written as:

$$L = \sum_{k=1}^{K} f(x_k) + \lambda_k^T c_k(x_1, ..., x_K) + \gamma_k^T s_k(x_k) \tag{2.12}$$

where λ_k and γ_k are Lagrange multipliers for the coupling and non-coupling constraints, respectively. Note that the inequality constraints in $c_k(x_1, ..., x_K)$ and $s_k(x_k)$ in (2.12) are transformed into equality constraints by adding slack variables, and the objective function $\sum_{k=1}^{K} f(x_k)$ includes the barrier term associated with the Interior Point method. Denoting all the variables that need to be determined including the Lagrange multipliers by y, the above procedure is equivalent to solving the following system of linear equations to get the update of variables Δy:

$$H_{sys}\Delta y = -r; \qquad H_{sys} = \nabla_y^2 L, \quad r = \nabla_y L \tag{2.13}$$

Here, r denotes the KKT conditions and they all have to be zero at optimality for the algorithm to stop. H_{sys} denotes the Jacobian of the KKT conditions, which is also the Hessian of the Lagrangian function (2.12). By rearranging the variables in H_{sys} according

to the areas they are assigned to, (2.13) is transformed to:

$$H \begin{pmatrix} \Delta y_1 \\ \vdots \\ \Delta y_K \end{pmatrix} = - \begin{pmatrix} r_1 \\ \vdots \\ r_K \end{pmatrix} \tag{2.14}$$

where

$$r_k = \nabla_{y_k} L; k = 1, ..., K \tag{2.15}$$

$$H = \begin{pmatrix} H_{11} & \dots & H_{1K} \\ \vdots & \ddots & \vdots \\ H_{K1} & \dots & H_{KK} \end{pmatrix} \tag{2.16}$$

$$H_{kl} = \nabla^2 L_{y_k, y_l}; k, l = 1, ..., K \tag{2.17}$$

The off-diagonal blocks H_{kl} where $k \neq l$ contain non-zero entries that couple the updates of the variables in the subproblems, which does not allow independent solutions of the subproblems. The approach in OCD is to decouple the subproblems by setting the off-diagonal block elements, i.e., the elements in H_{kl}, $k \neq l$, to zero. Hence, in (2.14) H is replaced by $\overline{H}$ which has the form:

$$\overline{H} = \begin{pmatrix} H_{11} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & H_{KK} \end{pmatrix} \tag{2.18}$$

Consequently, the subproblems become decoupled and each area can carry out the following Newton-Raphson step

$$\Delta y_k = -H_{kk}^{-1} \cdot r_k \tag{2.19}$$

and update its variables $y_k \leftarrow y_k + \Delta y_k$. After this update, the updated values of variables associated with buses at the boundaries are exchanged among subproblems to enable the calculation of H_{kk} and r_k as some of the entries in H_{kk} and r_k contain variables from other subproblems.

The sufficient condition for the convergence of OCD to a KKT point of problem (2.2) is given in [12] as

$$c = \rho(I - \overline{H}^{*-1} H^*) < 1, \tag{2.20}$$

where I is the identity matrix and $\rho(M)$ denotes the spectral radius of matrix M. H^* and $\overline{H}^*$ are defined in (2.16) and (2.18) and the superscript $*$ indicates that these matrices are evaluated at the KKT optimal point. Note that local convergence properties of the Newton-Raphson method can be preserved if H^* is sufficiently close to $\overline{H}^*$, i.e., the off-diagonal blocks in H^* are relatively sparse [11]. Moreover, OCD converges to the centralized solution if (2.20) is fulfilled.

2.3.2 Additional Correction Term

In the OCD method, the update of the variables in each area is calculated while neglecting the inter-area couplings at each iteration, which leads to an increase in the number of iterations until convergence. To reduce the number of iterations of OCD, we add a correction term to the update in (2.19) according to

$$\Delta y_k = H_{kk}^{-1} \cdot (-r_k + \hat{r}_k), \tag{2.21}$$

where the idea is to choose $\hat{r}_k$ so as to compensate part of the error made by setting the off diagonal elements in the Hessian matrix equal to zero. We give the derivation of the correction term for the following simple case where the overall problem is divided into two subproblems resulting in solving

$$\begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \cdot \begin{bmatrix} \Delta y_1 \\ \Delta y_2 \end{bmatrix} = - \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}, \tag{2.22}$$

where one can get

$$\begin{aligned} \Delta y_1 &= (H_{11} - H_{12} H_{22}^{-1} H_{21})^{-1} \cdot (-r_1 + H_{12} H_{22}^{-1} \cdot r_2) \\ &\approx H_{11}^{-1} \cdot (-r_1 + H_{12} H_{22}^{-1} \cdot r_2) \qquad (2.23) \\ \Delta y_2 &= (H_{22} - H_{21} H_{11}^{-1} H_{12})^{-1} \cdot (-r_2 + H_{21} H_{11}^{-1} \cdot r_1) \\ &\approx H_{22}^{-1} \cdot (-r_2 + H_{21} H_{11}^{-1} \cdot r_1). \qquad (2.24) \end{aligned}$$

The approximations in (2.23) and (2.24) are reasonable because $H_{12}H_{22}^{-1}H_{21}$ and $H_{21}H_{11}^{-1}H_{12}$ are sparse and only have limited impact. Hence, generalizing the above derivation to K subproblems, the correction term is approximated by

$$\hat{r}_k = \sum_{l=1,l\neq k}^{K} H_{kl}H_{ll}^{-1} \cdot r_l. \tag{2.25}$$

The correction term is calculated in each area for its neighboring areas by utilizing the area coupling information contained in H_{kl}; i.e., in (2.25), $H_{kl}H_{ll}^{-1} \cdot r_l$ is calculated by area l and communicated to area k. Due to the sparsity of H_{kl} and r_l, the resulting term $H_{kl}H_{ll}^{-1} \cdot r_l$ is also sparse and only the non-zero elements need to be exchanged with the neighbors resulting in only a few additional values to be communicated instead of the entire vector. Intuitively, the required number of iterations until convergence can be reduced because the distributed update is closer to the centralized update. We name the proposed OCD with the additional correction term OCD-C. The convergence proof for OCD-C is derived in the Appendix A.

2.4 Alternating Direction Method of Multipliers

2.4.1 Problem Decomposition for ADMM

ADMM is a state-of-the-art distributed method that has often been shown to exhibit good performance for non-convex optimization. In [10], ADMM is applied to solve the OPF problem in a distributed manner. Similar to OCD, the power system is first partitioned into smaller regions, and a subproblem is formulated for each region. However, different from the way OCD handles the coupling constraints, ADMM decouples the regions by duplicating variables. Specifically, the power system is decoupled by duplicating the voltages at the boundary buses of each region. Constraints are then added to enforce the duplicated voltages to be equal to one another. Figure 2.1 illustrates this decoupling of the system where bus i and bus j are the buses that are connected by tie line ij. The voltages at bus i and bus j are duplicated, and the copies assigned to Region A are denoted by $V_{i,A}$ and $V_{j,A}$. Similarly, Region B is assigned the copies $V_{i,B}$ and $V_{j,B}$. To ensure equivalence with the original problem, the constraints $V_{i,A} = V_{i,B}$ and $V_{j,A} = V_{j,B}$ are added to the problem. These operations remove the tie line and separate the two regions. As discussed in [10], the

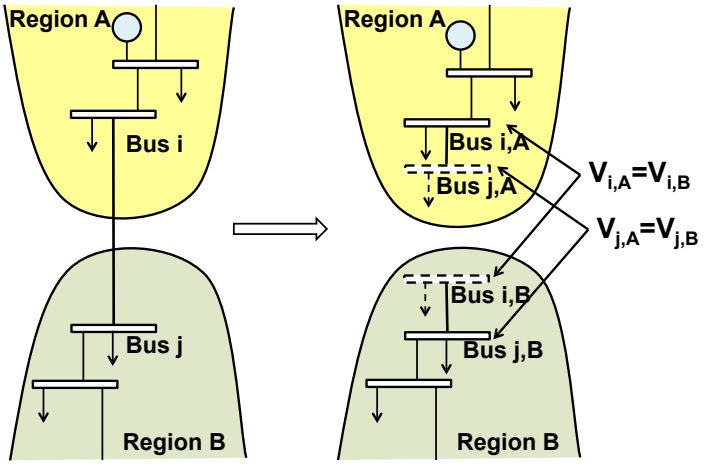


Figure 2.1: Duplicating voltages at boundaries of regions.

added constraints are equivalent to

$$\begin{aligned} V_{i,A} - V_{j,A} &= V_{i,B} - V_{j,B} \\ V_{i,A} + V_{j,A} &= V_{i,B} + V_{j,B}. \end{aligned} \tag{2.26}$$

The set $\mathcal{V}_k$ is introduced to denote the joint set of buses including the buses in $\mathcal{R}_k$ and the buses in neighboring regions that are directly connected to buses in $\mathcal{R}_k$. For each tie line ij with $i \in \mathcal{R}_A, j \in \mathcal{V}_A \setminus \mathcal{R}_A$ we further introduce two auxiliary variables $z^+_{i,j}$ and $z^-_{i,j}$ to Region A, and two auxiliary constraints

$$z^-_{i,j} = \beta^-(V_{i,A} - V_{j,A}), \qquad z^+_{i,j} = \beta^+(V_{i,A} + V_{j,A}), \tag{2.27}$$

where β^- and β^+ are scaling factors. Constant β^- is set to be larger than β^+ to give more weight to $V_{i,A} - V_{j,A}$, which is strongly related to the line flow through tie line ij [10]. Similarly, two auxiliary variables $z^-_{j,i}$ and $z^+_{j,i}$ are introduced to Region B. Hence, the feasible region of all the z's associated with tie lines is defined as

$$\mathcal{Z} = \{(z^-, z^+) \mid z^-_{i,j} = -z^-_{j,i},\ z^+_{i,j} = z^+_{j,i}, \forall (i,j) \in \mathcal{T}\}, \tag{2.28}$$

where $\mathcal{T}$ is the set of inter-region tie lines. Set (2.28) is derived from expressing the constraints in (2.26) using the definitions given in (2.27). Let $x_k = \{(V_i, P_i, Q_i) \mid i \in \mathcal{V}_k\}$ and $z_k = \{(z^-_{i,j}, z^+_{i,j}) \mid i \in \mathcal{R}_k, j \in \mathcal{V}_k \setminus \mathcal{R}_k\}$ denote all the primal variables and auxiliary variables associated with the buses in region k, respectively. Furthermore, $z = \{z_k | \forall k\}$. The OPF problem (2.1) can then be expressed in terms of variables assigned to different regions as

follows:

$$\underset{x,z}{\text{minimize}} \quad \sum_k f_k(x_k) \tag{2.29a}$$

$$\text{subject to} \quad A_k x_k = z_k, \ \forall k \tag{2.29b}$$

$$x_k \in \mathcal{X}_k, \ \forall k \tag{2.29c}$$

$$z \in \mathcal{Z}, \tag{2.29d}$$

where $f_k(x_k)$ is the generation cost in region k and constraint (2.29b) is obtained by expressing (2.27) using x_k and z_k. Constraint (2.29c) enforces the local feasibility constraints, namely, constraints (2.1b)-(2.1d) for $\forall i \in \mathcal{R}_k$, constraint (2.1f)-(2.1e) for $\forall i \in \mathcal{V}_k$, and constraints (2.1g) for transmission lines in region k. Note that by choosing A_k as the identity matrix, problem (2.29) reduces to the standard consensus problem where all workers should find a common variable z. Here we allow A_k to not have full column rank; i.e., the neighboring workers only need to find common values for the boundary variables but do not need to share information on all local variables, which greatly reduces the amount of information to be exchanged among regions. An important property of problem (2.29) is that if z is fixed, then problem (2.29) can be decomposed into subproblems where each subproblem only contains the local variables x_k. This property enables distributing the computations of ADMM to solve problem (2.29), as described next.

2.4.2 Distributed ADMM Algorithm

The ADMM algorithm minimizes the Augmented Lagrangian function of (2.29), which is given as follows [10]:

$$L(x, z, \lambda) = \sum_k \Big\{ f_k(x_k) + \lambda_k^\top (A_k x_k - z_k) + \frac{1}{2} \|A_k x_k - z_k\|_{\rho_k}^2 \Big\}, \tag{2.30}$$

where $\|x\|_\rho^2 = x^\top \text{diag}(\rho) x$ is the square of a weighted norm of x. The vector ρ is a vector of penalty parameters whose entries are increased during the iterative process to ensure convergence of ADMM [10] on large systems. With (2.30) formulated, the $\nu+1$-th iteration of ADMM consists of the following steps:

$$x^{\nu+1} = \text{argmin}_{x \in \mathcal{X}} \ \ L(x, z^\nu, \lambda^\nu) \tag{2.31a}$$

$$z^{\nu+1} = \text{argmin}_{z\in\mathcal{Z}} \;\; L(x^{\nu+1}, z, \lambda^{\nu}) \tag{2.31b}$$

$$\lambda^{\nu+1} = \lambda^{\nu} + \text{diag}(\rho^{\nu})(Ax^{\nu+1} - z^{\nu+1}). \tag{2.31c}$$

The x and λ updates can be solved locally as they only contain local variables. The z update involves information exchanges among regions, and can also be computed locally once the updated information from neighboring regions is acquired. Hence, the ADMM iterations can be carried out in a completely distributed fashion with only local information exchanges but no centralized coordination. Updating x requires solving non-convex subproblems, but with much smaller scale compared with the original centralized problem, which can be handled by many existing optimization solvers. Updating z solves a quadratic programming problem and updating λ is trivial.

To enhance the performance of ADMM on non-convex problems, the penalty parameter ρ is usually updated to make the Augmented Lagrangian function convex near the solution. Specifically, each region uses an individual penalty parameter ρ_k which is updated as follows [10]:

$$\tilde{\rho}_k^{\nu+1} = \begin{cases} \rho_k^{\nu_k} & \text{if } \; \Gamma_k^{\nu_k+1} \leq \xi \Gamma_k^{\nu_k} \\ \tau \rho_k^{\nu_k} & \text{otherwise} \end{cases} \tag{2.32a}$$

$$\rho_k^{\nu_k+1} = \max\{\tilde{\rho}_l^{\nu_k+1}, \forall l \in \{k\} \cup \mathcal{N}_k\}, \tag{2.32b}$$

with constants $0 < \xi < 1$ and $\tau > 1$, and with $\mathcal{N}_k$ denoting the set of neighbors that connect to region k. $\Gamma_k^{\nu+1} = \|A_k x_k^{\nu+1} - z_k^{\nu+1}\|_\infty$ is defined as the local primal residue [5], which measures the error in the coupling constraints and can be regarded as a measure for the feasibility of the solution. The update of penalty involves two steps. First, after local x and λ updates, the local penalty ρ_k should be increased if the local primal residue does not decrease sufficiently in the most recent local computation, as shown in (2.32a). Then, after receiving the penalty parameters from neighbors, the maximum penalty used by the neighbors and the region itself will be chosen to be used in the next local iteration to mimic the penalty increasing method used in centralized optimization [10], as shown in (2.32b).

A detailed procedure of the distributed ADMM algorithm is illustrated in Algorithm 1, where ν denotes the iteration counter. Notice that (2.32a) requires the exchange of $\tilde{\rho}_k$ and $\tilde{\rho}_l$

Algorithm 1 Distributed OPF in Region k with ADMM

1: **Initialize** x_k^0, $z_k^0 = 0$, $\lambda_k^0 = 0$, $\rho_k^0 = \rho_0$, $\nu = 0$

2: **while** Not converged **do**

3: $\nu \leftarrow \nu + 1$

4: Update x_k by solving the local OPF

$$x_k^\nu = \underset{x_k \in \mathcal{X}_k}{\operatorname{argmin}} \; f_k(x_k) + \lambda_k^{\nu-1\top}(A_k x_k - z_k^{\nu-1}) + \frac{1}{2}\|A_k x_k - z_k^{\nu-1}\|^2_{\rho_k^{\nu-1}}$$

5: Prepare messages $m_k^\nu = A_k x_k^\nu$

6: Send m_k^ν to regions $l \in \mathcal{N}_k$ and receive m_l^ν from regions $l \in \mathcal{N}_k$

7: Update z_k with respect to any tie line $ij, i \in \mathcal{R}_k, j \in \mathcal{V}_k \setminus \mathcal{R}_k$ using

$$z_{i,j}^{-\nu} = \frac{1}{2}(m_{k,i,j}^{-\nu} - m_{l,j,i}^{-\nu})$$
$$z_{i,j}^{+\nu} = \frac{1}{2}(m_{k,i,j}^{+\nu} + m_{l,j,i}^{+\nu})$$

8: Update λ_k using

$$\lambda_k^\nu = \lambda_k^{\nu-1} + \operatorname{diag}(\rho_k^{\nu-1})(A_k x_k^\nu - z_k^\nu)$$

9: Calculate the primal residue Γ_k^ν for each region k

10: Check convergence

11: Compute $\tilde{\rho}_k^\nu$ according to (2.32a)

12: Broadcast $\tilde{\rho}_k^\nu$ to regions $l \in \mathcal{N}_k$ and receive $\tilde{\rho}_l^\nu$ from regions $l \in \mathcal{N}_k$

13: Update ρ_k using (2.32b)

14: **end while**

between each pair of neighboring regions k and l. To reduce the communications cost, this exchange can be implemented during the information exchange before the z-update. This may lead to a slightly degraded performance of ADMM since the penalty parameters are not updated immediately, which, on the other hand, only requires one round of information exchange for each ADMM iteration.

A general way to check the convergence of (2.31) is to check whether the primal residue $(\Gamma_k, \forall k)$ is smaller than some ϵ [5]. However, in the AC OPF problem, power balance feasibility must also be ensured. This feasibility is checked after averaging the duplicate

voltages in each iteration. Convergence is declared when both the primal residue and the maximum bus power mismatch after voltage averaging fall below ϵ. The convergence of Algorithm 1 to a KKT stationary point $\{x^\star, z^\star, \lambda^\star\}$ is proved in [10] with the assumption that both x, λ and ρ are bounded and that a local minimum can be identified when solving the local subproblems. When ρ is unbounded, Algorithm 1 can only converge to a feasible point that is close to the local minimum for non-convex problems.

Some guidelines regarding the choice of parameters $\beta^+, \beta^-, \gamma, \tau$, and the initial value of ρ are provided in [10], which are summarized as follows:

- β^+ and β^- should be chosen in a way that $\beta^- > \beta^+ > 0$ to emphasize more on the power flow on the tie lines.
- For the update of penalty parameters ρ, ξ is usually set to a value slightly lower than 1, and τ is set to a value slightly larger than 1 to avoid rapid increase of the penalty parameter. The larger τ is, the faster ρ increases and consequently the faster ADMM converges, which, however, generally leads to solutions with worse quality since the algorithm proceeds more aggressively. Hence, there is a trade-off between the convergence speed of ADMM and the solution quality it achieves.
- For the initial value of ρ_0, it can be set to a value of the same or slightly larger magnitude as the initial objective function value if a good starting point is used; otherwise, it should be set to a much smaller value compared with the initial objective function to ensure convergence.

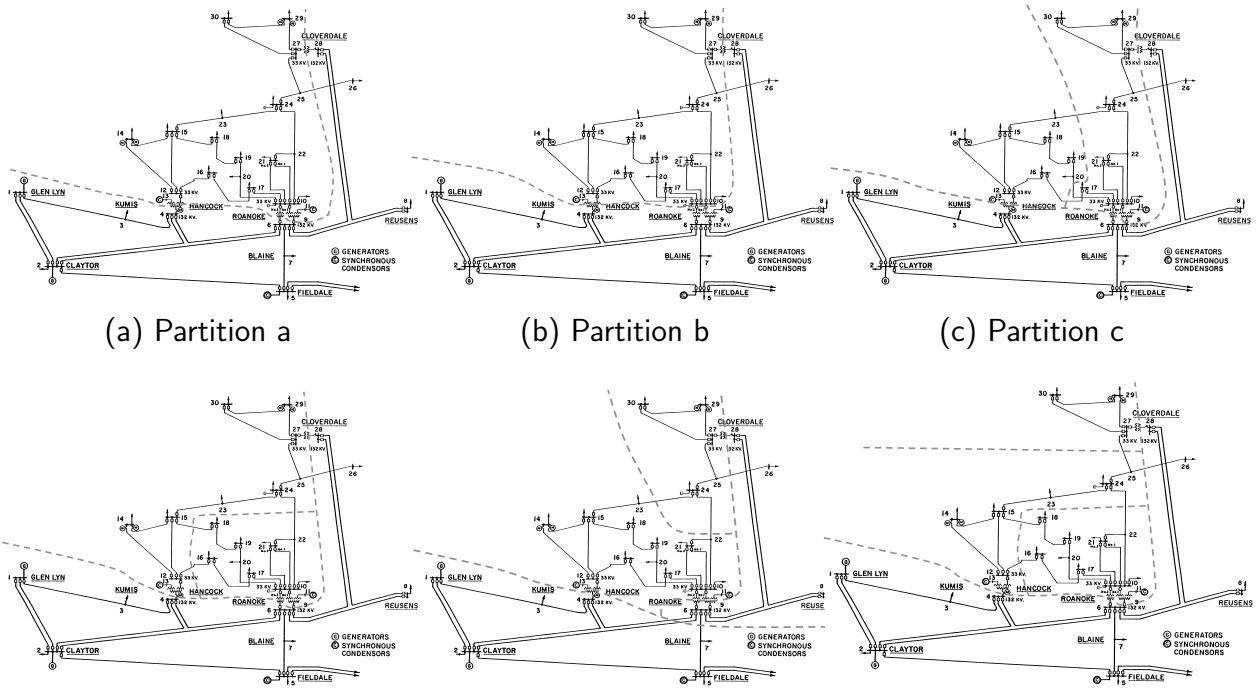
We should mention that the exact parameters used in ADMM are usually tuned through empirical studies. [80][64] provide some techniques on tuning ADMM parameters for convex optimization problems, while we will also provide guidelines on parameter tuning when we present numerical results of applying ADMM to various power systems.

Chapter 3

Power System Partitioning

This chapter first studies the impact of system partitioning on the convergence performance of distributed algorithms, and then proposes a partitioning technique based on spectral clustering. An affinity metric based on the Hessian matrix of the optimization problem is derived that measures the computational couplings among buses and therefore serves as the basis for partitioning. As a start, we develop this spectral partitioning technique based on the OCD method and show that using the proposed spectral partitioning method, both OCD and OCD-C can converge in significantly fewer iterations to a local optimum of the original problem. The applicability of spectral partitioning to ADMM will then be presented in Chapter 4.1.

The performance metrics used to measure the efficiency of OCD and OCD-C in this thesis include the number of iterations ν and the convergence time t. Note that t is an approximation of the convergence time assuming that the subproblems would be solved in parallel with no communications delays, as opposed to our actual implementation where we solve the problems in series. Specifically, if we use t_1^ν, t_2^ν, ..., t_K^ν to denote the time spent on local computation of iteration ν in each area, then $t = \sum_\nu \max\{t_1^\nu, t_2^\nu, ..., t_K^\nu\}$. The communication delay is omitted in the experiments in this chapter, and will be added and discussed in Chapter 5. The configurations of the test systems used in this thesis are all taken from MATPOWER [79].



(a) Partition a (b) Partition b (c) Partition c

(d) Partition d (e) Partition e (f) Partition f

Figure 3.1: Six partitions of the IEEE 30-bus system.

3.1 Impact of System Partitioning

To motivate the proposal of a system partitioning technique, we first investigate whether system partitioning has a significant impact on the performance of distributed optimization algorithms. To study this issue, we conduct an experiment on the IEEE 30-bus test system to show how different partitions of the system affect the convergence rate of OCD. The simulation is carried out in Matlab on an iMac with 8GB memory and 3.2 GHz Intel Core i5. Fig. 3.1 shows six different partitions of the IEEE 30-bus system into 2 to 4 regions and OCD is applied to each partition. Note that all buses are clustered by the dashed lines except for buses 25 and 26 which are in the same area in all partitions given the specific topology of the system. The same initial point and stopping criterion are used for all partitions as well as the centralized optimization. Initially, we set the voltage magnitudes to 1 p.u., the voltage angles to zero, and generations to the median of their upper and lower limits. This initialization will be referred to as a *flat start* in the following analysis. The stopping criterion is that the $L2$ norm of all constraints mismatch is smaller than 10^{-4}. It is guaranteed that OCD converges to the same result as the centralized optimization if it converges.

To characterize different partitions, we consider number of regions, number of tie lines, and number of coupling constraints to be possible metrics that represent the coupling among regions associated with a specific partition. Furthermore, recall that the convergence criterion for OCD is given by

$$c = \rho(I - \overline{H}^{*-1} H^*) < 1, \tag{3.1}$$

we also consider the spectral radius c as a potential metric for inter-region coupling as it is strongly related to the convergence of OCD.

Table 3.1 shows the convergence performance of OCD using the six partitions and the characteristics of these partitions. As a comparison, the centralized optimization using the Newton-Raphson approach is also carried out, which is equivalent to applying OCD but with only one region. From Table 3.1, the following key observations can be made:

1. For a fixed number of subproblems, different ways of partitioning the system significantly affect the convergence speed of OCD. For example, the number of iterations and convergence time approximately increases by 70% if using Partition b instead of Partition a, both of which partition the system into two areas.

2. The number of subproblems greatly impacts the computation efficiency of OCD. In general, as the number of subproblems increases, the number of iterations will increase while the time spent on each iteration will decrease due to the reduced subproblem size. Note that the time spent on each iteration depends on the size of the largest sub-

Table 3.1: Convergence of OCD using different partitions of the IEEE 30-bus system.

Partition	Regions	Tie lines	Coupling constraints	Spectral radius c	Iterations	Convergence time $t(s)$
None	1	0	0	N/A	30	0.09
a	2	4	14	0.6117	93	0.09
b	2	4	14	0.7647	158	0.15
c	3	7	24	0.8386	210	0.14
d	3	7	18	0.8550	251	0.25
e	4	9	24	0.8465	242	0.17
f	4	8	26	0.8893	321	0.21

problem. Hence, it is worth studying how to balance the coordination needed among subproblems associated with number of iterations and the size of each subproblem.

3. In general, the number of iterations and convergence time increase with the number of tie lines, the number of coupling constraints and the spectral radius c. However, in this example, the number of iterations only strictly increases with the increase in the spectral radius but not the other two metrics. Hence, we can conclude that c can be regarded as the most appropriate metric for measuring the computational coupling between partitioned areas. Bus i and j are strongly computationally coupled if they have influence on each other in many constraints or the objective function. The smaller c is, the less coupling exists between areas and generally the faster the convergence. If all areas are fully decoupled, i.e., no connections exist between any two areas, then c reduces to 0.

4. It is worth noticing that only by using Partition (a) can OCD show comparable computation efficiency compared with the centralized optimization, which shows that studying the partitioning problem can help evaluate whether a system can gain time benefit from performing distributed optimization.

To sum up, how a system is partitioned significantly affects the time efficiency of OCD. More importantly, only by using a good partition can OCD converge faster than or as fast as the centralized optimization. Even by using a toy example like the IEEE 30-bus system, one can see the significance of system partitioning, which will be further magnified in large systems.

Besides OCD, we also apply the OCD-C method with these six partitions and show that the partitioning of the system also impacts the convergence performance of OCD-C as well. The results are presented in Table 3.2 and Fig. 3.2. Another purpose of showing these results is to demonstrate that adding the correction term in OCD-C can effectively speed-up OCD. The same test cases shown in Fig. 3.1 are used. The same initial point and stopping criterion is used for both OCD and OCD-C in all partitioning cases. Table 3.2 shows the comparison of convergence speed between OCD-C and OCD, where ν and t are the number of iterations and convergence time respectively. Note that for OCD-C, t is the sum of the computation time of the basic Newton-Raphson step in OCD (t_{OCD}) and

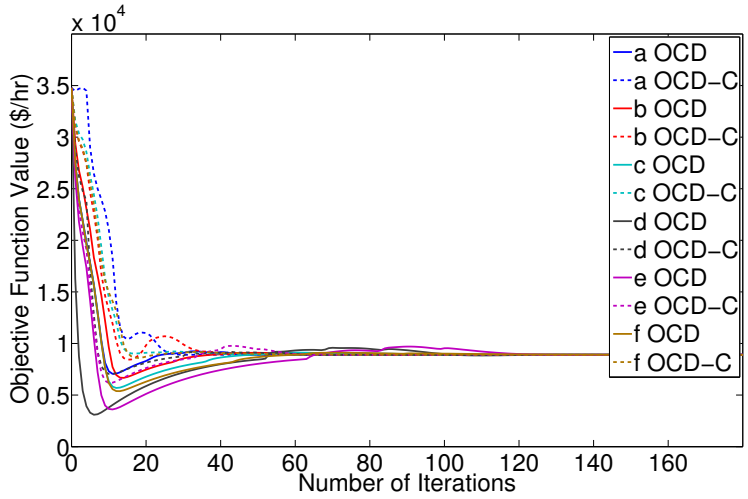
Table 3.2: Speed-up of OCD-C compared with OCD.

Partition	OCD		OCD-C				Speed-up	
	ν	t(s)	ν	t_{OCD}(s)	t_C(s)	t(s)	Iterations	Time
a	93	0.09	53	0.07	0.01	0.08	43%	11%
b	158	0.15	90	0.10	0.02	0.12	43%	20%
c	210	0.14	125	0.10	0.03	0.13	40%	7%
d	251	0.25	131	0.15	0.04	0.19	48%	24%
e	242	0.17	125	0.11	0.04	0.15	42%	12%
f	321	0.21	175	0.13	0.04	0.17	45%	19%

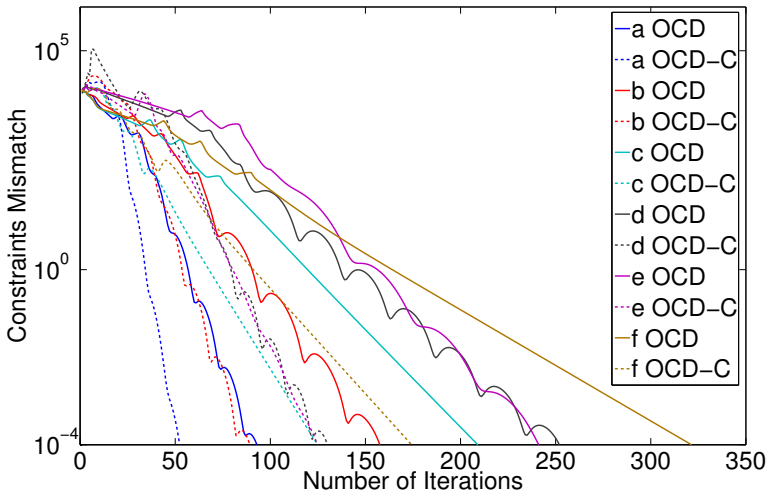
the time calculating the correction terms (t_C). The last two columns show the speed-up of OCD-C compared with OCD in terms of both number of iterations and convergence time.

As shown in Table 3.2, OCD-C reduces the number of iterations by around 45% in all test cases, which is significant. The speed-up in convergence time is less significant due to the additional time spent on calculating the correction terms. However, the speed-up in time is expected to be improved if the time spent on communications is considered because the reduced iterations lead to reduced time on information exchange which happens after each iteration.

Fig. 3.2 also confirms the effectiveness of OCD-C. The constraints mismatch in Fig. 3.2b denotes the 2-norm of mismatch in all constraints associated with the optimization problem. To show a clearer convergence trend, only the first 160 iterations are included in Fig. 3.2a. It is worth noticing that by using OCD-C, the oscillation of the value of the objective function is dampened around the optimal value, which shows the fact that each region can better revise its search direction for the overall optimality with more information acquired from neighboring areas. From Fig. 3.2, it is further validated that different system partitions can lead to very different performances of both OCD and OCD-C. Hence, a good partition of the system needs to be determined by a systematic approach before any distributed optimization process.



(a) Convergence of objective function



(b) Constraints mismatch over iterations

Figure 3.2: Convergence property of OCD and OCD-C.

3.2 The Coupling Parameter

Chapter 3.1 shows that the spectral radius c is an effective metric to measure the coupling among areas. A smaller c indicates less coupling between subproblems and therefore generally results in fewer iterations until convergence is reached. In the following analysis, we will refer to c as the coupling parameter. By choosing c as the coupling parameter, the definition of coupling refers to a computational coupling between the areas as opposed to a purely physical coupling. Hence, to reduce convergence time of OCD and OCD-C, one reasonable approach is to minimize this coupling parameter by intelligently choosing the partitions.

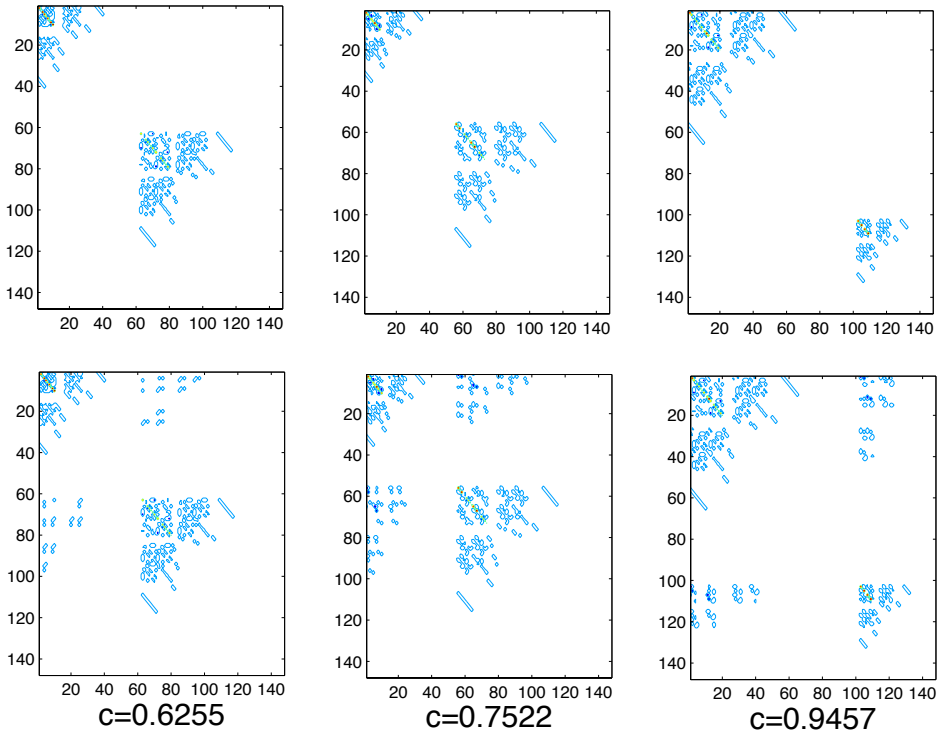


Figure 3.3: Contour of Hessian matrices of two-region partitions of the IEEE 14-bus system.

Several methods have been proposed for minimizing the spectral radius of a matrix [81][82]. However, these methods are not suitable for the considered distributed optimization problem because the targeted matrix defined in those methods is the affine matrix of a vector in which the elements can be changed to achieve the minimal spectral radius. In other words, it is required that the structure of the matrix is explicitly given beforehand, whereas in our problem the structure of the matrix depends on the chosen partition and therefore should be the result of the minimization process and not an input to it. Therefore, we need to define the relation between the coupling parameter and the structure of H^* and then construct H^* in such a way that the corresponding c is minimized. To that end, we consider the structures of H^* and $\overline{H}^*$ and the value of the corresponding coupling parameter. As an example, Fig. 3.3 shows the contour of $\overline{H}^*$ (first row) and H^* (second row) of three different two-region partitions of the IEEE 14-bus system where a contour plot displays the isolines of a matrix. The value of the coupling parameter associated with the partition is given below the contour plot of matrix H^*. As seen from Fig. 3.3, if the contour of H^* contains less off-diagonal

'noise' compared with $\overline{H}^*$, the coupling parameter is smaller. Hence, the off-diagonal blocks in $\overline{H}^*$ should contain as many zeros or entries with small absolute values as possible. This assumption can be proven using the following simple case:

Suppose a system is partitioned into two subproblems and has the following H^* and $\overline{H}^*$ matrices:

$$H^* = \begin{pmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{pmatrix}, \quad \overline{H}^* = \begin{pmatrix} H_{11} & 0 \\ 0 & H_{22} \end{pmatrix}, \tag{3.2}$$

with $H_{12} = H_{21}^T$. Let $M = I - \overline{H}^{*-1} H^*$, then

$$\begin{aligned} M &= I - \begin{pmatrix} H_{11}^{-1} & 0 \\ 0 & H_{22}^{-1} \end{pmatrix} \begin{pmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{pmatrix} \\ &= \begin{pmatrix} 0 & -H_{11}^{-1} H_{12} \\ -H_{22}^{-1} H_{21} & 0 \end{pmatrix}. \end{aligned} \tag{3.3}$$

Here, we introduce the theorem on bounds of eigenvalues from [83]:

Theorem 1 (Eigenvalues bound) *Let M be an $n \times n$ matrix, and let C_i, $i = 1, 2, ..., n$ be the discs with centers a_{ii} and radii $R_i = \sum_{k=1, k \neq i}^{n} |a_{ik}|$. Let D denote the union of the discs C_i. Then all the eigenvalues of M lie within D.*

The eigenvalue bound is the same as the spectral radius bound, since the spectral radius is the largest absolute value of all eigenvalues. From Theorem 1, the R_i of M depends on the absolute value of entries in $H_{11}^{-1} H_{12}$ and $H_{22}^{-1} H_{21}$ and therefore significantly on the values of the entries in H_{12} and H_{21}, i.e., on the computational coupling between region 1 and region 2. Obviously, if H_{12} has a higher degree of sparsity or more entries with small absolute values, then R_i tend to be smaller, generally resulting in a lower spectral radius of M.

Hence, to minimize the coupling parameter, one can reorder the rows and columns in H_{sys}^* evaluated at the optimal point where each row/column is associated with a bus so that the resulting H^* is as close to block diagonal as possible. Such matrix manipulation is equivalent to clustering the buses with large coupling to the same area. Hence, a method which achieves such clustering, provides a means to determine the partition which ensures superior computational performance of the OCD/OCD-C approach.

3.3 Spectral Partitioning

Based on the analysis above, we propose an intelligent partitioning method for providing a good partition of the system with the objective to optimize the convergence performance of the distributed optimization method. To fully determine the partition of the system, one needs to determine the number of subsystems and which bus should belong to which subsystem. In the proposed method, we fix the number of subsystems but find the optimal way to partition the system into this number of fixed areas.

3.3.1 Affinity Metric

The basic idea of the proposed partitioning method is to group the buses which are computationally strongly coupled into one area whereas weakly coupled buses should be assigned to different areas. This is based on the premise that the greater coupling among areas is, the greater is the mutual impact which eventually results in a longer convergence time of the distributed optimization. Based on the discussion in Chapter 3.2, we group the buses in a way that the associated H is as close to being block-diagonal as possible and the coupling parameter is minimized.

To achieve this goal, we use the spectral clustering method [84]. Spectral clustering [84] is a graph partitioning method which takes into account the affinity between any two elements in the target data set, which in our considered problem corresponds to the computational coupling between any two buses. Compared with conventional graph partitioning methods, spectral clustering is simpler to implement and forms tighter clusters before clustering so that the inaccuracy in clustering non-convex regions can be avoided [84]. However, as spectral clustering deploys the K-means clustering algorithm as the last step, it might provide more than one solution if one runs the K-means algorithm in multiple trials with different initializations. In our case, this means multiple "good" partitions are obtained for the given test system and number of areas. Consequently, the optimal partition is then chosen out of this set of "good" partitions by selecting the partition with the minimum coupling parameter.

Since the clustering is performed on the so-called affinity matrix, the key to devising a

good partitioning method is to define an affinity matrix which quantifies the computational coupling between the buses as accurately as possible. The choice of the definition of the affinity matrix is crucial as different affinity matrices lead to different measures to quantify such coupling and will therefore also result in different optimal partitions. Here, we define the affinity matrix based on the Hessian matrix H^*_{sys} associated with the considered AC OPF problem. We take advantage of the fact that if the entry $H_{i,j}$ in H^*_{sys} which denotes the second derivative of the Lagrange function with respect to the two variables with indices i and j is non-zero, then these two variables are coupled and the larger the absolute value of $H_{i,j}$ is, generally the stronger is the coupling.

Note that H^*_{sys} needs to be calculated for a particular operating point, which requires that the AC OPF problem needs to be solved first. However, if the optimal partition of the system once found does not depend much on the operating point, then the solving of the AC OPF problem is only a one-time effort. The entries in H^*_{sys} are functions mainly of the line admittance, the voltage magnitude at each bus, the *sin* and *cos* of the differences between two bus angles and the Lagrange multipliers for active and reactive power balances. Since the voltage magnitudes are all around 1 p.u. and the terms including bus angles are either around 0 or 1, what changes most with the operating point are the entries which are functions of the Lagrange multipliers, especially the multipliers for reactive power balances. Hence, it is expected that the entries in H^*_{sys} do not depend that significantly on the operating point. In Chapter 3.4.2, we will further discuss this problem related to the operating point and confirm through simulations that the optimal partition of the system once found is not sensitive to the operating point of the system.

Hence, to account for the computational coupling denoted by the entries in H^*_{sys} and capture the electrical characteristics of the system, the pairwise affinity metric $S_{i,j}$ is calculated by first summing up all the entries in H^*_{sys} associated with bus i and j then adding the admittance matrix with a certain weight. Particularly, if we use Ψ_i and Ψ_j to denote the sets of the indices of the variables associated with buses i and j, respectively, which include the voltage magnitudes, voltage angles, generations connected to the buses, Lagrange multipliers of the power balances at the buses and slack variables introduced to deal with

the inequality constraints, the $S_{i,j}$ is then calculated by

$$S_{i,j} = (1-w) \cdot \sum_{m \in \Psi_i} \sum_{n \in \Psi_j} |H^*_{sys\ m,n}| + w \cdot Y_{i,j}, \tag{3.4}$$

where $Y_{i,j}$ is the element in the admittance matrix of the system and $0 < w < 1$ is the weight used to weigh it in the affinity matrix. This parameter w is set to 0.5 in the simulations. Note that while the definition of the affinity matrix proposed here works well empirically, it is not the only way to define the affinity matrix. In fact, for the considered OPF problem, constructing the affinity matrix by using only the admittance matrix Y can also yield good partitions due to the fact that the entries in Y are close to many entries in H^*_{sys}. How to construct the optimal affinity matrix is an open question in spectral clustering. However, we recommend using H^*_{sys} or some estimates of H^*_{sys} for distributed optimization which captures the computational coupling between the variables.

3.3.2 Spectral Clustering

After the affinity matrix is derived, spectral clustering is applied. Spectral clustering first represents the original data points using eigenvectors of the Laplacian associated with the affinity matrix and then sorts the data points into different clusters employing the K-means algorithm, which is a popular clustering technique. The reasons that spectral clustering is used here but not K-means are as follows: 1) spectral clustering can be applied as long as an affinity matrix is derived while K-means requires explicit representation of each data point (such as its coordinates), which is not available in our considered application; 2) spectral clustering is more capable of identifying non-convex clusters. A detailed procedure of spectral clustering is described in [84] and can be applied to cluster a given set of n_b buses into K regions according to the following procedure:

1. Derive the elements in the affinity matrix according to (3.4) if $i \neq j$, and set the entries on the diagonal $S_{i,i} = 0$.

2. Define diagonal matrix D where $D_{i,i} = \sum_{n=1}^{n_b} S_{i,n}$ and construct the matrix $P = D^{-1/2} A D^{-1/2}$.

3. Find the K largest eigenvectors of P and form the matrix V by stacking the eigenvectors in column. Then renormalize each row of the generated matrix V.

4. Treat each row in V as a data point and cluster these data points into K clusters using the K-means algorithm.

5. Assign bus i to cluster k if and only if row i of V was assigned to cluster k.

3.3.3 Implementation

The implementation of the proposed spectral partitioning method to decompose the AC OPF problem includes four steps:

Step 1: Initialization. The objective of this step is to compute H^*_{sys} and define the number of regions K. H^*_{sys} is computed by first obtaining the optimal solution for a particular operating point by solving the AC OPF problem and then taking the second derivatives of the Lagrangian function (2.12).

Step 2: Define the affinity matrix S. S is calculated according to (3.4). S is a $n_b \times n_b$ matrix where the indices of the entries corresponds to the indices of the buses in the system.

Step 3: Clustering. Cluster the buses into K areas using spectral clustering based on the affinity matrix S in N trials. The centroids of the K-means method which are the initial center of the clusters are randomly chosen for each trial. Due to the random centroids, more than one possible clustering solution can be found which compose a reduced search space for the optimal solution.

Step 4: Selection. Compute the coupling parameter of the clustering solutions found in Step 3. The solution corresponding to the minimum coupling parameter is selected to be the optimal partitioning of the system.

These implementation steps are summarized in Algorithm 2. Finally, we would like to discuss three issues regarding the implementation of the proposed spectral partitioning approach. First, the partition of the system only affects the assignment of variables into subproblems in the computation, but does not affect the *physical structure of the power system*. Second, it is possible that the buses in the same partition obtained by the partitioning method are not physically in one connected graph. However, only a very small percentage

Algorithm 2 Spectral Partitioning

1: **Input** system configuration and number of regions K
2: Derive the Hessian matrix H^*_{sys} based on empirical information of the entire system
3: Derive the affinity matrix S according to (3.4)
4: **for** n=1 to N **do**
5: Perform spectral clustering based on the affinity matrix S to cluster the buses into K regions with the K-means method as the last step of spectral clustering
6: **end for**
7: Compute the coupling parameter c of all solutions
8: **Output** the partitioning solution with the minimum c which represents the assignment of each bus to the K regions

(around 2 %) of the total number of regions are disconnected. Moreover, the subproblem is still well-defined even if the region is not fully connected. Hence, the convergence of ADMM is not affected with those partitions including disconnected regions. Third, the choice of number of trials N in the K-means step should be chosen large enough such that it is highly unlikely to find new partitions by doing more trials than N. N generally should increase with the number of regions K or the system size. However, for large-scale complex systems with thousands of buses, one can limit N to several hundreds because it will be hard to enumerate all possible partitions using spectral partitioning and a set of hundreds of partitions make a large candidate set for choosing a good partition. Note that as spectral clustering is a heuristic method, it does not guarantee the solution found is the optimal partition of the system. However, the actual partitioning of a power system will also be restrained by other factors in practice, and therefore, the proposed heuristic approach is helpful as it can provide a set of good partitions that one can choose from, as we will show in the numerical results.

3.4 Simulations and Discussion

We now present simulation results to provide a proof of concept of the proposed spectral partitioning method. The proposed partitioning method is tested on the IEEE test systems

with 14, 30, 57 and 118 and 300 buses and a 1416-bus system consisting of 12 IEEE 118-bus systems to evaluate its effectiveness. The simulations are run in Matlab on an iMac. The same flat start and stopping criterion are used for the distributed optimization for the different partitions and the centralized optimization. The system configurations are acquired from MATPOWER [79]. For the number of areas, we choose 2 for the IEEE-14 and IEEE-30 bus test systems, 2 and 3 for the IEEE-57 bus system and 2 to 5 for the IEEE-118 bus test system. For all test cases, we apply the spectral partitioning method to obtain the best partition. For comparison purposes, arbitrary partitions are chosen by either simply searching for a natural geographical partition of the system or using the predefined partition of the system (e.g., IEEE 300-bus system). Both OCD and OCD-C as well as the centralized optimization approach with Newton-Raphson iterations are applied to the all partitions of the systems and computation times until convergence are compared. Each experiment is run five times and the average convergence time is recorded as the result for this experiment. The convergence criterion is chosen the same for all cases and is defined as the L_2 norm of all constraints mismatch being smaller than 10^{-4}. For *Step 3* in Chapter 3.3.3, the number of trials in the clustering method N is set to 100. Moreover, we have confirmed that all test cases for a particular test system converge to the same solution; i.e., distributed and centralized solutions are equal. The line limit constraint (2.1g) is first omitted, then added in Chapter 3.4.2 where scenarios with line congestion are particularly discussed.

The performance of the partitioning method is reflected by the convergence speed of the applied decomposition methods. The convergence speed of OCD and OCD-C is measured by the number of iterations ν and the convergence time t which is an approximation of the time spent on solving the subproblems in parallel. The smaller ν and t are, the better the partitioning.

3.4.1 Effectiveness of Spectral Partitioning

The performance of OCD and OCD-C with spectral partitioning and arbitrary partitioning are shown in Table 3.3. The first column denotes the test system and number of regions that the system is partitioned into; e.g., '14-2' denotes that the IEEE 14-bus test system is partitioned into 2 regionss. The speed-up of OCD and OCD-C is the speed-up in convergence

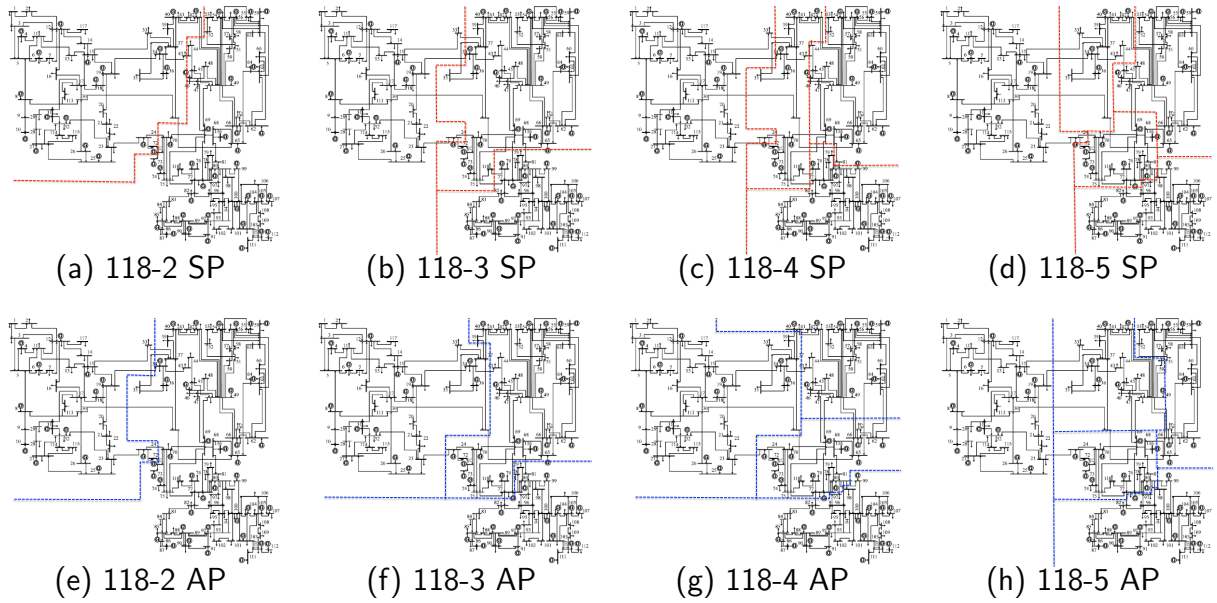


Figure 3.4: Spectral partitioning and arbitrary partitioning of the IEEE-118 system.

time compared with the centralized optimization. The partitions of the IEEE-118 system are presented in the form of diagrams in Fig. 3.4, while part of other partitions are given in dense form in Table 3.4. In the figures and tables presented, we use 'SP' and 'AP' to denote spectral partitioning and arbitrary partitioning, respectively. For the IEEE 57-bus and 118-bus system, we also experimented with more than one arbitrary partitions and present the best arbitrary partition in the tables for comparison with the spectral partitioning technique. For the 1416-bus system, a 12-region optimal partition is found which is identical to the original 12 118-bus systems. This is due to the fact that this large system is configured in a way that each pair of neighboring 118-bus systems are only connected by two tie lines, which naturally leads to little computational couplings between neighboring systems. In other words, the proposed partitioning method can successfully identify those weak inter-area couplings and partition the system accordingly.

As shown in Table 3.3, by using the proposed spectral partitioning approach, both OCD and OCD-C converge faster in all test cases compared with using arbitrary partitions. Interestingly, in Case 14-2 and 30-2, only by using spectral partitioning and OCD-C can distributed optimization converge faster than the centralized approach, which not only confirms the effectiveness but also shows the necessity of the partitioning method. However, in Case 57-2 and Case 57-3, we see that even by using spectral partitioning one can gain no

Table 3.3: Comparison of the convergence speed of OCD and OCD-C using spectral partitioning and arbitrary partitioning.

Case	Partition	Coupling	Centralized		OCD			OCD-C		
		Parameter	ν	$t(s)$	ν	$t(s)$	Speed-up	ν	$t(s)$	Speed-up
14-2	SP	0.6255	29	0.028	82	0.036	-29%	43	0.026	7%
14-2	AP1	0.7857	29	0.028	222	0.067	-139%	126	0.058	-107%
14-2	AP2	0.7522	29	0.028	173	0.082	-193%	100	0.066	-136%
30-2	SP	0.6117	30	0.09	93	0.09	0%	53	0.08	11%
30-2	AP1	0.7632	30	0.09	158	0.15	-67%	90	0.12	-33%
30-2	AP2	0.8566	30	0.09	282	0.38	-322%	174	0.31	-244%
57-2	SP	0.6850	30	0.3	93	0.85	-183%	58	0.58	-93%
57-2	AP	0.8970	30	0.3	307	1.35	-350%	204	1.08	-260%
57-3	SP	0.8334	30	0.3	214	0.70	-133%	126	0.62	-107%
57-3	AP	0.8903	30	0.3	315	0.88	-193%	213	0.80	-167%
118-2	SP	0.5124	33	3.8	105	3.4	11%	71	2.7	29%
118-2	AP	0.7261	33	3.8	197	8.9	-134%	116	5.9	-55%
118-3	SP	0.7948	33	3.8	225	2.3	39%	137	1.5	61%
118-3	AP	0.9107	33	3.8	454	4.3	-13%	270	3.5	8%
118-4	SP	0.8044	33	3.8	246	1.4	63%	128	1.2	68%
118-4	AP	0.8935	33	3.8	378	3.0	21%	231	2.8	26%
118-5	SP	0.8217	33	3.8	250	1.3	66%	132	1.2	68%
118-5	AP	0.8745	33	3.8	330	2.1	45%	195	1.9	50%
300-3	SP	0.9068	40	13.6	341	11.6	15%	186	6.8	50%
300-3	AP	0.9645	40	13.6	806	68.1	-401%	535	35.7	-163%
1416-12	SP	0.9442	37	625	608	30.4	95%	284	17.8	97%

efficiency benefit from the distributed approaches which is a valuable result that shows that the IEEE-57 bus test system is not suitable to be partitioned into subsystems for deploying OCD or OCD-C given its specific system topology. Hence, the proposed partitioning method can help to evaluate whether a system is suitable for distributed optimization.

As for larger systems, the distributed optimization methods show improved efficiency compared with smaller test systems which is generally the case with all distributed ap-

Table 3.4: Spectral partitioning and arbitrary partitioning of IEEE 14-bus, 30-bus and 57-bus systems.

Case	Partition	Regions (shorthanded by 'R')
14-2	SP	R1:1-5; R2:6-14
14-2	AP1	R1:1,5,6,10-14; R2:2-4,7-9
14-2	AP2	R1:1,5,6,12,13; R2:2-4,7-11,14
30-2	SP	R1:1-8,28; R2:9-27,29,30
30-2	AP1	R1:1-9,11,28; R2:10,12-27,29,30
30-2	AP2	R1:1,3,4,12-15,18,23; R2:2,5-11,16,17,19-22,24-30
57-2	SP	R1: 1-24,26-29,34-57; R2: 25,30-33
57-2	AP	R1: 1-20,27-29,41,43-55; R2: 21-26,30-40,42,56,57
57-3	SP	R1: 1-20,26-29,41,43,52-55; R2: 21-26,34-40,42,44-51,56,57; R3: 25,30-33
57-3	AP	R1: 1-20,27,29,43,52-55; R2: 21-26,28,34-42,44-51,56,57; R3: 25,30-33

proaches. Again, the speed-up can be significantly improved by using spectral partitioning. For example, in cases 118-4 and 118-5, both OCD and OCD-C can improve the centralized convergence time by over 60% if spectral partitioning is used whereas the speed-up can be insignificant if one uses random partitioning. For the IEEE-300 system, its predefined partition documented in MATPOWER is not suitable for implementing the distributed methods while our partitioning method finds a partition for which distributed optimization yields an improved performance over the centralized case. For the 1416-bus system as well, the speed-up of our spectral partitioning method is 95% for OCD and 97% for OCD-C which is quite impressive. The centralized method is extremely time-consuming for this large system due to the inversion of a matrix of the size of tens of thousands of elements. OCD and OCD-C can significantly alleviate such computation burden. Furthermore, OCD-C is superior to the original OCD approach in terms of both number of iterations and actual convergence time, especially when OCD takes many iterations to converge. Note that OCD and OCD-C would always take more iterations than the centralized method due to the fact that only partial information of the system is available at each subproblem and frequent information exchange needs to be made to achieve the overall optimality.

In addition, even with similar partitions such as the spectral and arbitrary partitions

in cases 57-3, 118-2 and 118-3, the resulting convergence speeds of distributed approaches vary substantially, which shows that mere geographical partitioning may lead to suboptimal solutions and a rigorous method as the proposed partitioning method is needed to determine a more suitable partition for implementation of distributed algorithms.

3.4.2 Operating Points

As introduced in previous chapters, the first step of the spectral partitioning is to establish an operating point of the system and solve the associated AC OPF problem. However, as the operating point corresponds to a specific load level, the entries in the affinity matrix change as the load level changes. Two questions that one can ask are the following: 1) Can one find the best partition at any operating point? 2) Does the best partition need to be changed when the load level changes? The latter is particularly important for the case including line constraints as increasing load levels may result in congestions in the system.

To answer the above questions, we incorporate the line constraints into the derivation of the affinity matrix and then apply the proposed partitioning method to find the best partition for different operating points with such constraints in place. In this discussion, the IEEE 118-bus test system is used. Specifically, at each operating point, the AC OPF problem with line constraints is solved first and the Hessian matrix associated with that operating point is calculated. Then the proposed partitioning method is applied to find the best 2-region partitions of the system for each operating point. The load level ranges from 0.5 to 1.7 of the base load level given by the original system specifications. For example, load level 0.5 denotes that all the loads are reduced to 0.5 of their base values. The line limits are chosen such that different lines may become congested at the optimal solution at different operating points. The congested lines are shown in Table 3.5 where a line is denoted by the two buses it connects.

Table 3.6 shows the best partitions found by the proposed partitioning method (denoted by SP) for each operating point. It can been seen that different partitions might be found depending on the operating point. This is partially due to the fact that lines become congested, especially when the tie lines associated with the original partition, i.e., the best partition without congestions, become congested. This is because when a line becomes congested,

Table 3.5: Congested lines at different operating points.

Load Level	Congested Lines	Load Level	Congested Lines
0.5	None	1.2	4-5
0.6	None	1.3	88-89
0.7	42-49	1.4	88-89,68-116
0.8	42-49, 38-65	1.5	15-33,43-44,88-89,89-92,68-116
0.9	38-65, 100-103	1.6	5-8,15-33,43-44,88-89,89-92
1.0	38-65	1.7	5-8,15-33,43-44,64-65,88-89,89-92
1.1	None		

Table 3.6: Best partitions for different operating points using spectral partitioning.

Partition	Load Levels	Regions
SP1	0.5,0.6,0.9-1.7	R1: 1-43,72,113-115,117; R2: 44-71,73-112,116,118
SP2	0.7	R1: 1-41,43,72,113-115,117; R2: 42,44-71,73-112,116,118
SP3	0.8	R1: 1-68,72,113-117; R2: 69-71,73-112,118

the Lagrange multiplier associated with the limit on that line will become non-zero which leads to changes in the entries of H_{sys} and the affinity matrix. Such changes might lead to a different partitioning solution because the partitioning is performed based on the affinity matrix.

To evaluate the performance of the optimal partitions listed in Table 3.6, the coupling parameters associated with the best partitions and three other arbitrarily chosen partitions (AP) are plotted in Fig. 3.5. The smaller the coupling parameter is, the better the partition is. It can be seen that the partition found by our method at each load level performs better than arbitrary partitions, which demonstrates the effectiveness of the partitioning method with line constraints considered.

Furthermore, as shown in Table 3.6, SP1 serves as the optimal partition for a fairly large range of load levels even when different lines become congested. Note that SP1 is also the best partition determined at the base level without line constraints considered. To further evaluate how SP1 performs at load levels 0.7 and 0.8 where the partitioning method

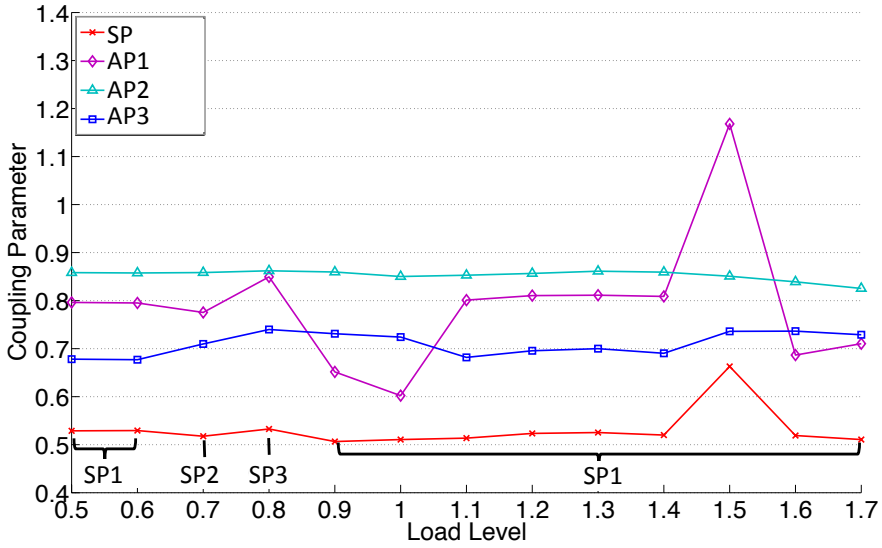


Figure 3.5: Performance of spectral and arbitrary partitioning of the IEEE 118-bus system with line congestion.

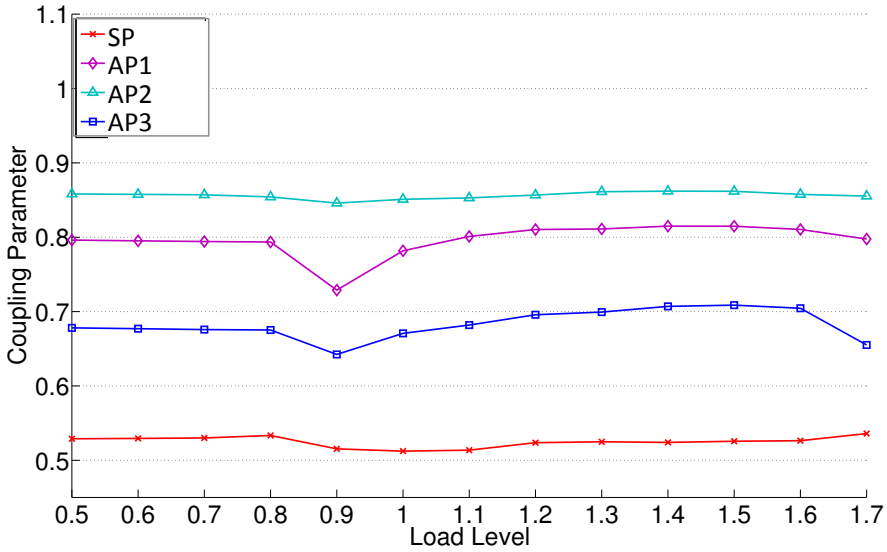


Figure 3.6: Performance of spectral and arbitrary partitioning of the IEEE 118-bus system without line congestion.

finds a different partition other than SP1, the coupling parameters of SP1, SP2 and SP3 are compared in Table 3.7. The coupling parameter of SP1 is very close to the coupling parameters of SP2 and SP3, which indicates that SP1 is also a good partition at such load levels.

For comparison, we also apply the proposed partitioning method to different operating

Table 3.7: Coupling parameters of best partitions at different load levels.

Load Level: 0.7	SP2: c=0.5177	SP1: c=0.5478
Load Level: 0.8	SP3: c=0.5327	SP1: c=0.5505

points with the line limits set to infinity where there is no congestion occurring in the system. For all load levels from 0.5 to 1.7, the same optimal partition SP1 is found. Figure 3.6 shows the coupling parameters associated with SP1 and the same three arbitrary partitions used in Fig. 3.5. It is clear that the coupling parameter of SP1 remains stable for all operating points, which shows that SP1 is the best partition for all operating points.

In summary, the proposed partitioning method can effectively determine the best partition for different operating points with or without line congestions considered. In addition, the best partition determined without line constraints is either the best partition or very close to the best partition for a large range of load levels, even when lines within the areas become congested. Hence, it is fair to say that the partition does not need to be changed unless the power flows suffer from drastic changes caused by congested tie lines. In such cases, one can apply the partitioning method to find good partitioning solutions for each line congestion scenario. Then a look-up table similar to Table 3.6 can be generated to provide guidelines on which partition to use when different line congestions occur.

Chapter 4

Applications of Spectral Partitioning

In Chapter 3, we proposed a partitioning method which defines an affinity metric that captures the coupling among the regions and maps the power system partitioning problem to a graph partitioning problem. The partitioning method was derived from OCD and tested for this decomposition method on the traditional IEEE benchmark systems. Yet, it remains to be seen whether this partitioning technique can devise good partitions of real-world large-scale systems for an efficient implementation of other distributed methods beyond OCD/OCD-C. Moreover, the best partition of the system was computed and evaluated for a single time step, and it is not clear whether the same partition can be used for solving an optimization problem spanning multiple time steps. Therefore, to address these two concerns, in this chapter, we first apply this partitioning technique in conjunction with the ADMM for large-scale OPF problems and then solve a multi-step OPF problem with wind and storage. The intention is to demonstrate the general applicability and scalability of the proposed partitioning approach.

4.1 Large-Scale Optimal Power Flow

4.1.1 Applicability of Spectral Partitioning to ADMM

From our preliminary studies, we have found our that OCD or OCD-C generally has difficulty to converge while applied to large-scale transmission networks such as the Polish network due to the large coupling among regions that results in the violation of its convergence

criterion even with a good partitioning of the system. Therefore, for large-scale networks, we choose to use ADMM as it has been shown in literature that ADMM can deal with large-scale optimization problems in many fields, such as machine learning, networking, and power systems. It is presented in Chapter 2.4.2 how to solve the OPF problem (2.1) using ADMM in a distributed manner.

To apply the proposed spectral partitioning technique with ADMM, we follow the same procedure as Algorithm 2. A difference here from applying the partitioning technique with OCD lies in how to choose the best partition out of a set of good partitions. For ADMM, the most balanced partition, i.e., the one with the smallest largest region is chosen to be the best partition of the system. The reason for this choice is that the subproblems need to be solved in parallel before each information exchange occurs, thus a balanced partition is preferable for reducing the maximum subproblem solution time in each ADMM iteration. This criterion for choosing the best partition is different from the spectral radius criterion used with OCD. While the convergence of OCD is directly dependent on the spectral radius, the convergence of ADMM is more affected by how long the slowest region solves its subproblem especially in large-scale systems, which will be demonstrated in the following experiments.

The affinity matrix used is calculated using (3.4), same as that used for OCD. Here, we will show that this choice of affinity matrix based on the Hessian matrix is highly related to the evaluation of the first-order optimality conditions using the distributed ADMM approach, and therefore works well with ADMM. The OPF problem (2.1) considered in this paper can be written in the following general form:

$$\underset{x\in\mathbb{R}^p}{\text{minimize}} \quad f(x) \tag{4.1a}$$

$$\text{subject to} \quad g(x) = 0 \tag{4.1b}$$

$$h(x) \leq 0, \tag{4.1c}$$

where $g : \mathbb{R}^p \to \mathbb{R}^{q_1}$ and $h : \mathbb{R}^p \to \mathbb{R}^{q_2}$ are continuously differentiable functions. We say that $x^* \in \mathbb{R}^p$, $\mu^* \in \mathbb{R}^{q_1}$, and $\eta^* \in \mathbb{R}^{q_2}$ satisfy the first-order optimality conditions of problem (4.1) if the following hold:

1. Lagrangian vanishes:
   $\nabla f(x^*) + \mu^{*\top} \nabla g(x^*) + \eta^{*\top} \nabla h(x^*) = 0.$

2. Primal feasibility: $g(x^*) = 0$ and $h(x^*) \leq 0$.

3. Dual feasibility: $\eta^* \geq 0$.

4. Complementary slackness: $\eta_i^* h_i(x^*) = 0, i = 1, ..., q_2$.

If problem (4.1) contains no inequality constraints, then the optimality conditions can be written in the compact form of $F(\phi^c) = 0$ where $\phi^c = (x^*, \mu^*) \in \mathbb{R}^{p+q_1}$ denotes the centralized solution to problem (4.1). Let H denote the Jacobian of $F(\phi)$, which is also the Hessian matrix of the Lagrangian function evaluated at the optimal solution ϕ^c with respect to both the primal and the dual variables. Note that H here is the same as H_{sys}^* used in Chapter 3.3.1. If problem (4.1) contains inequality constraints, a slack variable s is introduced to transform the inequalities into $h(x^*) + s = 0, s \geq 0$. We then consider the optimality conditions of the vanishing Lagrangian, $g(x^*) = 0$, and $h(x^*) + s = 0$ and rewrite them in the compact form of $F(\phi^c) = 0$ where $\phi^c = (x^*, \mu^*, \eta^*) \in \mathbb{R}^{p+q_1+q_2}$, and H is again defined as the Jacobian of $F(\phi)$ evaluated at ϕ^c. The reason for this choice of $F(\phi^c) = 0$ is that the Jacobian of $F(\phi^c)$ in terms of ϕ^c will be the same as the Hessian of the Lagrangian function of problem (4.1), as the readers may easily check. For the simplicity of presentation, we consider problem (4.1) with only equality constraints in the following analysis.

Now we introduce ϕ^d to denote the counterparts to the centralized solution ϕ^c obtained by solving the local OPF problems in ADMM. Note that ϕ^d does not include any auxiliary variable z and to make ϕ^d the same dimension as ϕ^c, the average of the two copies of a duplicated variable is taken to be the resulting solution of that variable. As ϕ^d is generally different from ϕ^c, we would like to evaluate how well the distributed solution ϕ^d satisfies the centralized optimality conditions. An important observation here is that the optimality conditions associated with the non-boundary buses derived for the local OPF problems are identical to those associated with the non-boundary buses derived for the centralized OPF problem, which is shown explicitly in Appendix B. Therefore, these optimality conditions can be satisfied under the assumption that a local minimum can be found for the local OPF problems. Furthermore, it can be easily checked that the primal feasibilities (2.1c)-(2.1f) at all buses are satisfied by solving local OPF problems. Hence, it only remains to be evaluated how optimality conditions apart from (2.1c)-(2.1f) associated with the boundary buses are

satisfied by ϕ^d. Denoting this part of optimality conditions by $F^b(\phi^d) = 0$ and assuming that ν is large enough such that ϕ^d is in the vicinity of ϕ^c, then $F^b(\phi^d)$ can be approximated by $F^b(\phi^d) \approx H^b(\phi^d - \phi^c)$. Here, H^b is constructed by taking all the rows in H that contain any non-zero entry $H_{m,n}$ where its associated two variables ϕ_m and ϕ_n belong to different subproblems. It is expected that if the entries in H^b are generally small in terms of their absolute values, then ϕ^d can better satisfy the centralized optimality conditions, which, on the other hand, shows that ADMM can converge easier to an optimal point with possibly fewer iterations.

Hence, we aim to find a partition such that H^b contains mostly small values. Since minimizing the entries in H^b exactly is generally hard, a heuristic way is to consider $|H_{m,n}|$ as an affinity measure between variables ϕ_m and ϕ_n in spectral partitioning. If $|H_{m,n}|$ is large, ϕ_m and ϕ_n are more likely to be grouped into one subproblem, hence $H_{m,n}$ is less likely to appear in H^b based on the construction of H^b. Therefore, the derivation presented in (3.4) is also a viable way for constructing the affinity matrix for ADMM.

As most distributed methods solve for the optimality conditions in either a direct (such as OCD) or indirect (such as ADMM) way, one can follow similar analysis on the optimality conditions to define an affinity metric that works effectively with a specific distributed algorithm. In fact, we have used the same affinity matrix for OCD and ADMM. The derivation above provides a more general justification for the usefulness of this affinity matrix also for other distributed approaches which is important for the generalization of the partitioning technique.

4.1.2 Simulation Results and Discussion

To demonstrate that the proposed spectral partitioning technique is essential for applying ADMM to large-scale OPF problems, three sets of simulation results are presented. First, the partitions computed by the spectral partitioning method are compared with other partitions used in the literature in terms of the resulting performance of ADMM. Then, the impact of the number of regions on ADMM is presented. Finally, the performance of ADMM is evaluated after line flow limits are added to the OPF problem.

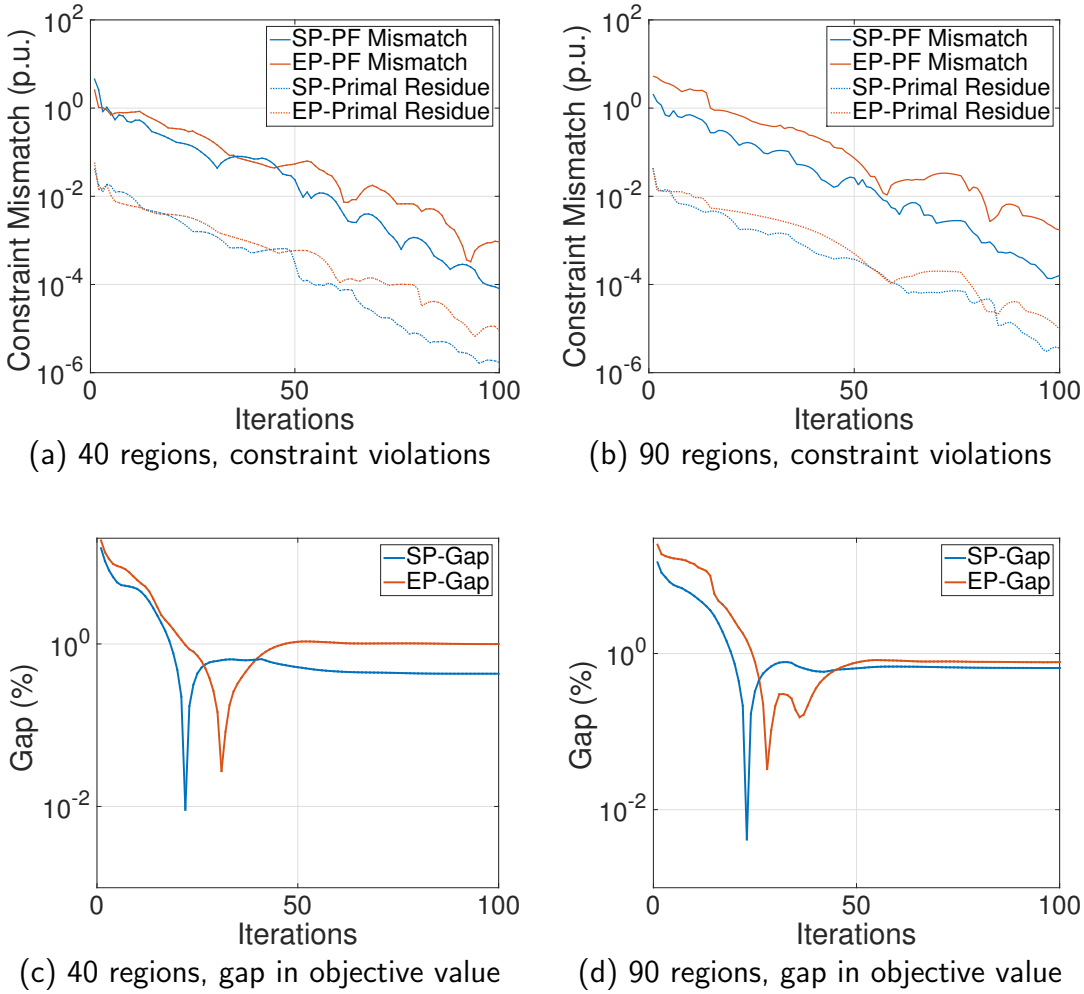
The test case used is the Polish 2383-bus system during the winter period, which contains

327 generators and 2896 transmission lines, and its configuration is taken from MATPOWER [85]. The SNOPT package in TOMLAB [86], which finds local optimums for large-scale non-convex optimization problems, is used for solving the local OPF subproblems at each ADMM iteration. For comparison, the centralized AC OPF problem is also solved using the SNOPT solver. The simulations are run on a MacBook Air in Matlab.

The algorithm is started from two different initial conditions, and we refer to these cases as the warm start case and the flat start case. In the warm start case, the starting point is a feasible power flow solution provided by MATPOWER. The warm start is a common choice in practice to solve large-scale OPF problems. In the flat start case, the voltages at all buses are initially set to 1 p.u., from which convergence is typically hard to achieve. For both cases, the initial Lagrange multiplier estimates are set to zero. The initial penalty parameter ρ_0 and its incremental step size τ are set to 10^7 and 1.1, respectively, in the warm start case to achieve fast convergence, while in the flat start case, they are set to lower values. Parameters γ, β^+, and β^- are set to 0.9, 0.5, and 2, respectively. Convergence is declared when the largest primal residue $\|A_k x_k - z_k\|_\infty$ falls below 10^{-4} and the largest bus power mismatch falls below 10^{-4} p.u.

Evaluation of Partitions

In this section, the impact of different partitions on the performance of ADMM is evaluated. Line limits are not included in the simulations in this section, but will be added later on. The partition obtained using spectral partitioning is compared with the partition used in [10], which is based on the electrical distance between buses. Specifically, in [10], the partition is chosen in a way such that each load bus is assigned to the same region as its closest generator bus in terms of line impedance. If a total of K regions is desired, K generator buses are selected uniformly as the centers of each region. The distance between any two buses is then defined as the length of the shortest path in terms of line impedances. This partition considers the topology and electrical properties of the system, and hence it is a good benchmark for evaluating the performance of the spectral partitioning technique. This partitioning method based on electrical distance is denoted as EP in the following analysis, while the spectral partitioning method is denoted as SP.



(a) 40 regions, constraint violations

(b) 90 regions, constraint violations

(c) 40 regions, gap in objective value

(d) 90 regions, gap in objective value

Figure 4.1: Convergence of ADMM with different partitions.

The Warm Start Case

Figure 4.1 shows the performance of ADMM with the partitions obtained by SP and EP, respectively, using a warm starting point. The 'Iterations' on the x-axis denotes the ADMM iteration as defined by (2.31). The Polish system is divided into 40 and 90 regions using both methods. In Fig. 4.1a and 4.1b, the blue curves show the progress of ADMM with the partitions obtained by SP, while the red curves show the progress of ADMM using EP partitions. The solid curve shows the largest bus power mismatch which is the maximum power flow balance violation, and the dotted curve shows the largest primal residue. With both 40-region and 90-region partitions, the blue curves are generally lower than the red curves, which shows that ADMM with SP approaches feasibility faster than with EP.

It is worth noticing in Fig. 4.1a and Fig. 4.1b that the primal residue reaches 10^{-4} first

compared with the power flow mismatch. This is due to the fact that in ADMM, the primal residue represents the difference between the copies of the voltages at the same bus. Since voltage errors propagate to line flow errors through multiplication by line admittances, it is necessary to add power flow feasibility checks in the termination condition of ADMM.

Figures 4.1c and 4.1d show the relative error (in %) of the objective value achieved by ADMM with respect to the one obtained by a centralized method, which is denoted as the gap in the objective value. As ADMM with increasing penalty generally provides no guarantee to converge to a local minimum for non-convex problems, a gap is always observed at the solution obtained by ADMM. Note that by increasing the number of iterations, the gap can only be reduced very slowly but not completely eliminated. As shown in Fig. 4.1c and 4.1d, ADMM with SP converges faster than with EP and achieves a smaller gap.

Table 4.1 further compares the effects of partitions on the number of iterations, the estimated computation time, and achieved objective value of ADMM. Again, the computation time is estimated by assuming that all subproblems are solved in parallel. Hence, it assumes that the time it takes to solve all subproblems at each ADMM iteration equals the maximum time needed to solve a single one. The time spent on information exchange is not accounted for in this chapter. The number of iterations and computation time measure the efficiency of the distributed method, while the gap measures the quality of the solution achieved by the distributed method. As shown in Table 4.1, the computation time of ADMM using SP is significantly smaller than when using EP. The reason is that the regions defined by EP can be highly imbalanced, having several large regions that contain generators electrically close to many loads. As a consequence, solving the subproblems associated with those large regions can be time-consuming. In contrast, spectral partitioning aims to find balanced partitions [84], hence it usually generates more balanced regions. Furthermore, the solution quality achieved by ADMM is satisfactory as the gap stays below 1% if SP is used.

Here, we would like to provide further information regarding the convergence characteristics of Lagrange multipliers associated with power flow constraints (2.1b), which could be of particular interest in practice. It has been proven in [87] that for non-convex problems, a sufficient condition for these Lagrange multipliers to converge is that the multiplier λ of the constraint (2.29b) converges to a finite value, which is generally the case with a *fixed*

Table 4.1: Comparison of different partitions with ADMM.

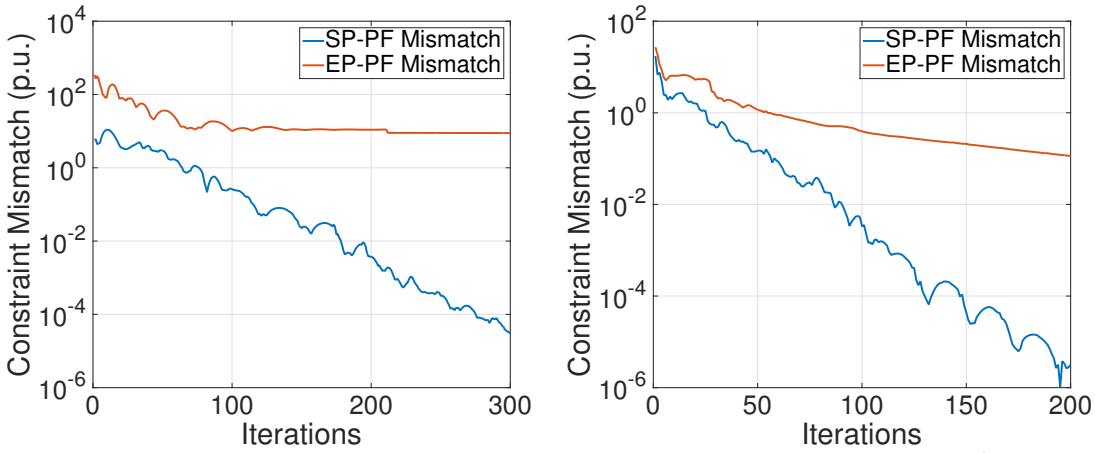
Partition	SP, 40 regions	EP, 40 regions	SP, 90 regions	EP, 90 regions
Iterations	97	119	110	128
Time (s)	218	2442	133	1039
Gap	0.43%	1.10%	0.65%	0.78%

Table 4.2: Convergence of Lagrange multipliers with different upper bounds on the penalty parameter.

Maximum penalty	Iterations	Gap	Difference of marginal costs		
			Mean	Median	Max
10^7	>500	0.01%	0.5%	0.3%	2.0%
10^9	332	0.22%	6.1%	5.2%	22.4%
10^{11}	120	0.43%	35.1%	22.8%	121.7%

penalty parameter ρ. However, to speed up ADMM, the penalty parameter ρ is usually increased over iterations, which, on the other hand, does not guarantee the convergence of the Lagrange multipliers. To enforce the convergence of the multipliers, an upper bound of the penalty parameter is imposed and its impact on the performance of ADMM as well as the convergence of the Lagrange multipliers is shown in Table 4.2. Specifically, we study the convergence of marginal costs, i.e., the multipliers associated with the nodal active power flow equations. The difference between the centralized and distributed marginal cost at one bus is calculated by $\frac{\text{marginal cost solved by centralized solver - marginal cost solved by distributed approach}}{\text{marginal cost solved by centralized solver}} \cdot 100\%$ and some statistics over the marginal costs at all buses are reported in Table 4.2.

As shown in Table 4.2, the larger the upper limit of the penalty parameter, the faster ADMM converges to a feasible solution, but the larger the gap and the difference between the marginal costs solved by the centralized and distributed approaches. The reason for this is that a large penalty parameter enforces the subproblems to reach an agreement more quickly, which, however, leads to a large deviation of the optimality conditions solved by subproblems from the centralized optimality conditions. Hence, there is a tradeoff between fast convergence and small gap with ADMM using an increasing penalty parameter.



(a) 40 regions, the flat start case (b) 90 regions, $\rho_0 = 10^6$

Figure 4.2: Mismatch in the power flow constraint with different partitions.

The Flat Start Case

We now evaluate the performance of ADMM with different partitions in the flat start case and with different penalty parameters. We expect that with a good partition, ADMM should also converge from a remote starting point and not be affected much by the choice of parameters.

In the flat start case, the initial penalty parameter ρ_0 and the incremental rate τ need to be set to lower values than those used in the warm start case. For the experiments, we use $\rho_0 = 10^4, 10^5$ and $\tau = 1.05, 1.1$, respectively. A smaller value of ρ_0 may enable ADMM to converge if the algorithm runs long enough but will lead to significantly larger computation time which is impractical. The system is partitioned into 40 regions using the SP and EP methods. With the partition found by EP, ADMM fails to converge with all combinations of parameters stated above, while with the partition found by SP, ADMM only fails when $\rho_0 = 10^5$ and $\tau = 1.1$. Figure 4.2a shows the best performance of ADMM out of the four settings of parameters using the two 40-region partitions computed by SP and EP, respectively, with a flat start. With SP, ADMM still converges to a feasible point, while with EP, ADMM fails to converge. For the solved case, ADMM takes 276 iterations, which results in 674 seconds, and the gap is 2.92 %. Although the speed and accuracy of ADMM degrade in the flat start case compared with the warm start case, ADMM with the SP partition still converges to a feasible point. The feasible solution obtained by ADMM with the SP partition can be used to initialize another run of the distributed solver, which

will further reduce the gap.

Figure 4.2b shows the performance of ADMM using different 90-region partitions with a different initial penalty $\rho_0 = 10^6$ in the warm start case. Again, ADMM only converges with the SP partition, while it fails to converge with the EP partition, which shows that ADMM is more robust with respect to the parameter settings using SP.

Impact of the Number of Regions

In this section, we evaluate how the performance of ADMM changes as the number of regions changes. In the following experiment, the SP method is used to find partitions of the Polish system with 10 to 140 regions with an increment of 10 regions between any two partitions. These partitions cover both a mild partitioning of the system where there are hundreds of buses in a region and a severe partitioning of the system where there are only tens of buses in a region. The performance of ADMM with different number of regions is illustrated in Fig. 4.3 for the warm start case.

Figure 4.3a shows the decrease of the maximum bus power mismatch, where each curve corresponds to a partition with a different number of regions. Impressively, with all the partitions, a feasible point can be found within 150 iterations regardless of the level of partitioning of the system. Figure 4.3b shows that the number of iterations does not increase significantly as the number of regions increases. Figure 4.3c shows the computation time with different number of regions, where the grey horizontal line denotes the time spent on solving the AC OPF problem in a centralized manner using the SNOPT solver. The blue curve shows the convergence time of ADMM and the red curve shows the speed-up of ADMM with respect to the centralized approach calculated by "centralized computation time/convergence time of ADMM". As expected, the computation time of ADMM tends to decrease as the system is partitioned into more regions, since the time spent on solving subproblems can be greatly reduced if the scale of the subproblems is small. However, as shown in Fig. 4.3d, the gap increases with the increase of the number of regions.

Notably, in Fig. 4.3c, there is a cross-over point (around 40 regions) of the blue and grey curves beyond which the computation time of the distributed approach is smaller than the centralized approach. This shows that in terms of the estimated computation time, there

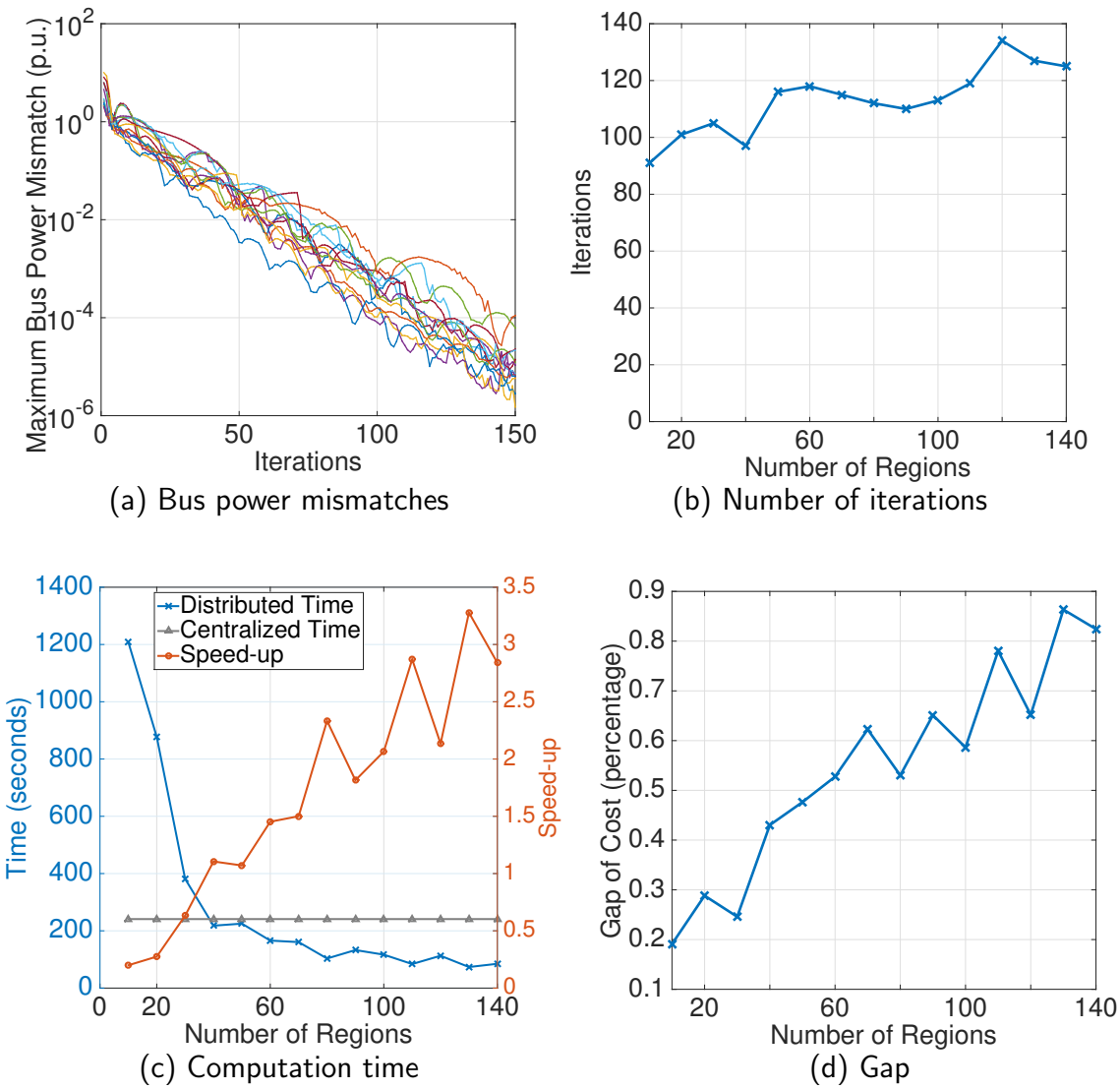


Figure 4.3: Performance of ADMM with different number of regions.

is a critical number of regions in implementing the distributed method to outperform the centralized approach. However, this does not necessarily imply that one should partition the system into as many regions as possible because more regions imply more communications, which is an important concern in distributed optimization that will be investigated in the next chapter. While the time needed for information exchange is not accounted for in this chapter, it is clear that partitioning the system into more regions would require more communications among the regions, which may lead to large delays. In addition, a partition with more regions would require more regional computational entities. Hence, the number of regions is a design parameter that should satisfy specific requirements of the utilities in

terms of both the performance of the distributed method and the available computational and communication resources.

Adding Line Limits

We now evaluate the performance of ADMM with thermal line limits (2.1g) included. We consider the following two scenarios. In the first scenario, the line limits of all the transmission lines are added to the constraint set while in the second scenario only the line limits of several lines prone to congestion are added. With the partitions devised in previous sections, there are tie lines congested in each of the partitions. To investigate whether the congestion of tie lines has significant impact on the performance of ADMM, another partition is devised by manually increasing the affinity between the buses that the congested lines connect to in the SP method. As a consequence, the obtained partition does not include any congested tie lines as each congested line is included in a single region.

Table 4.3 shows the performance of ADMM with all line limits added. In this setting, the centralized approach takes 355 seconds to convergence, which is longer than the case without line limits. As shown in Table 4.3, with line limits ADMM requires more iterations

Table 4.3: Performance of ADMM with different partitions when all line limits are included.

Partition	40 regions, no tie line congested	40 regions, tie line congested	80 regions, tie line congested
Iterations	161	124	147
Time (s)	679	511	237
Gap	1.92%	1.56%	2.25%

Table 4.4: Performance of ADMM with different partitions when few line limits are included.

Partition	40 regions, no tie line congested	40 regions, tie line congested	80 regions, tie line congested
Iterations	129	125	127
Time (s)	304	346	122
Gap	1.05%	0.96%	1.25%

and time to terminate and the optimality gap is also larger. This is due to the fact that solving the subproblems takes longer and an agreement on the voltages at the boundaries is harder to achieve with the additional line constraints. However, comparing the two 40-region partitions given in Table 4.3, we notice that the performances of ADMM are similar whether the congestion occurs on tie lines or non-tie lines. With the partition with tie lines congested, ADMM even performs better than with the partition where the congested lines are forced to be included in a single region. This observation suggests that the partition obtained by SP does not need to be changed even if congestion occurs on tie lines.

Better performance of ADMM can be achieved when only a few line limits are incorporated, as shown in Table 4.4. The included line limits are the ones associated with the congested lines when all the line constraints are considered. With all the three partitions, a gap of around 1% can be achieved. Hence, with only certain line constraints considered, ADMM can still reach a high quality solution efficiently.

4.1.3 More Test Cases

Our results provide initial evidence that the recently developed partitioning technique [88] in conjunction with the ADMM method can be used for distributed optimization in real-world large-scale power systems most of which are non-convex. However, it remains to be seen whether this approach can apply to other transmission networks of similar scales as well. This is a legitimate concern that requires further research. As a first step, we have also tested the developed distributed optimization approach on two other large-scale transmission networks, namely, the Great Britain 2224-bus system and the Polish 3012-bus system during the winter period, whose configurations are also taken from MATPOWER. These new simulations are conducted using Matlab v8.5 on an iMac. As shown in Table 4.5, the proposed partitioning method can again find good partitions for these two test systems with which ADMM can converge to a solution close to the local optimum very efficiently compared with the time required by the centralized solution. It is worth mentioning that, in contrast, for all the additional test cases, with the partitions found by the electrical distance-based partitioning (EP) method, ADMM fails to converge to a feasible point within 200 iterations which also takes more than 500 seconds. These results provide additional evidence on the validity of

Table 4.5: Convergence of ADMM on real-world systems.

Test system	GB 2224-bus		Polish 3012-bus	
Centralized time (s)	485		208	
Number of Regions	53	70	50	86
Iterations	165	144	132	159
ADMM Convergence Time (s)	342	187	214	197
Gap	0.51%	0.52%	0.22%	0.52%

the proposed approach.

While the evidence shown in this chapter is not sufficient to guarantee that this approach would work effectively for all non-convex problems and for other decomposition techniques, it is a promising step in this direction. As demonstrated in Section 4.1.2, the efficiency and robustness of a distributed method is highly dependent on the system partitioning method used. Fortunately, the spectral partitioning technique is indeed very effective in improving the efficiency and robustness of distributed methods.

While the results of this chapter look very promising, one limitation of the presented approach is that it can only find solutions close to *local optima* of non-convex problems, which, however, is also a general limitation of most distributed optimization approaches applied to non-convex problems. Another issue regarding ADMM is that there is usually a trade-off between faster convergence and higher solution quality. This is due to the fact that the Lagrange multipliers associated with the constraints are not guaranteed to converge with the penalty parameter increasing to infinity in the non-convex case [87]. As presented in the results, the larger the penalty parameter, the faster ADMM converges but the larger the gap and the difference between the Lagrange multipliers solved by the centralized approach and ADMM. One way to enhance the convergence of multipliers is to impose an upper bound on the penalty parameter, which, on the other hand, might lead to an increased number of iterations.

4.2 Multi-Step Optimal Power Flow

4.2.1 Introduction

With an increasing number of intermittent energy resources and storages integrated into the power system, the question that arises is how to optimally coordinate these resources to overcome the uncertainty introduced by the intermittent resources and the inter-temporal coupling of storages. Multi-step optimization approaches based on Model Predictive Control (MPC) have been proposed to address this challenge by determining the optimal states of the controllable devices with a look-ahead scheme that accounts for the temporal characteristics of both intermittent resources and storages [89, 90]. Such MPC-based approaches can effectively coordinate the intermittent resources and storages, hence reduce the total generation cost over a certain period of time.

However, the MPC approach is computationally expensive since the problem size grows drastically as the optimization horizon increases. Such computation complexity restricts the practical use of MPC because no control actions can be taken if an MPC problem is not solved within the required amount of time due to the lack of computation capability or storage capacity at the central computation entity. To address this issue, distributed MPC has been studied and applied to various applications [91, 92, 93, 94], where one common approach is to use decomposition techniques to decompose the optimization problem into subproblems that can be solved in parallel [95, 96] with only a small amount of data shared among neighboring regions to achieve the overall optimality of the entire system.

While distributed approaches can alleviate the centralized computational burden, most distributed optimization methods are iterative and generally take many iterations to converge for the MPC problem. To improve the efficiency of the MPC approach, we apply the proposed spectral partitioning method in conjunction with OCD-C to solve a multi-step AC OPF problem. Specifically, we optimize the usage of the wind generation and minimize the total generation cost over a 24-hour time period by optimally setting the charging/discharging status of the storages. This section contributes to existing work in the following two aspects. First, we show that the efficiency of the distributed MPC can be significantly improved by using a good partition of the system. Such efficiency gain can be very beneficial for the

integration of renewable resources in practice. For example, if an MPC problem can be solved in shorter time, the calculation can start closer to the dispatch time (e.g., an hour or two ahead of time) when the wind forecasts are more accurate compared to day-ahead forecasts. Moreover, a faster distributed method provides more capability to handle MPC problems with longer time horizons or higher time resolutions. Second, different from Chapter 3.4 where the best partition of the system is computed and evaluated for a single time step, we demonstrate in this paper that the same partition can be used for solving the MPC problem for multiple time steps, which eases the practical use of distributed MPC as the partition of the system does not need to be changed frequently. Apart from AC OPF, the proposed approach can also be applied to solve similar multi-step optimization problems in a distributed fashion as well.

4.2.2 Problem Formulation

Figure 4.4 shows an example of the wind and load data for a 24-hour period obtained from the Bonneville Power Administration at a 10-minute increments. As seen from Fig. 4.4, the wind generation at different times of the day is usually random and it is possible that the wind generation is high while the demand of the system is low. Hence, to reduce the generation cost at the peak demand, the procedure is to store the excessive energy generated by the cheap generators during the time when the wind generation can serve most of the system load and use the stored energy when the demand is high. With this purpose in mind, we formulate a multi-step AC OPF problem where the objective is to minimize the total generation cost of non-renewable generations over a finite time horizon. It is assumed that the wind generation must be consumed when available, hence it can be treated as negative load at the bus where the wind generation is placed. The overall multi-step AC OPF problem at time step T is formulated as follows:

$$\text{minimize} \quad f(P) = \sum_{t=T}^{T+N-1} \left(\sum_{i=1}^{n_b} \left(a_i P_i^2(t) + b_i P_i(t) + c_i \right) \right) \tag{4.2a}$$

subject to

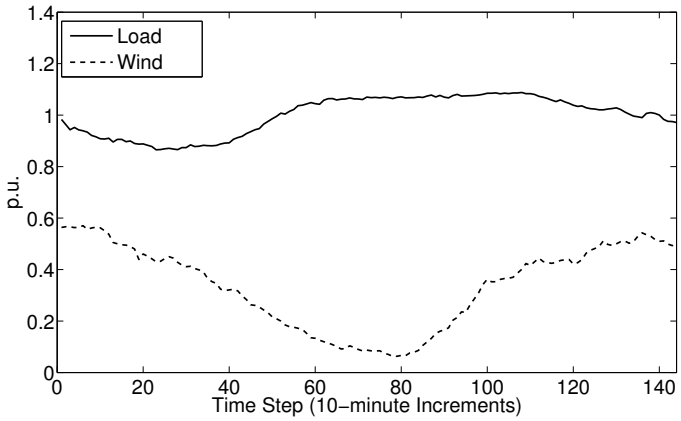


Figure 4.4: 24-hour load and wind data with 10-minute intervals.

$$P_i(t) + P_i^W(t) - P_i^{In}(t) + P_i^{Out}(t) - P_i^d(t) = |V_i(t)| \sum_{j \in \Omega_i} |V_j(t)|(G_{ij} \cos\theta_{ij}(t) + B_{ij} \sin\theta_{ij}(t)) \tag{4.2b}$$

$$Q_i(t) - Q_i^d(t) = |V_i(t)| \sum_{j \in \Omega_i} |V_j(t)|(G_{ij} \sin\theta_{ij}(t) - B_{ij} \cos\theta_{ij}(t)) \tag{4.2c}$$

$$E_i(t+1) = E_i(t) + \eta_c P_i^{In}(t) - \frac{1}{\eta_d} P_i^{Out}(t) - \epsilon_{sbl} \tag{4.2d}$$

$$V_i^{\min} \le V_i(t) \le V_i^{\max} \tag{4.2e}$$

$$\theta_i^{\min} \le \theta_i(t) \le \theta_i^{\max} \tag{4.2f}$$

$$P_i^{\min} \le P_i(t) \le P_i^{\max} \tag{4.2g}$$

$$0 \le P_i^{In}(t) \le P_i^{In,\max} \tag{4.2h}$$

$$0 \le P_i^{Out}(t) \le P_i^{Out,\max} \tag{4.2i}$$

$$E^{\min} \le E_i(t+1) \le E^{\max} \tag{4.2j}$$

$$(|V_i(t)||V_j(t)||Y_{ij}| \cos(\theta_{ij}(t) - \delta_{ij}))^2 + (|V_i(t)||V_j(t)||Y_{ij}| \sin(\theta_{ij}(t) - \delta_{ij}))^2 \le (S_{ij}^{\max})^2 \tag{4.2k}$$

for $t = \{T, \ldots, T+N-1\}$ and $i = \{1, \ldots, n_b\}$. Most notations used in the above formulation are the same as those used in the standard OPF formulation (2.1) except that they are dependent on time step t. N and T denote the length of optimization horizon and the initial time step, respectively. P_i^{In}, P_i^{Out} and E_i denote the power injected into/drawn from storage and the energy level in the storage at bus i, respectively. θ_{ij} is the difference of voltage angles between bus i and bus j. G_{ij} and B_{ij} are the real part and imaginary part of

the admittance matrix element y_{ij}, respectively. ϵ_{sbl}, η_c and η_d are the standby loss, charging and discharging efficiency of the storage.

Equations (4.2b) and (4.2c) are the active and reactive power flow balances at each bus. Equation (4.2d) corresponds to the inter-temporal constraints on storages. As a standard procedure in MPC, the solution found for the first time step is applied once the overall problem is solved. Then the optimization time horizon is shifted by one time step and the optimization problem is formulated and solved for the next time step.

In the following, we use the spectral partitioning with OCD-C to solve problem (4.2). For the MPC problem, the operating point changes with the time step as the load and wind generation vary during a day. We calculate the best partition at the operating point of the base load level with $N = 1$. This partition is applied to solving the MPC problem at all time steps, which, as will be shown later, is also good for solving the MPC problem with an increased time horizon. In Chapter 4.2.3, it will be further discussed how to choose the operating point for applying the partitioning method and handle the scenarios when different lines become congested.

4.2.3 Case Study

The distributed MPC approach is tested on the IEEE-14 bus and 118-bus test systems. For the 14-bus system, a wind generator is located at Bus 5 and a storage device is located at Bus 14. While for the 118-bus system, a wind generator is placed at Bus 19 and a storage device at Bus 70. In this section, we will present two sets of simulation results. First, the results using different time horizons are compared, which show the benefit of the MPC approach in terms of reducing the generation cost and the ramping of the generators. Next, the convergence speeds of OCD-C for different partitions are compared, which demonstrates the importance of system partitioning and the fact that the previously developed partitioning method can be effectively applied to the MPC problem to enable an efficient implementation of the distributed approach.

The simulations are run in Matlab on an iMac. The storage device has a roundtrip efficiency of $\eta_c \cdot \eta_d = 0.95\%$, standby loss of 0.005 $p.u.\cdot$10-minute and maximum capacity of 1.0 $p.u.\cdot$ 10-minute. The wind and load data in Fig. 4.4 is used which is normalized

according to the loads of both systems. The simulations are run for a 24-hour period using the time horizons of $N = 1, 3, 6$ and 9 with the time interval of 10 minutes, which correspond to no horizon, 30-minute, 60-minute, and 90-minute horizon, respectively. As the longest time horizon is 90 minutes and there are in total 144 time intervals over the 24-hour period, a total of 135 time steps are simulated using the available data. A multi-step AC OPF problem is solved at each time step. For comparison, the centralized optimization which uses Interior Point in combination with Newton-Raphson to update variables is also simulated. Convergence is declared if the L_2 norm of all the mismatch between the constraints is lower than 10^{-3}. The same starting point and convergence criterion are used for the OCD-C method with different partitions and the centralized approach. Again, note that the OCD-C always converges to the same solution as the centralized approach regardless of what partition is used.

Impact of the Optimization Horizon

In this section, we evaluate the impact of the length of the optimization horizon. Figure 4.5 shows the optimal storage energy level with different time horizons denoted by N. It is clear that the utilization of the storage increases as the time horizon increases for both test systems. The benefit of optimizing the usage of the storage is demonstrated in Table 4.6 which shows the total generation cost and the total generator ramping over a 24-hour period with different horizons. Again, as the time horizon increases, both the generator ramping

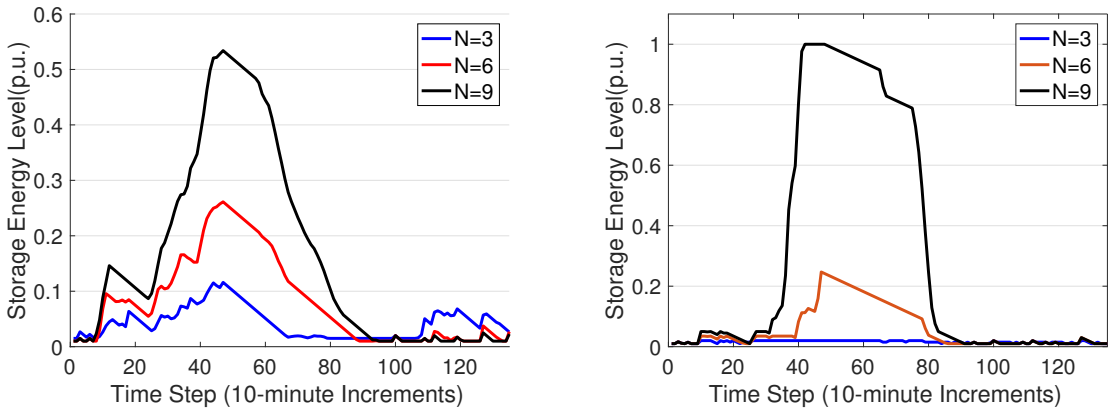


Figure 4.5: Optimal storage level of the storage device.

Table 4.6: Total generator ramping and total generation cost.

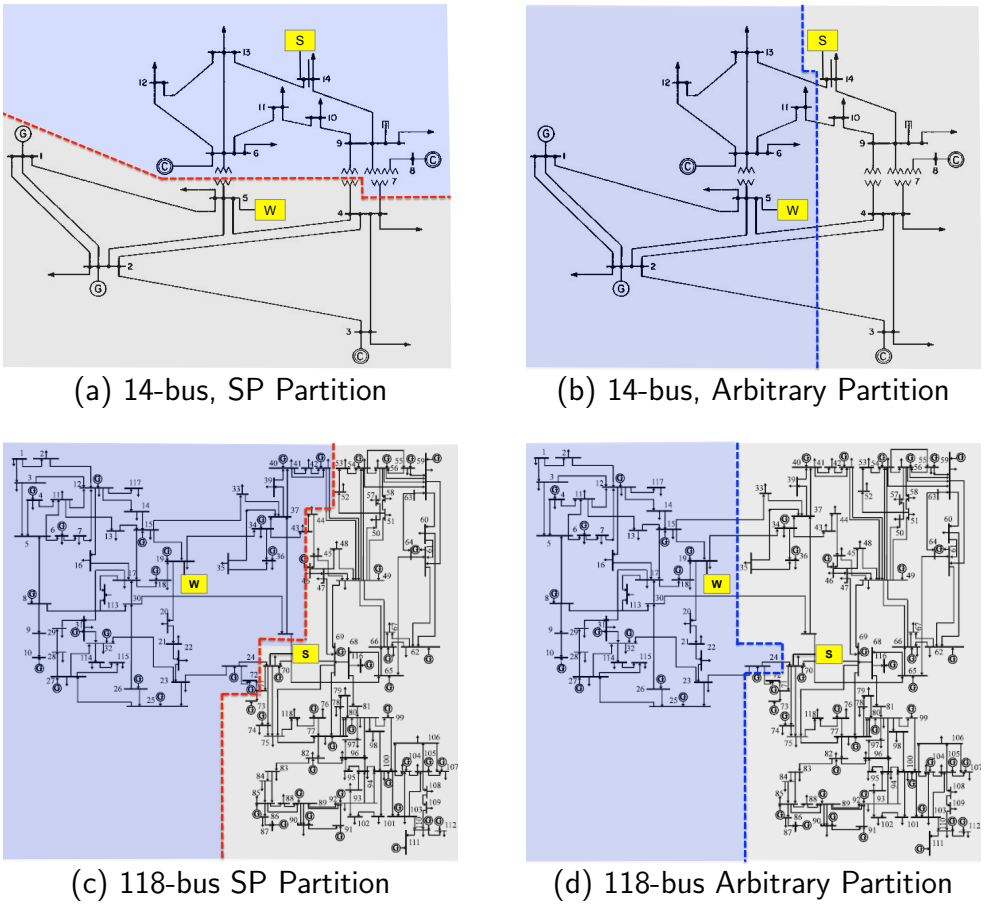
System	Time horizon N	1	3	6	9
14-bus	Generator ramping (p.u.)	4.1604	3.9598	3.7841	3.7708
	Generation cost ($)	679,145	679,144	678,985	678,874
118-bus	Generator ramping (p.u.)	54.70	54.53	54.27	53.93
	Generation cost ($)	12,067,322	12,067,313	12,067,198	12,066,768

and the generation cost decrease. Even though the generator ramping is not included as a hard constraint in the optimization problem, it has been reduced as the utilization of the storages smoothes out the fluctuations in the load. These results indicate that the MPC approach can effectively integrate the wind generation and storages especially with a longer time horizon. However, this does not indicate that one should extend the time horizon as long as possible, due to the fact that the problem size increases with the length of the time horizon. Hence, it takes more time and computation resources to solve the MPC problem with a longer time horizon. Besides, the forecasted wind and load data might not be available for a long time horizon at the required resolution. Overall, the choice of the length of the time horizon depends on specific applications and the computation capability and is beyond the scope of this thesis.

Impact of Partitioning

In this section, we focus on the efficiency of the distributed MPC approach and show that a good partition of the system is the key to efficiently implementing distributed optimization methods. For both the 14-bus and 118-bus systems, two partitions are used for the decomposition of the problem as shown in Fig. 4.6. "SP Partition" denotes the best partition determined by the spectral partitioning technique, while "Arbitrary Partition" denotes a geographical partition of the system which is likely to be chosen if one determines the partition only by observing the diagram of the system. Note that here we only partition the systems into two areas for comparison, but as shown in Chapter 3, the spectral partitioning method can be applied to find a partition of the system with more than two areas.

Of all the 135 time steps, the average, median and maximum number of iterations until



(a) 14-bus, SP Partition (b) 14-bus, Arbitrary Partition

(c) 118-bus SP Partition (d) 118-bus Arbitrary Partition

Figure 4.6: Partitions of the IEEE test systems with wind generation and storages.

convergence of OCD-C using the two different partitions of both test systems are shown in Table 4.7 and Table 4.8, respectively, and they are compared with the centralized approach. As shown in both tables, using spectral partitioning will lead to significantly reduced iterations compared to the arbitrary partition. Another observation is that with an increasing length of the optimization horizon N, the number of iterations does not necessarily increase substantially, especially while using the spectral partitioning approach. This is due to the fact that spectral partitioning identifies partitions with little coupling and such coupling does not increase with the optimization horizon.

In terms of the actual computation time, the average convergence time is shown in Table 4.9 and Table 4.10, respectively. It can be seen from Table 4.9 using spectral partitioning, the convergence time of OCD-C is only slightly higher compared to the centralized approach, while the convergence using the arbitrary partition is much slower. Note that the sparsity of

Table 4.7: Number of iterations to convergence for MPC with the IEEE 14-bus system.

Iterations	N	Centralized	SP Partition	Arbitrary Partition
Average	1	34	65	124
Median	1	34	73	135
Maximum	1	37	74	139
Average	3	46	76	148
Median	3	46	83	156
Maximum	3	66	108	210
Average	6	46	78	158
Median	6	46	84	159
Maximum	6	69	131	216
Average	9	47	80	166
Median	9	46	85	166
Maximum	9	91	132	235

the matrices is exploited in the simulation to speed up the calculation, which works more to the advantage of the centralized approach where the matrices involved in calculations are relatively sparser. Hence, on this toy example, it is fairly impressive that the distributed MPC approach achieves a comparable time efficiency as the centralized approach with spectral partitioning.

Even though for the IEEE-14 bus system, the centralized approach converges the fastest on average, the time efficiency of OCD-C will be superior than that of the centralized approach while deployed on the larger 118-bus systems, as shown in Table 4.10. This result shows an increased benefit of implementing distributed approaches on larger systems and that one can expect an efficiency gain of distributed approaches deployed on large-scale systems if the partition is chosen properly.

Now, to evaluate the robustness of the partition with respect to multiple time steps, the convergence time with the time horizon N=9 over all time steps are shown in Fig. 4.7. In the 14-bus case, there is no line congestion observed over all time steps. As shown in Fig. 4.7a, SP Partition, which is the best partition found by the partitioning method at base load level for N=1, always leads to a reduced convergence time compared to Arbitrary

Table 4.8: Number of iterations to convergence for MPC with the IEEE 118-bus system.

Iterations	N	Centralized	SP Partition	Arbitrary Partition
Average	1	43	61	102
Median	1	44	61	103
Maximum	1	57	64	127
Average	3	54	72	128
Median	3	52	70	127
Maximum	3	83	97	176
Average	6	54	76	145
Median	6	53	75	145
Maximum	6	84	106	225
Average	9	56	80	157
Median	9	54	77	154
Maximum	9	81	112	264

Table 4.9: Average convergence time (in seconds) for MPC with the IEEE 14-bus system.

N	Centralized	SP Partition	Arbitrary Partition
1	0.015	0.027	0.040
3	0.072	0.093	0.164
6	0.211	0.223	0.380
9	0.348	0.408	0.695

Partition for all time steps. Hence, the best partition is fairly robust and there is no need to change the partition for different time steps in this case. For the 118-bus system, a tie line associated with the best partition becomes congested during time step 1 to 13, which, however, does not affect the convergence time of the OCD-C method much. In other words, the best partition is robust for all time steps even when line congestion occurs. However, there could be cases where the convergence performance of the distributed method degrades once the tie line constraints become binding due to the increased computational coupling between the areas that the tie line connects.

Finally, we would like to provide insights into how one can determine the partition of

Table 4.10: Average convergence time (in seconds) for MPC with the IEEE 118-bus system.

N	Centralized	SP Partition	Arbitrary Partition
1	0.287	0.256	0.528
3	2.210	1.766	3.861
6	7.689	7.112	16.297
9	17.247	16.173	38.254

a system for different time steps when the load levels vary heavily over the considered simulation time frame. Since the best partition performs well for a fairly large range of load levels, one can find the best partition at the load level that occurs during most of the day. When the tie lines associated with the best partition are not severely congested as in the considered case, the best partition can be applied to all time steps. When the tie lines become severely congested, the partitioning method can be applied for that particular operating point to find a new partition. Overall, due to the robustness of the best partition, it can be expected that the computation effort spent on determining the partition of the system is quite low because the partition only needs to be computed for a few operating points with different line congestion scenarios.

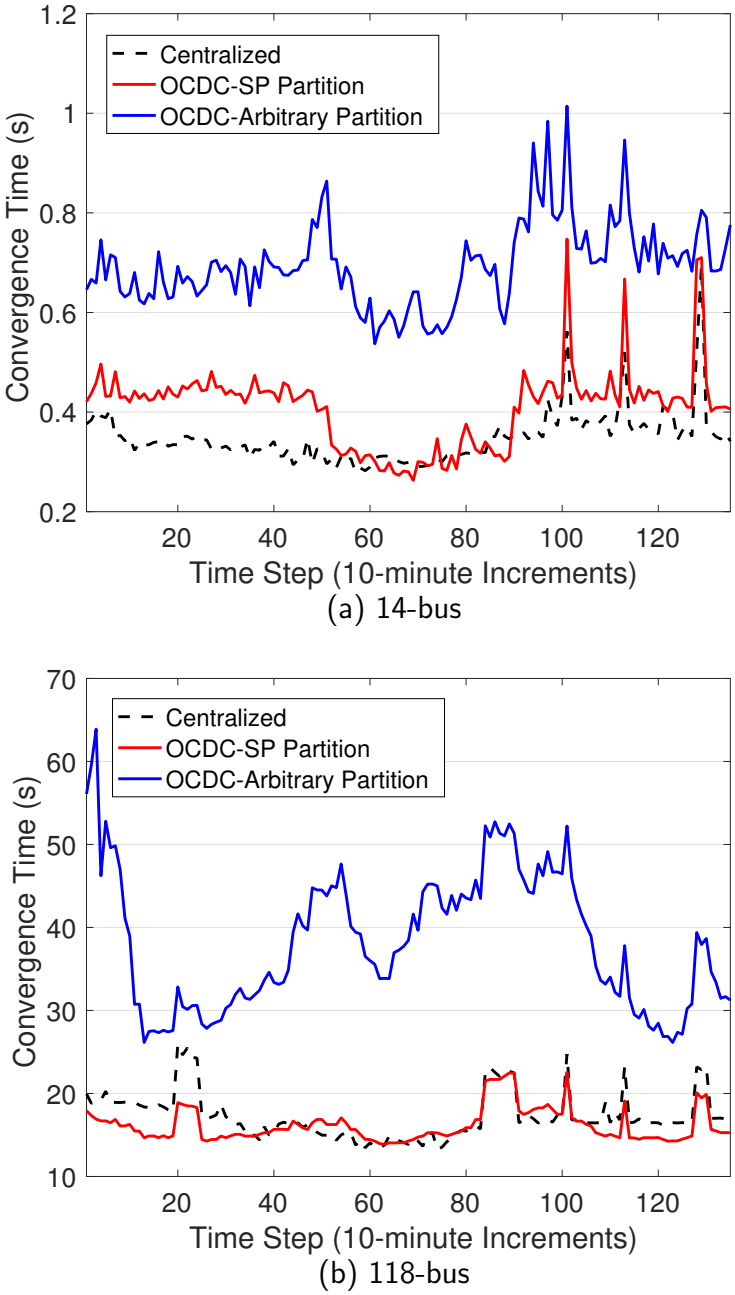


Figure 4.7: Convergence time of OCD-C with N=9.

Chapter 5

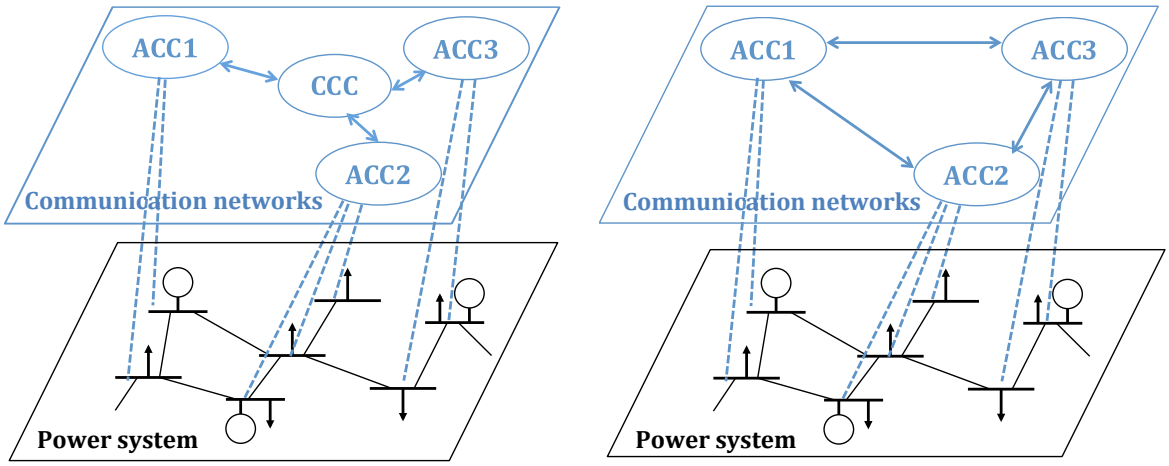
Role of Communications Plane

As shown in the previous chapters, the implementation of distributed methods depends on the information exchange among multiple control entities. Therefore, compared to the centralized approach, communications will have greater impact on distributed methods. However, it is not clear what communications infrastructures and technologies should be adopted for distributed methods in practice. To investigate this issue and bridge the gap between the power system communications research community and the distributed optimization research community, this chapter models the communications infrastructures and schemes required by distributed methods using the OPNET modeler. The capabilities and limitations of centralized and distributed communications infrastructures are investigated in terms of communications delays incurred and their impact on the convergence of OCD and ADMM.

5.1 Communications Technologies for Distributed Optimization

5.1.1 Communications Infrastructures

The smart grid is typically treated as a two-layered system: the power system overlaid with a communications plane that comprises a communications network, as shown in Fig. 5.1, where the dotted blue lines denote the communication links between devices and control



(a) Centralized Infrastructure (b) Distributed Infrastructure

Figure 5.1: Representations of the communications plane and the underlying communications infrastructures for distributed optimization.

centers, and the solid double-arrowed lines denote two-way communication links among the control centers. This integrated two-layered system entails a decentralized architecture, where a control center serves as a coordinator or an aggregation node for a geographical region. This paper only focuses on the upper-layer communications network and treats the control centers as end nodes. Thereby, the two communications infrastructures studied are defined as the centralized infrastructure and the distributed infrastructure, as illustrated in Fig. 5.1.

Even though no centralized coordinator is needed in distributed algorithms, the centralized communications infrastructure is still considered in this paper due to the fact that it resembles the SCADA system currently used in most utility networks, where the substations (or RTUs) can exchange data with SCADA, but not directly with other substations. As this centralized infrastructure is likely to be in use for the next ten to twenty years in many places, a better understanding of its capabilities for supporting distributed methods is needed. As shown in Fig. 5.1a, in the centralized infrastructure, a CCC connects to all the ACCs in a star topology. To deploy a distributed algorithm, the computation is still carried out at individual ACCs, while the CCC only controls the communications. Specifically, at each iteration, after all ACCs have solved their subproblems, the CCC collects from all ACCs the data that needs to be shared and redistributes that data to the designated ACCs.

In the distributed infrastructure, the communications topology is partially a meshed net-

work as shown in Fig. 5.1b. The major difference between the two infrastructures is that in the distributed infrastructure, ACCs can directly communicate with neighbors locally bypassing the CCC. However, note that a CCC might still exist in the system but only perform reduced functionalities such as determining the partition of the system or coordinating the ACCs under abnormal conditions.

The distributed infrastructure potentially has the following advantages: 1) if communication fails at one ACC, other parts of the grid might resume proper operations quickly whereas the failure at the CCC might harm the entire network in the centralized infrastructure; 2) the data generated by the intermediate steps of the optimization process does not need to be sent to one single hub such as the CCC, thus reducing the probability of traffic jamming at a certain node; 3) generally, the communication delay is shorter if a shorter direct path is available; 4) it is more scalable to large systems and an increased number of ACCs. However, additional communication links might need to be built among ACCs which may incur additional cost depending on the communications technologies adopted. Nevertheless, many actions are being taken to enhance direct and fast data communications among substations and the grid is evolving towards a more distributed architecture. Therefore, it is worth studying whether distributed methods yield better results when one uses a distributed communications infrastructure.

In the following sections, we introduce the communications protocols and schemes suitable for these two infrastructures, provide their models in OPNET, measure the communication delays, and evaluate under which circumstances they can support distributed methods. Although the choices of the communications schemes and technologies may vary with applications, similar analysis can be performed for other communications infrastructures following the proposed approach.

5.1.2 Communications Technologies

The choice of communications technologies in a smart grid depends on many factors such as the requirements of specific applications, the geographical characteristics of the network, the existing communications infrastructure, and so on, and there is no single solution as to which is the optimal technology to use. For distributed optimization, technologies that support

long distance communications are preferable as control centers are generally geographically far apart. Potential choices include wired communications technologies such as optical fiber and power line communication, as well as some wireless communications technologies such as cellular networks and WiMAX. However, deploying wireless technologies in the considered application is still at a very early stage where extensive studies are needed for demonstrating its feasibility and potential advantages.

This paper assumes optical fiber communication, which is adopted as the main technology for the power system backbone network to support fast and reliable data transmission [45, 97]. The topology of the communications network can be built either dependently or independently of the power grid. Same as the approach pursued in [41, 42], we choose to build the topology dependently of the grid in the simulations. More specifically, there are dependent media such as Optical Power Ground Wire (OPGW) and All-Dielectric Self Supporting (ADSS) cables that support optical fiber communication in power systems which are installed along existing overhead transmission lines [41]. Such dependent media is usually owned by utilities, hence can be used to build utility-proprietary networks over which the utilities have complete control.

5.1.3 Communications Protocols and Schemes

The Inter-Control Center Communications Protocol (ICCP or IEC 60780-6/TASE.2), which is the current protocol used in industry for communications between control centers, is a natural choice for distributed optimization [98]. The ICCP is based on the client/server principle, where the data exchange process is always initiated by a request from a client (one control center) to the server (another control center). In the considered infrastructures, the CCC and ACCs can act as both clients and servers. The ICCP is built over TCP/IP, with lower layer protocols Ethernet and SONET for the communications within a substation and among the control centers, respectively.

Based on the client/server principle of ICCP, we propose to study different communications schemes for the centralized and the distributed communications infrastructures. In the centralized infrastructure, the CCC requests from all the ACCs the information that each ACC needs to share with others in a polling fashion. The polling scheme is currently used for

the communications between the control center and substations [99]. After the collection of information from all ACCs, the CCC replies to each ACC with the information it requests. In contrast, in the distributed infrastructure, each ACC directly requests information from its neighboring ACCs in a polling fashion.

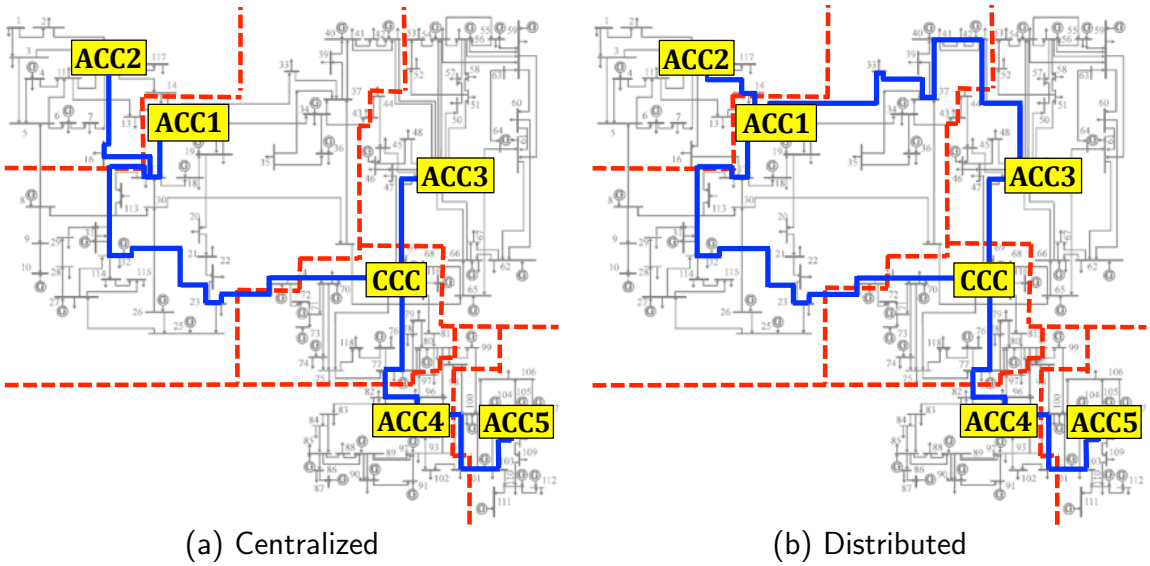
5.2 Communications Modeling in OPNET

The OPNET modeler is an event-based network simulator that enables the simulation of various communications networks with different applications. It adopts a hierarchical structure where the user can model the network topologies in the network domain, set up the nodes' properties in the node domain, and define specific states of the protocols in the process domain. In this section, we describe in detail how we use OPNET to model and simulate the communication process involved in distributed optimization.

5.2.1 Network Topology

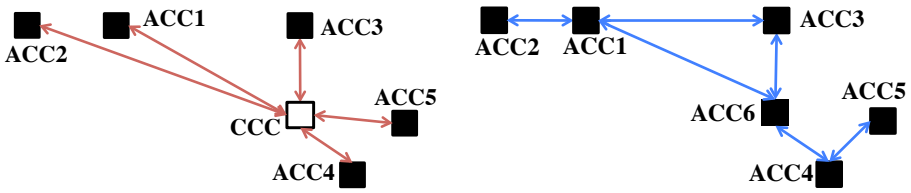
Note that the communication infrastructures discussed in Section 5.1.1 denote the *logical* infrastructure that defines which pair of control centers (CCs) can request and receive messages from each other.The first step of modeling communications in OPNET is to construct the network topology. The design of the actual *physical* network topology to connect the CCs is then dependent on the placement of routers, the choice of dependent/independent media and many other factors. In this study, we assume that a router is placed at each bus to connect the bus to the backbone network [45, 41] and dependent communications media is used; i.e., the communication link between any two buses only exists if there is a transmission line connecting them, and the length of the communication line is identical to the transmission line. The transmission line length between any two nodes is estimated using the line impedance from the system data.

The IEEE 118-bus test system is used as an example. Figure 5.2 shows the partition of the 118-bus system into six areas where the boundaries are marked by red dotted lines. The logical centralized and distributed network topologies associated with this particular partition is shown in Fig. 5.3, which demonstrates which pair of control centers need to



(a) Centralized (b) Distributed

Figure 5.2: Centralized and distributed network topologies associated with a 6-region partition of the IEEE 118-bus system.



(a) Centralized (b) Distributed

Figure 5.3: Logical communications topologies with centralized and distributed infrastructures.

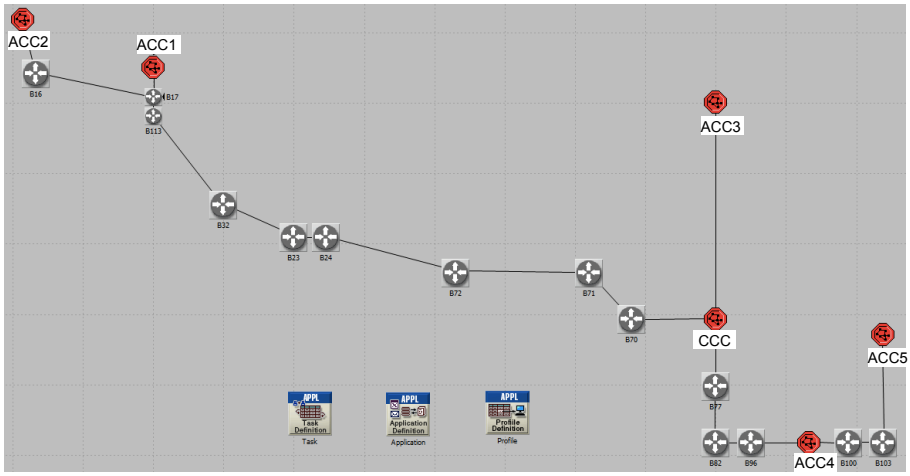
communicate according to their geographical connectivities as well as the chosen communications infrastructure and scheme. In Fig. 5.3, the black square and the white square denote ACC and CCC, respectively. Red lines denote communication links between CCC and ACC, and blue lines denote direct communication links between ACCs.

Next, the locations of the control centers need to be determined. Since an ACC needs to communicate with all the buses in its area, we consider locating it as close to most buses as possible to reduce the propagation delay. In the distributed infrastructure, the ACC is thus placed at the bus that has the shortest paths to all other buses within the same area on average. Then a shortest communication path between any pairs of neighboring ACCs is built using the Dijkstra algorithm [100], which is illustrated in Fig. 5.2b with the blue solid lines. In the centralized infrastructure, the ACCs are first located in the same way as

in the distributed infrastructure. Then, slightly different from Fig. 5.1a where an additional control center is built to act as the CCC, the ACC (ACC6 in this case) that has on average the shortest paths to all other ACCs is chosen to be the CCC. The network topology is then built by constructing a shortest path between the CCC and each ACC, as shown in Fig. 5.2a. The number of routers to be placed equals the number of buses traversed by the shortest path between any two control centers. The reason for using the shortest path algorithm here is that our goal is to reduce the communication delays involved in running distributed algorithms, and as it will be shown later, propagation delay is dominant in the end-to-end delay of any communication path for the considered application. The network topology can be built differently if other goals need to be met; e.g., using minimum spanning trees if the total length of the communication links needs to be minimized.

Figure 5.4 shows the network modeling in OPNET employing the described procedures. The red icons in the form of an octagon denote the control center subnets and the grey round icons denote the routers. The communication links between any two routers are modeled as SDH STM-1 links that support data transmission rates up to 155.52 Mbps (also known as OC-3), which is sufficient for distributed optimization. Comparing Figs. 5.4a and 5.4b, one can see that more communication paths might need to be constructed using the distributed infrastructure to reduce the communication delays, such as the communication path between ACC1 and ACC3 in the upper part of Fig. 5.4b. Moreover, as shown in Fig. 5.4a, the shortest path from the CCC to ACC5 passes through ACC4. Hence, it would be very inefficient if ACC5 still communicates with the CCC as it only needs information from ACC4. This observation indeed shows that *the distributed communications infrastructure is closer to the architecture of distributed optimization schemes, hence is a more natural choice for the deployment of distributed methods.*

The control center is modeled as the subnet shown in Fig. 5.5 that consists of a workstation, an Ethernet switch and a router connecting the control center to the backbone communications network. The subnet configuration can be extended to simulate other applications, but the simple topology shown in Fig. 5.5 is sufficient for the purpose of this thesis. As with the specifications used in [45], the communications links in the control center subnet are modeled as 100Base-T links with the data transmission rate of 100 Mbps.



(a) Centralized

(b) Distributed

Figure 5.4: Centralized and distributed network topologies with 6-area partitions in OPNET.

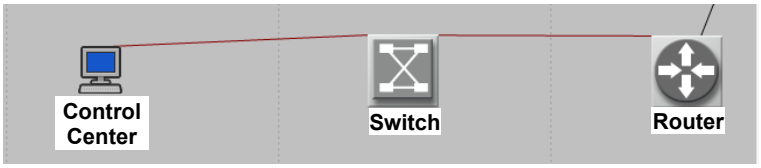


Figure 5.5: A simple topology of the control center subnet.

5.2.2 Application Definition

In OPNET, the communications schemes to be simulated can be defined as applications and then assigned to the nodes for execution. Different from the approach in [45] that approximately models the communication traffic using predefined applications in OPNET, we explicitly define a new application to implement the communications schemes presented

in Chapter 5.1.3. This is done by defining each request and response process as a task and executing them in a consecutive manner to emulate the communication phase where a control center requests information from others using a polling scheme.

In the centralized infrastructure, the defined communications scheme is deployed at the CCC periodically to simulate the communications throughout the iterative distributed optimization process whereas in the distributed infrastructure, the defined scheme is deployed at all ACCs periodically to simulate the procedure where ACCs directly communicate with their neighbors.

The polling periods are set to the computation time per iteration, and are therefore different for different number of regions and methods. The data to be exchanged is limited to the variables associated with the boundary buses and for the considered ADMM and OCD method, 8 and 6 variables need to be exchanged for each tie line [10, 11]. As the neighboring areas are generally connected by few tie lines, a total of tens of variables are expected to be exchanged. Hence, the message size is set to 1 kB in our simulation, which is large enough to contain these variables assuming single precision is used.

5.2.3 Performance Metrics

To compare the centralized and distributed communications infrastructures, we consider three performance metrics, namely, number of routers (NoR), length of media (LoM) [41] and communication delay. NoR is defined as the number of nodes on the path connecting two control centers including the end nodes and LoM is defined as the length of a communication path between any two control centers. Thereby, NoR and LoM can be obtained directly after a communication path is built among any two control centers.

Since there is a desired time limit for solving an optimization problem in the power grid to ensure that the determined settings can be applied in time, time efficiency is of great concern when evaluating distributed optimization methods. Therefore, we particularly estimate the communication delay and evaluate its impact on the execution time of distributed optimization methods. The communication delay per iteration is defined by the Application Response Time (ART) in OPNET, which is an estimated time of finishing the entire communication phase as defined in Chapter 5.1.3. Figure 5.6 shows the protocol stacks a packet

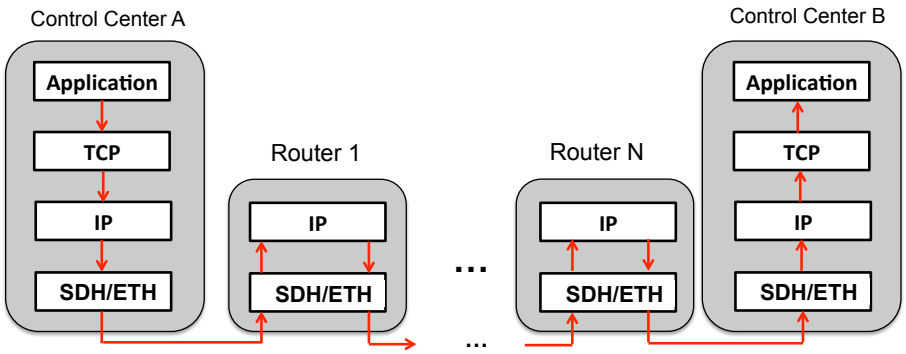


Figure 5.6: A simple network showing the protocol layers used to connect two control centers.

traverses when sent from one control center to the other, where SDH and ETH denote the protocols running over the fiber optics network and the Ethernet, respectively. Specifically, the communication delay for transmitting one packet is the time that the packet travels from the application layer of the sending entity (Control Center A in Fig. 5.6) until it reaches the application layer of the receiving entity (Control Center B in Fig. 5.6). Compared with the end-to-end delay at the IP layer, the ART is expected to be a more realistic estimation of the communication delay as it also includes the time of processing the packets at the TCP and application layers.

In the centralized infrastructure, the communication delay is the ART of the defined application at the CCC, which is expected to increase with the number of areas due to the considered polling scheme. Let $K-1$ denote the total number of ACCs, then the ART per iteration can be approximately expressed as

$$t_{\text{centralized}}^{\text{delay}} \simeq 2\sum_{i=1}^{K-1}(t_{\text{ete}}^{\text{CCC,ACC}_i} + t_{\text{app}}^{\text{CCC,ACC}_i}), \tag{5.1}$$

where $t_{\text{ete}}^{\text{CCC,ACC}_i}$ denotes the IP-layer end-to-end delay for any source-destination pair $(\text{CCC}, \text{ACC}_i)$ and $t_{\text{app}}^{\text{CCC,ACC}_i}$ denotes the delay occurring at the TCP and application layers. The factor 2 accounts for the round-trip-time for polling one ACC.

Similarly, the delay for the distributed infrastructure is estimated by the maximum ART of the applications deployed at all ACCs, i.e.,

$$t_{\text{distributed}}^{\text{delay}} \simeq \max_{k=1,\ldots,K}\{2\sum_{i\in\mathcal{N}_k}(t_{\text{ete}}^{\text{ACC}_i,\text{ACC}_k} + t_{\text{app}}^{\text{ACC}_i,\text{ACC}_k})\}, \tag{5.2}$$

where $\mathcal{N}_k$ denotes the set of neighbors of region k. It can be seen from (5.2) that the delay in the distributed infrastructure is generally dependent on the ACC that has the most neighbors.

The end-to-end delay can be further estimated by the following formula [41]:

$$t_{\text{ete}} = t_{\text{trans}} + t_{\text{prop}} + t_{\text{router}} + t_{\text{queue}}. \tag{5.3}$$

Here, t_{trans}, t_{prop}, t_{router} and t_{queue} denote the transmission delay, the propagation delay, the routing delay and the queueing delay, respectively. t_{trans} and t_{prop} can be calculated as follows:

$$t_{\text{trans}} = \frac{P_s}{R_l}, \tag{5.4}$$

$$t_{\text{prop}} = \frac{\text{LoM}}{v}, \tag{5.5}$$

where P_s is the packet size and R_l is the link data rate. v denotes the propagation speed of light in fiber, which is taken to be $\frac{2}{3}c$ with c denoting the speed of light in free space.

The routing delay t_{router} is equal to the sum of all the routing delays at the routers on the considered communication path. It is assumed that the routing delay is the same for all routers and is equal to the transmission delay at the router, then the routing delay experienced by a packet from source to destination (excluding the end nodes) can be represented as

$$t_{\text{router}} = (\text{NoR} - 2) \times t_{\text{trans}}. \tag{5.6}$$

Then, (5.3) can be expressed as follows:

$$t_{\text{ete}} = (\text{NoR} - 1) \times \frac{P_s}{R_l} + \frac{\text{LoM}}{v} + t_{\text{queue}}. \tag{5.7}$$

Therefore, a smaller value of NoR or LoM indicates smaller end-to-end delay of a communication path. Furthermore, in terms of reliability, it is assumed that all routers have the same reliability values and the reliability of a link will reduce if its length increases [41]. Hence, a smaller value of NoR or LoM also indicates higher reliability of a communication path. Note that we omit the between packet delay in (5.3) as it is assumed that each message only contains one packet.

Table 5.1: Computation time (milliseconds) per iteration and number of iterations of the distributed methods.

Method	K=3		K=6		K=9	
	Time	Iterations	Time	Iterations	Time	Iterations
OCD	16	124	7	215	4	324
ADMM	310	62	260	67	160	78

5.3 Simulation Results

Same as previous chapters, we consider the application of distributed optimization methods OCD and ADMM to solve the AC OPF problem, but with the communications plane taken into account. The experiments are implemented in Matlab with the Tomlab SNOPT solver [86] on an iMac. The test system used are the IEEE 118-bus test system and the large-scale Polish power system. The partition of the system is obtained by using the spectral partitioning method proposed in Chapter 3.

5.3.1 Convergence Performance Without Delays

We first compare the convergence performance of these two distributed algorithms *without* considering communication delay, as done in previous chapters and in most studies on distributed optimization. Figure 5.7 illustrates the decrease of the maximum constraint mismatch and the convergence of the objective function to the centralized solution with OCD and ADMM. Particularly, we plot the convergence versus number of iterations and computation time, respectively. We define the stopping criterion for both methods to be that the maximum constraint mismatch should be smaller than 10^{-4} p.u. The average computation time per iteration and the number of iterations at convergence are shown in Table 5.1, where K denotes the number of areas. Note that at convergence, OCD achieves the same solution as the centralized optimization approach, while ADMM finds a feasible solution within 1% of the centralized solution which is generally acceptable for non-convex problems.

As illustrated by Fig. 5.7 and Table 5.1, the performances of these two methods are quite different. At each iteration, OCD spends very little time on computation as it only carries out one Newton-Raphson step but does not solve the subproblems to optimality

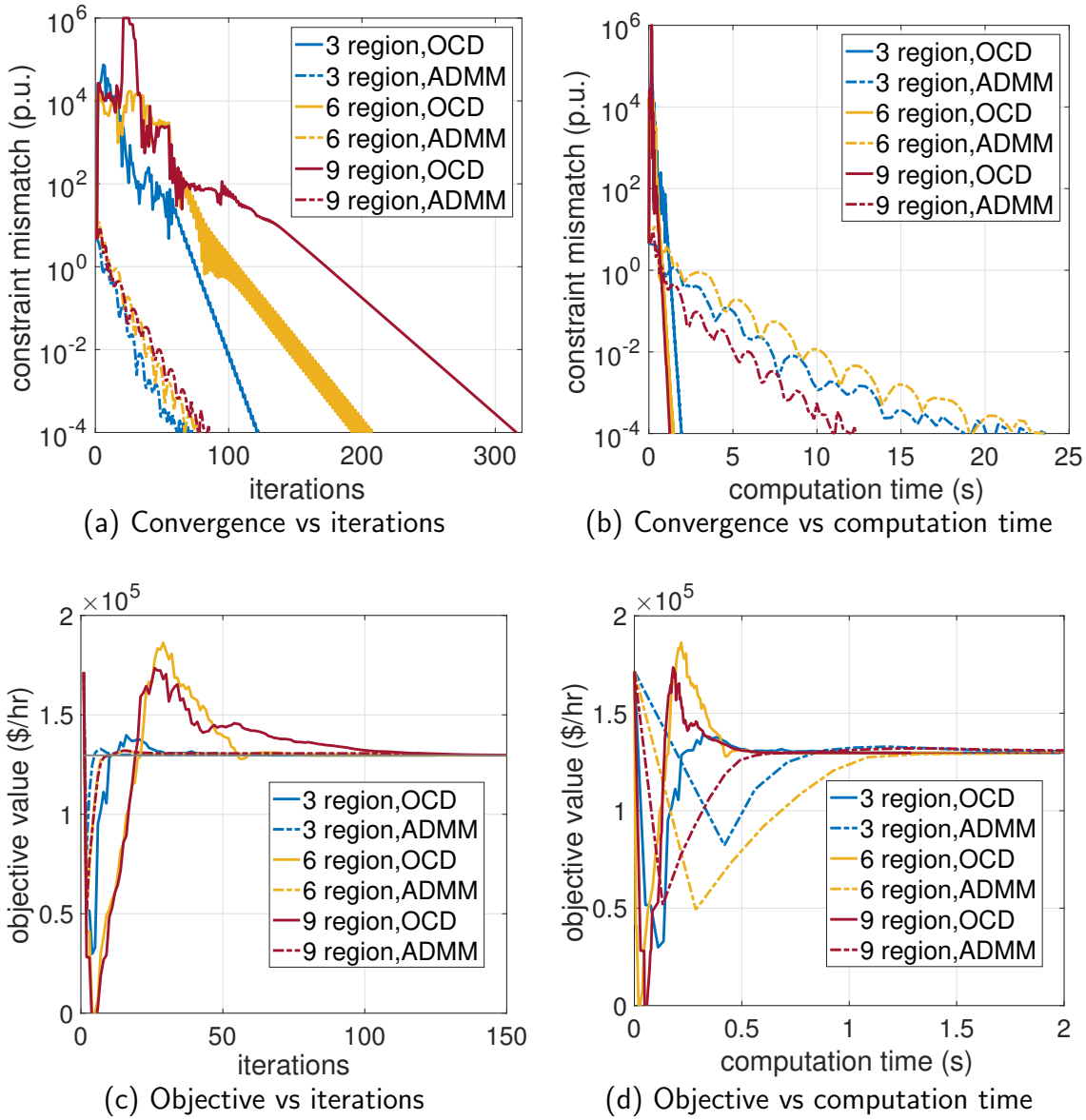


Figure 5.7: Convergence performance of OCD and ADMM.

[11]. However, the number of iterations increases significantly as the number of areas grows. In contrast, ADMM spends more time solving the subproblems because an optimal point needs to be found, but the iterations do not increase as much with an increasing number of areas [10]. Note that the exact value of the computation time might differ if one uses a different solver or machine. However, what is important here is the fact that for some distributed methods, the local computation can be finished very fast (in milliseconds) but more iterations are needed while for other methods, the subproblem may take seconds to be

solved but require fewer iterations. It is then expected that the impact of communication delays will be very different on these two distributed algorithms.

5.3.2 Comparison of Infrastructures

For each number of areas of the IEEE 118-bus system, both centralized and distributed communications networks are modeled following the approach presented in Chapter 5.2. The centralized and the distributed infrastructures for the 118-bus test system are compared in terms of NoR and LoM in Table 5.2. The LoM is normalized with respect to the longest transmission line length. The mean and variance are calculated over the communication paths between all pairs of nodes that need to communicate; i.e., the paths between the CCC and ACCs and the paths between neighboring ACCs for the centralized and distributed infrastructures, respectively. Note that for $K = 3$, the best partition obtained contains three areas that are connected in a row, where the areas at the two ends only need to communicate with the area in the middle. In this case, the centralized and distributed infrastructures are the same, and therefore have the same measures of NoR, LoM, and delay.

For 6 or 9 areas, the average NoR and LoM in the distributed infrastructure are both smaller than those in the centralized infrastructure, which indicates that the communication path between any pair of nodes in the distributed infrastructure is in general faster and more reliable. In addition, the variance of NoR is larger in the centralized case, which shows that the number of hops an ACC is away from the CCC can be very heterogeneous and it is inefficient for the ACC far away to communicate with the CCC. A special case here is that for $K = 6$, the variance of LoM is larger in the distributed infrastructure due to the existence of two long paths among the other short paths. However, this is a matter of design. There is a tradeoff between how close an ACC is to its internal nodes and to other ACCs, and the network topology can be designed differently if one wants to eliminate long communication links between the ACCs.

Table 5.3 shows the total LoM of different infrastructures. The distributed infrastructure has a larger LoM due to the additional links built to enable direct and faster communications between ACCs. However, most of the communication links in the distributed infrastructure might already exist in the centralized case. Hence, such increase in the LoM can be incre-

Table 5.2: Mean Value and Variance of NoR and LoM.

Metric	Infrastructure	K=3		K=6		K=9	
		Mean	Variance	Mean	Variance	Mean	Variance
NoR	Centralized	7	2	7.20	13.70	7.00	6.29
	Distributed	7	2	5.17	8.57	5.33	2.97
LoM	Centralized	1.027	0.087	0.862	0.186	0.938	0.166
	Distributed	1.027	0.087	0.651	0.244	0.694	0.088

Table 5.3: Total LoM of the communications network.

Infrastructure	K=3	K=6	K=9
Centralized	2.053	2.725	4.151
Distributed	2.053	3.908	4.740

Table 5.4: Network utilization with dedicated links in percentage (%).

Infrastructure	K=3		K=6		K=9	
	Mean	Max	Mean	Max	Mean	Max
Centralized	0.41	0.43	0.30	0.37	0.22	0.74
Distributed	0.41	0.43	0.45	0.87	1.20	2.64

mental which may not incur significant cost.

Figure 5.8 demonstrates the pair-wise end-to-end delay with 6 and 9 regions partitions. The source-destination pairs are sorted according to their end-to-end delay for better presentation. The grey bar shows the delay calculated by (5.3) assuming t_{queue} to be zero. This could provide a good estimate for the end-to-end delay if dedicated links are used; i.e., with no background traffic as illustrated by the dark blue bars. As shown in Table 5.4, the utilization rate of all links are very low if one uses dedicated links, and therefore the queueing delay is insignificant compared to other terms in (5.3). We also obtain the end-to-end delay with 50% and 90% background traffic on all links, where increased queueing delay occurs. Comparing Fig. 5.8a and Fig. 5.8b (or Fig. 5.8c and Fig. 5.8d), it is clear that the end-to-end delay in the distributed infrastructure is generally smaller than that in the centralized infrastructure, which also follows the trend observed for NoR and LoM.

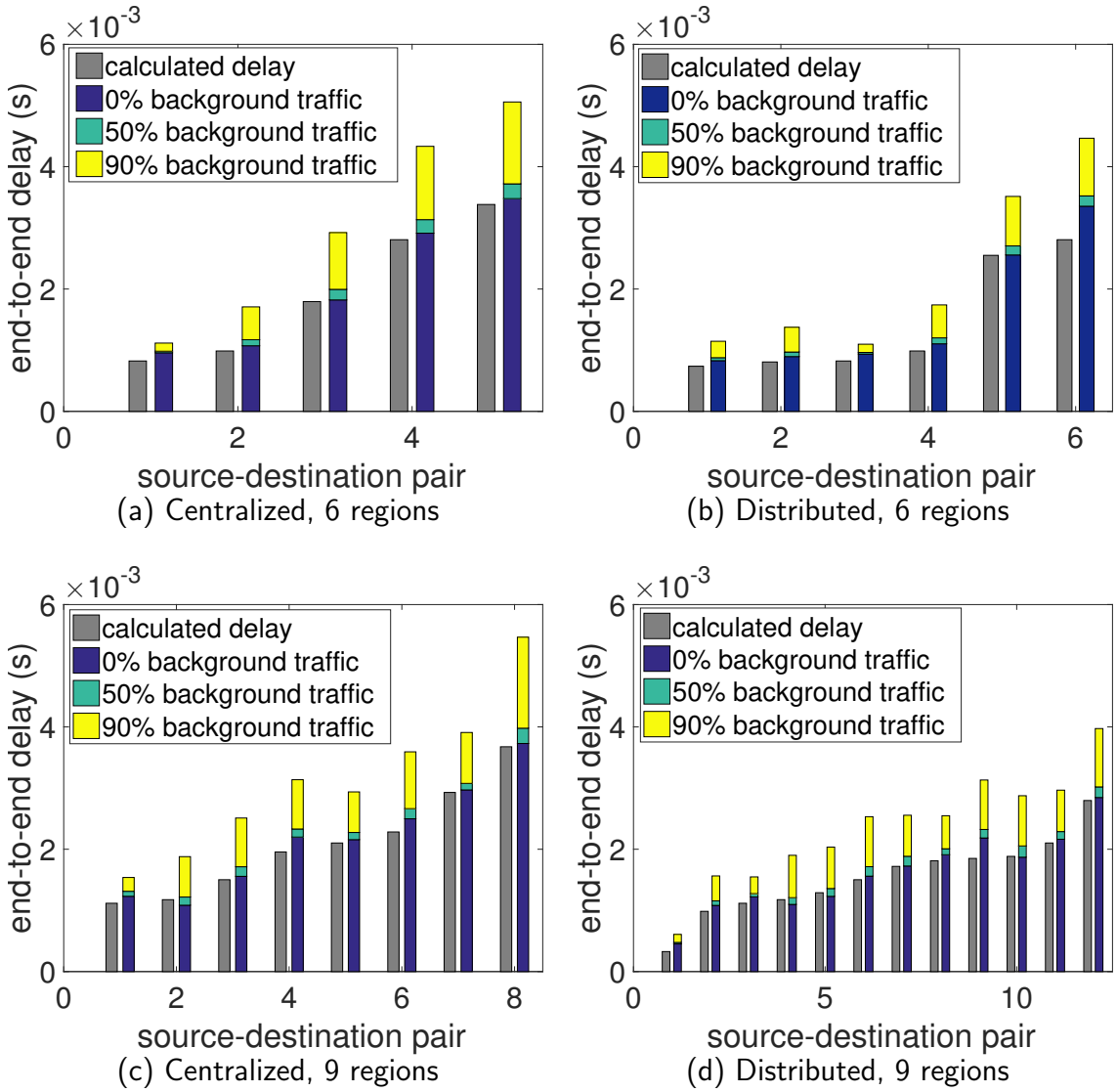


Figure 5.8: End-to-end delay in seconds with different background traffic.

Finally, the communication delays per iteration defined in (5.1) and (5.2) are shown in Table 5.5. As the delay at the TCP and application layers is usually dependent on the software and hardware used and therefore hard to estimate, it is only obtained via simulations. For 6 and 9 areas, the communication delay is always smaller using the distributed infrastructure than that using the centralized infrastructure. Moreover, the delay tends to increase more with the centralized infrastructure if the links become heavily loaded.

Table 5.5: Communication delay (milliseconds) per iteration.

Infrastructure	Background traffic	K=3	K=6	K=9
Centralized (ms)	0 %	19.9	66.8	113.8
	50 %	20.5	71.5	119.4
	90 %	27.0	97.2	159.2
Distributed (ms)	0 %	19.9	29.5	37.1
	50 %	20.5	30.5	38.6
	90 %	27.0	38.7	51.7

5.3.3 Impact of Communication Delays

Now we evaluate the impact of communication delays on different distributed optimization methods. Let t^{comp}, t^{delay}, ν denote the computation time per iteration, the communication delay per iteration and the number of iterations, respectively, then the total execution time is estimated by $(t^{\text{comp}} + t^{\text{delay}}) \times \nu$. The computation time at each iteration is approximated by the maximum time spent on solving any of the subproblems assuming that the subproblems are solved in parallel. Trade-offs between the communication delay and computation time, as well as the execution time per iteration and the number of iterations have to be considered when targeting to fulfill the time requirement, which is harder to achieve in the centralized infrastructure. In (5.1), the most prohibitive factor is the number of areas K. If K is large, the method to be deployed should have either small t^{comp} or ν, or both, which is thus limiting the choice of distributed methods. However, as the number of neighbors of one ACC is usually bounded, the total solution time increases less for the distributed case as K increases. Therefore, the distributed infrastructure is more scalable and better supports methods with more iterations.

As shown in Fig. 5.7b, OCD outperforms ADMM in terms of the total computation time regardless of the number of areas. *However, performances of these two techniques change significantly when communication delays are taken into account.* Figure 5.9 shows the execution time OCD and ADMM take to converge with communication delay (assuming dedicated links) added to each iteration. For OCD, the execution time increases with the number of regions because more iterations need to be carried out which incur large com-

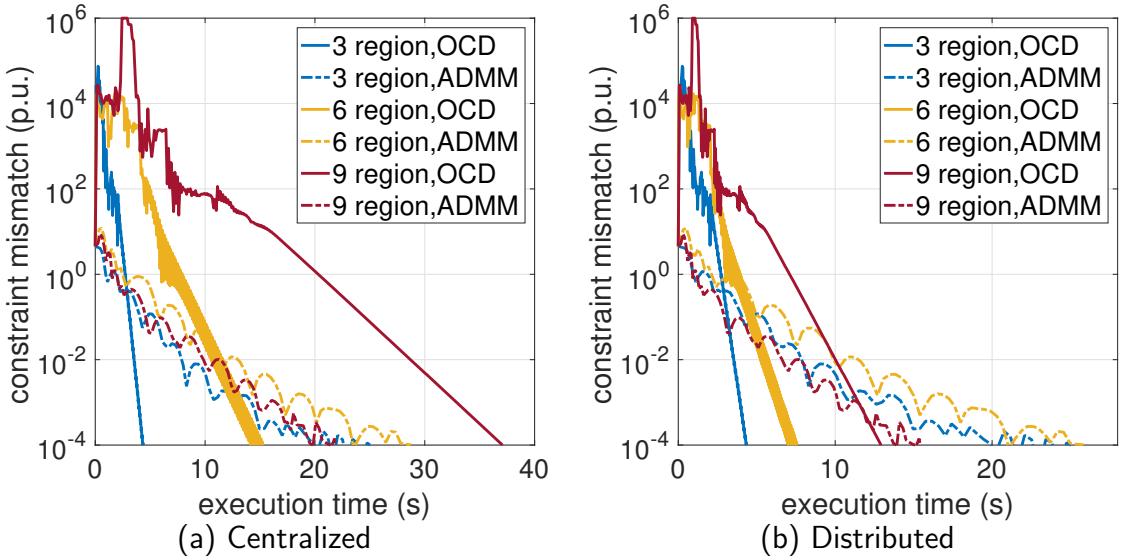


Figure 5.9: Convergence speed of OCD and ADMM with different communications infrastructures.

munication delays. When deployed in the centralized infrastructure with many regions, as shown in Fig. 5.9a, OCD becomes very slow, which makes it incompetent and not suitable for large-scale systems. On the contrary, for ADMM, the execution time depends more on how fast the subproblems can be solved than on the communication delay. Therefore, its convergence speed does not change as drastically as OCD. Hence, using a distributed infrastructure, ADMM achieves the fastest convergence with the partition of 9 areas due to the significantly reduced computation time on solving smaller subproblems. Comparing the two methods, OCD is more efficient with mild partitions of the systems while ADMM is more scalable if the system is partitioned into more areas. By comparing Fig. 5.9 with Fig. 5.7b, one can observe that the difference in the infrastructure has a more significant impact on the efficiency of OCD than of ADMM as communications dominates the execution time of OCD. Hence, it is important to incorporate the impact of communications to better determine which distributed method should be implemented subject to the communications capability of the system.

5.3.4 Application to a Large-Scale Network

To further evaluate the impact of communications infrastructure on distributed algorithms applied to real-world large-scale systems, we also conducted simulations on the Polish 2383-bus power system. The system is partitioned into 40 regions. Our previous studies show that ADMM is more suitable for solving large-scale problems on real power systems, therefore, we only consider using ADMM in this case. We say that ADMM has converged if all the constraint mismatches have fallen below 10^{-3} per unit. To solve the AC OPF problem, ADMM converges in 99 iterations with the total computation time of 149.5 seconds using the Tomlab SNOPT solver [86]. On average, each iteration of computation takes 1.51 seconds. Table 5.6 shows the communication delay per iteration and the total execution time of ADMM with 0%, 50% and 90% background traffic. Again, the communication delay using the distributed infrastructure is nearly one minute smaller than that using the centralized infrastructure under all traffic scenarios. Note that the computation time here is measured by running ADMM on a personal computer, which could be reduced further in practice if more powerful machines are used. As AC OPF is required to be solved by system operators every five minutes in less than one minute [2], the communications delay incurred by the centralized infrastructure makes it prohibitive for the deployment of distributed algorithms to solve the OPF problem.

Due to the significantly larger communication delay involved, the total execution time of ADMM using the centralized infrastructure is almost 1 minute longer than using the distributed infrastructure. Such 1-minute difference could result in the failure of reaching a dispatch solution if the AC OPF were to be run every 5 minutes by the system operators.

Table 5.6: Execution time of ADMM on Polish network with various background traffic.

Infrastructure	Communication delay(s)			Total execution time(s)		
	0%	50%	90%	0%	50%	90%
Centralized	0.741	0.765	0.903	222.8	225.2	238.9
Distributed	0.199	0.202	0.233	169.2	169.5	172.6

5.4 Discussion

The presented analysis shows that the centralized communications infrastructure imposes many restrictions on what distributed optimization method one can deploy and how many regions the system can have. For example, with many regions to be coordinated in a medium sized system, the centralized infrastructure could not support methods like OCD which requires lots of communications. And for large-scale systems with thousands of buses, the centralized infrastructure cannot support ADMM-type distributed optimization algorithms even though they require less information exchange. The reason for this is that the trade-offs between communications and computation is harder to handle in the centralized infrastructure, which makes it prohibitive for deploying distributed methods if the number of control regions is large or the information exchange is very frequent.

Our results clearly indicate that the distributed communications infrastructure is a more suitable and scalable infrastructure for distributed optimization applications, especially for large-scale power grids. However, the transition to a fully distributed infrastructure might not be straightforward as investments and a paradigm change in operational practices are necessary. The utility networks are likely to go through a transitional phase with the centralized SCADA system mostly in control while adding direct communication links among the substations, as illustrated by Fig. 5.10. Black squares denote ACCs that communicate with CCC, shaded squares denote ACCs that do not communicate with CCC, and the white square denotes the CCC. Red lines denote the communication links between CCC and ACC and blue lines denote direct communication links between ACCs. Note that the control and network management of hybrid networks is a complex topic and is beyond the scope of this thesis. However, the simulation method proposed in this chapter would be helpful for the evaluation of the benefits of distributed optimization during the transition phase.

One of the key objectives of this chapter is to provide a generic method that evaluates the communications network topology and scheme that should be used for distributed algorithms. In this chapter, we consider *synchronous distributed algorithms* where the communication and computation processes take turns. Therefore, a polling scheme can be deployed in the communications process. However, the proposed network models can be modified and extended to different distributed optimization schemes, network conditions, and the fields

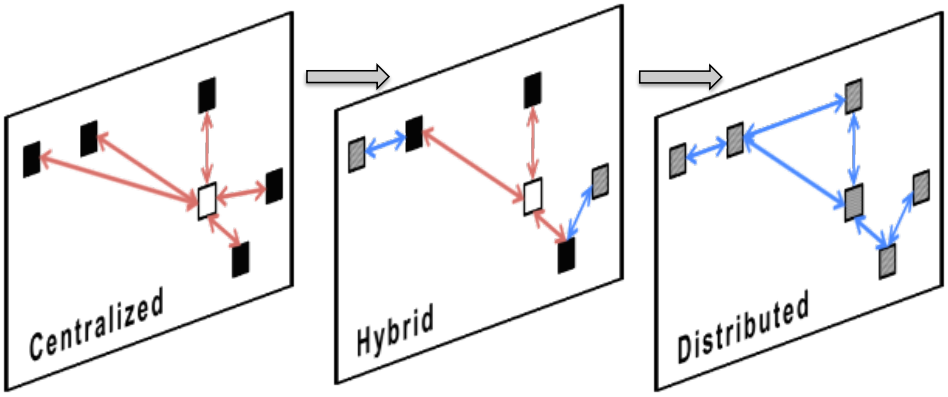


Figure 5.10: Transition from centralized to distributed infrastructure.

where distributed methods are intended to be deployed. For example, if *asynchronous distributed algorithms* are studied, then random access schemes need to be applied instead of polling. In distribution network management, the distributed algorithms can be deployed on devices located in a small geographical area. In this case, wireless communications is usually preferred, and the variance and distribution of the involved communication delays then become important metrics.

Furthermore, due to the use of optical fiber communication which is very reliable, the reliability of communication links is not a major concern in this study, which, however, could be an important factor to consider when one uses other communications technologies. Most distributed optimization algorithms including the two studied in this paper will fail to converge to a feasible solution if the information from one region is not accessible due to a broken communication path. Therefore, under the condition of unreliable communications, a fault-tolerant or fail-safe design of the network topology is required. On the other hand, more robust distributed optimization methods are preferable for such cases that are resilient to large communication delays or even failures of data delivery.

Chapter 6

Asynchronous Distributed ADMM

In this chapter, we propose a distributed asynchronous optimization approach which is based on the ADMM method. We extend the synchronous ADMM method used in previous chapters to fit into an asynchronous framework where a message-passing model is used and each worker is allowed to perform local updates with partial but not all updated information received from its neighbors. We will show that the proposed asynchronous ADMM algorithm asymptotically satisfies the first-order optimality conditions of problem (2.2), under the assumption of bounded delay of the worker and some other mild conditions on the objective function and constraints.

Then, we provide insights into the practical performance of the proposed distributed asynchronous ADMM method. Specifically, we identify three factors, namely, penalty parameter, system partitioning, and communication delay, that have significant impact on the convergence rate and solution quality of the proposed asynchronous ADMM method. Through extensive numerical simulations on the IEEE 118-bus test system, the Polish power system and the Great Britain power system, we provide quantitative analyses on how the performance of asynchronous ADMM is affected by these three factors.

6.1 Asynchronous ADMM

In Chapter 2.4.1, we have presented the reformulation of the AC OPF problem into the form of (2.29) such that one can apply the synchronous ADMM algorithm. To simplify the

analysis in this chapter, we slightly change the formulation (2.29) into the following form:

$$\underset{x,z}{\text{minimize}} \quad \sum_k F_k(x_k) \tag{6.1a}$$

$$\text{subject to} \quad A_k x_k = z_k, \ \forall k \tag{6.1b}$$

$$z_{k,l} = z_{l,k}, \ \forall (k,l) \in \mathcal{T}, \tag{6.1c}$$

where $F_k(x_k) = f_k(x_k) + \eta_{\mathcal{X}_k}(x_k)$ denotes the local objective function and $\eta_{\mathcal{X}}(\cdot)$ is the indicator function of set $\mathcal{X}$, with $\eta_{\mathcal{X}}(x) = 0$ if $x \in \mathcal{X}$ and $+\infty$ if $x \notin \mathcal{X}$. $\mathcal{T}$ is the set of inter-region tie lines, $z_{k,l} = \{(z^-_{i,j}, z^+_{i,j}) \mid i \in \mathcal{R}_k, j \in \mathcal{R}_l\}$ denote the boundary conditions that neighboring regions k and l should agree on, and $z_k = \{z_{k,l} | \forall l \in \mathcal{N}_k\}$ for each region k. Formulation (6.1) is equivalent to (2.29). We will also use 'worker' to denote the control entity that solves the local subproblem of each region. We only consider fixed penalty in the development and convergence analysis of asynchronous ADMM, but show how one can apply an increasing penalty approach later on. The Augmented Lagrangian function associated with (6.1) with fixed penalty is then given by

$$L(x, z, \lambda) = \sum_k \left\{ F_k(x_k) + \lambda_k^\top (A_k x_k - z_k) + \frac{\rho}{2} \|A_k x_k - z_k\|^2 \right\} + \eta_{\mathcal{Z}}(z). \tag{6.2}$$

Now, we extend the synchronous ADMM into an asynchronous setting where each worker determines when to perform local updates based on the messages it receives from neighbors. We say that a neighbor l has 'arrived' at worker k if the updated information of l is received by k. We assume partial asynchrony [56] where the delayed number of iterations of each worker is bounded by a finite number. We introduce a parameter p with $0 < p \leq 1$ to control the level of asynchrony. Worker k will update its local variables after it receives new information from at least $\lceil p|\mathcal{N}_k| \rceil$ neighbors with $|\mathcal{N}_k|$ denoting the number of neighbors of worker k. In the worst case, any worker should wait for at least one neighbor because otherwise its local update will make no progress as it has no new information.

Figure 6.1 illustrates the proposed asynchronous scheme by assuming three workers, each connecting to the other two workers. The blue bar denotes the local computation and the grey line denotes the information passing among workers. The red dotted line marks the count of iterations, which will be explained in Chapter 6.2. As shown in Fig. 6.1a, synchronous ADMM can be implemented by setting $p = 1$, where each worker does not

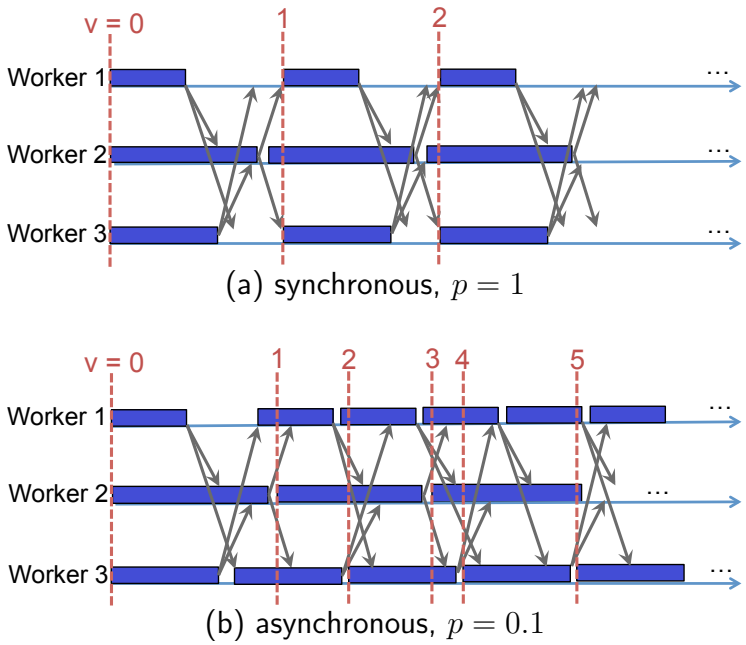


Figure 6.1: Illustration of synchronous and asynchronous distributed ADMM.

perform local computation until all neighbors arrive. Figure 6.1b shows an asynchronous case where each worker can perform its local update with only one neighbor arrived, which could reduce the waiting time for fast workers. In the rest of this chapter, we do not specify the setting of p but only consider it to be a very small value such that each worker can perform its local update as long as it received information from *at least one* neighbor, which indeed represents the highest level of asynchrony.

Algorithm 3 presents the asynchronous ADMM approach from each region's perspective with ν_k denoting the local iteration counter. The x-update and λ-update are similar to those in synchronous ADMM presented in Algorithm 3. However, for the z-update, each worker k does not need to wait for updated information from all neighbors, which is the major difference between the synchronous and asynchronous schemes. Also different from synchronous ADMM, for the z-update, we add a proximal term $\frac{\alpha}{2}\|z_k - z_k^{\nu_k}\|^2$ with $\alpha \geq 0$ which is a sufficient condition for proving the convergence of ADMM under asynchrony. The intuition of adding this proximal term is to reduce the stepsize of updating variables to offset the error brought by asynchrony. Also, in the z-update, only the entries corresponding to arrived neighbors in z_k are updated.

Algorithm 3 Asynchronous ADMM in region k

1: **Initialization**

Given x_k^0, set $\lambda_k^0 = 0$, $\rho = \rho_0$, $\nu_k = 0, z_k^0 \in \mathcal{Z}$

2: **Repeat**

3: **Update** x by

$$x_k^{\nu_k+1} = \underset{x_k}{\operatorname{argmin}}\ L(x_k, z_k^{\nu_k}, \lambda_k^{\nu_k}) \tag{6.3}$$

4: **Update** λ using

$$\lambda_k^{\nu_k+1} = \lambda_k^{\nu_k} + \rho(A_k x_k^{\nu_k+1} - z_k^{\nu_k}) \tag{6.4}$$

5: Send $\{A_{k,l}x_k^{\nu_k+1}, \lambda_{k,l}^{\nu_k+1}\}$ to region l, $\forall l \in \mathcal{N}_k$

6: **Repeat**

7: **Wait** until at least $\lceil p|\mathcal{N}_k|\rceil$ neighbors arrive

8: **Update** z_k associated with arrived neighbors

$$z_k^{\nu_k+1} = \underset{z_k}{\operatorname{argmin}}\ \ L(x_k^{\nu_k}, z_k, \lambda_k^{\nu_k}) + \frac{\alpha}{2}\|z_k - z_k^{\nu_k}\|^2 \tag{6.5}$$

9: Set $\nu_k \leftarrow \nu_k + 1$

10: **Until** a predefined stopping criterion is satisfied

6.2 Convergence Analysis

6.2.1 Main Theorem

For analyzing the convergence property of Algorithm 3, we introduce a global iteration counter ν and present Algorithm 3 from a global point of view in Algorithm 4 *as if* there is a master node monitoring all the local updates. Note that such master node is not needed for the implementation of Algorithm 3 and the global counter only serves the purpose of slicing the execution time of Algorithm 3 for analyzing the changes in variables during each time slot. We use the following rules to set the counter ν: 1) ν can be increased by 1 at time $t_{\nu+1}$ when some worker is ready to start its local x-update; 2) the time period $(t_\nu, t_{\nu+1}]$ should be as long as possible; 3) there is no worker that finishes x-update more than once in $(t_\nu, t_{\nu+1}]$; 4) ν should be increased by 1 before any worker receives new information after it has started its x-update. The third rule ensures that each x-update is captured in one individual iteration and the fourth rule ensures that the z used for any x-update during

Algorithm 4 Asynchronous ADMM from a global view

1: **Initialization**

Given x^0, ρ, set $\lambda^0 = 0$, $\nu = 0, z^0 \in \mathcal{Z}$

2: **Repeat**

3: **Update**

$$x_k^{\nu+1} = \begin{cases} \text{argmin } F_k(x_k) + \lambda_k^{\nu\top} A_k x_k \\ \qquad + \dfrac{\rho}{2}\|A_k x_k - z_k^{\bar{\nu}_k+1}\|^2 & \text{if } k \in \mathcal{A}_\nu \\ x_k^\nu & \text{otherwise} \end{cases} \tag{6.6}$$

$$\lambda_k^{\nu+1} = \begin{cases} \lambda_k^\nu + \rho(A_k x_k^{\nu+1} - z_k^{\bar{\nu}_k+1}) & \text{if } k \in \mathcal{A}_\nu \\ \lambda_k^\nu & \text{otherwise} \end{cases} \tag{6.7}$$

For $\forall k \in \mathcal{A}_\nu, l \in \mathcal{N}_k$ and $\forall (l,k) \in \mathcal{T}_{(t_{\nu-1},t_\nu]}$:

$$z_{k,l}^{\nu+1} = \left(\lambda_{k,l}^{\nu+1} + \lambda_{l,k}^{\nu+1} + \rho A_{k,l} x_k^{\nu+1} + \rho A_{l,k} x_l^{\nu+1} + \alpha z_{k,l}^\nu\right)/(2\rho + \alpha) \tag{6.8}$$

4: Set $\nu \leftarrow \nu + 1$

5: **Until** a predefined stopping criterion is satisfied

$(t_\nu, t_{\nu+1}]$ is equal to the z measured at $t_{\nu+1}$. This global iteration counter is represented by red dotted lines in Fig. 6.1.

We define $\mathcal{A}_\nu$ as the index subset of workers who finishes x-updates during the time $(t_\nu, t_{\nu+1}]$ with $0 \leq |\mathcal{A}_\nu| \leq K$. Note that with $|\mathcal{A}_\nu| = K$, Algorithm 4 is equivalent to synchronous ADMM. We use $\mathcal{U}_{(t_{\nu-1},t_\nu]}$ to denote the set of workers that exchange information at iteration ν; i.e., $(k,l) \in \mathcal{U}_{(t_{\nu-1},t_\nu]}$ denotes that the updated information from worker k arrives at l during the time $(t_{\nu-1}, t_\nu]$.

Now, we formally introduce the assumption of partial asynchrony (bounded delay).

Assumption 1 *Let $\omega > 0$ be a maximum number of global iterations between the two consecutive x-updates for any worker k; i.e., for all k and global iteration $\nu > 0$, it must hold that $k \in \mathcal{A}_\nu \cup \mathcal{A}_{\nu-1} \cdots \cup \mathcal{A}_{\max\{\nu-\omega+1,0\}}$*

Define $\bar{\nu}_k$ as the iteration number at the start of the x-update that finishes at iteration ν. Then, under Assumption 1 and due to the fact that any worker can only start a new

x-update after it has finished its last x-update, it must hold that

$$\max\{\nu-\omega,0\}\leq\bar{\nu}_k<\nu,\quad\forall\nu>0. \tag{6.9}$$

The z-update (6.8) is derived from the optimality condition of (6.30). As an example, we show how to update variable $z_{k,l}$ (same for $z_{l,k}$) for one pair of neighboring workers k and l. To fulfill $z_{k,l}=z_{l,k}$, we substitute $z_{l,k}$ with $z_{k,l}$ and then remove the part $\eta_{\mathcal{Z}}(z)$ from (6.2). The remaining part that contains $z_{k,l}$ in (6.2) can be written as:

$$\begin{aligned}&L'(x_k^{\nu+1},z_{k,l},\lambda_k^{\nu+1})\\&=-(\lambda_{k,l}^{\nu+1}+\lambda_{l,k}^{\nu+1})z_{k,l}+\frac{\rho}{2}\|A_{k,l}x_k^{\nu+1}-z_{k,l}\|^2\\&\quad+\frac{\rho}{2}\|A_{l,k}x_l^{\nu+1}-z_{k,l}\|^2+\frac{\alpha}{2}\|z_{k,l}-z_{k,l}^{\nu}\|^2\end{aligned} \tag{6.10}$$

The optimality condition of (6.30) then yields

$$\lambda_{k,l}^{\nu+1}+\lambda_{l,k}^{\nu+1}+\rho(A_{k,l}x_k^{\nu+1}-z_{k,l}^{\nu+1})+\rho(A_{l,k}x_l^{\nu+1}-z_{k,l}^{\nu+1})-\alpha(z_{k,l}^{\nu+1}-z_{k,l}^{\nu})=0, \tag{6.11}$$

which results in (6.8). Thereby, worker k will update z_k locally once 1) it receives λ_l, ρ_l and $A_{l,k}x_l$ from $\forall l\in\mathcal{N}_k$ or 2) it finishes local x and λ updates.

Before we state our main results of convergence analysis, we need to introduce the following definitions and make the following assumptions with respect to problem (6.1).

Definition 1 (Restricted prox-regularity) *[73] Let $D\in\mathbb{R}_+$, $f:\mathbb{R}^N\to\mathbb{R}\cup\{\infty\}$, and define the exclusion set*

$$S_D:=\{x\in dom(f):\|d\|>D\ \ for\ all\ \ d\in\partial f(x)\}. \tag{6.12}$$

f is called restricted prox-regular *if, for any $D>0$ and bounded set $T\subseteq dom(f)$, there exists $\gamma>0$ such that*

$$\begin{aligned}&f(y)+\frac{\gamma}{2}\|x-y\|^2\geq f(x)+\langle d,y-x\rangle,\\&\forall x\in T\setminus S_D,y\in T,d\in\partial f(x),\|d\|\leq D.\end{aligned} \tag{6.13}$$

Definition 2 (Strongly convex functions) *A convex function f is called strongly convex with modulus σ if either of the following holds:*

1. *there exists a constant* $\sigma > 0$ *such that the function* $f(x) - \frac{\sigma}{2}\|x\|^2$ *is convex;*

2. *there exists a constant* $\sigma > 0$ *such that for any* $x, y \in \mathbb{R}^N$ *we have:*

$$f(y) \geq f(x) + \langle f'(x), y - x\rangle + \frac{\sigma}{2}\|y - x\|^2. \tag{6.14}$$

The following assumptions state the desired characteristics of the objective functions and constraints.

Assumption 2 *$\mathcal{X}$ is a compact smooth manifold and there exists constant $M_2 > 1$ such that, $\forall \nu_1, \nu_2$*

$$\frac{1}{M_2}\|A_k x_k^{\nu_1} - A_k x_k^{\nu_2}\| \leq \|x_k^{\nu_1} - x_k^{\nu_2}\| \leq M_2\|A_k x_k^{\nu_1} - A_k x_k^{\nu_2}\|.$$

Assumption 2 allows A_k to not have full column rank, which is more realistic for region-based optimization applications. Since $\mathcal{X}$ is compact and $\|A_k\| < \infty$, Assumption 2 is satisfied.

Assumption 3 *Each function F_k is restricted prox-regular (Definition 1) and its subgradient ∂F_k is Lipschitz continuous with a Lipschitz constant $M_1 > 0$.*

The objective function F_k in problem (6.1) includes indicator functions whose boundary is defined by $\mathcal{X}$. Recall that we assume $\mathcal{X}$ is compact and smooth and as stated in [73], indicator functions of compact smooth manifolds are restricted prox-regular functions.

Assumption 4 *The subproblem (6.3) is feasible and a local minimum $x_k \in \mathcal{X}_k$ is found at each x-update.*

Assumption 4 can be satisfied if the subproblem is not ill-conditioned and the solver used to solve the subproblem is sufficiently robust to identify a local optimum, which is generally the case observed from our empirical studies.

Assumption 5 *$A_k A_k^\top$ is invertible for all k, and define $B_k = (A_k A_k^\top)^{-1} A_k$. Also, let $\sigma_{\max}(\cdot)$ denote the operator of taking the largest eigenvalue of a symmetric matrix and define $C = \max\{\sigma_{\max}(B_k^\top B_k), \forall k\}$.*

Assumption 6 *λ is bounded, and*

$$-\infty < L(x^\nu, z^\nu, \lambda^\nu) < \infty, \text{if } \ x \in \mathcal{X}$$

λ can be bounded by the projection onto a compact box, i.e., $\lambda^\nu \leftarrow \max(\lambda^{\min}, \min(\lambda^\nu, \lambda^{\max}))$. Then Assumption 6 holds as all the terms in $L(x^\nu, z^\nu, \lambda^\nu)$ are bounded with x in the compact feasible region.

The main convergence result of asynchronous ADMM is stated below.

Theorem 2 *Suppose that Assumptions 1 to 6 hold. Moreover, choose*

$$\begin{aligned} &\rho > (\gamma + CM_1^2)M_2^2 + \sqrt{(\gamma + CM_1^2)^2 M_2^4 + 4CM_1^2 M_2^2}, \\ &\alpha > \frac{(2\rho M_2^4 + 1)(\omega - 1)^2}{2} - \rho. \end{aligned} \tag{6.15}$$

Then, $(\{x_k^\nu\}_{k=1}^K, \{z_k^\nu\}_{k=1}^K, \{\lambda_k^\nu\}_{k=1}^K)$ generated by (6.3) to (6.5) (or equivalently (6.6) to (6.8)) are bounded and have limit points that satisfy the KKT conditions of problem (6.1) for local optimality.

6.2.2 Proof of Theorem 2

The essence of proving Theorem 2 is to show the sufficient descent of the Augmented Lagrangian function (6.2) at each iteration and that the difference of (6.2) between two consecutive iterations is summable. The proof of Theorem 2 uses the following lemmas, which are proved in the Appendix C and D, respectively.

Lemma 1 *Suppose that Assumption 2 to 5 hold. Then it holds that*

$$\begin{aligned} &L(x^{\nu+1}, z^{\nu+1}, \lambda^{\nu+1}) - L(x^\nu, z^\nu, \lambda^\nu) \\ &\leq (\frac{\gamma + CM_1^2}{2} - \frac{\rho}{4M_2^2} + \frac{CM_1^2}{\rho}) \sum_{k \in \mathcal{A}_\nu} \|x_k^{\nu+1} - x_k^\nu\|^2 \\ &\quad - (\rho + \alpha)\|z^{\nu+1} - z^\nu\|^2 + \frac{2\rho M_2^4 + 1}{2} \sum_{k \in \mathcal{A}_\nu} \|z_k^{\bar{\nu}_k + 1} - z_k^\nu\|^2. \end{aligned} \tag{6.16}$$

Due to the term $\|z_k^{\bar{\nu}_k+1} - z_k^\nu\|^2$ which is caused by the asynchrony of the updates among workers, (1) is not necessarily decreasing. We bound this term by Lemma 2.

Lemma 2 *Suppose that Assumption 1 holds. Then it holds that*

$$\sum_{\phi=1}^{\nu}\sum_{k\in\mathcal{A}_\phi}\|z_k^{\bar{\phi}_k+1}-z_k^{\phi}\|^2\leq 2(\omega-1)^2\sum_{\phi=1}^{\nu}\|z^{\phi+1}-z^{\phi}\|^2. \tag{6.17}$$

Using Lemma 1 and Lemma 2, we now prove Theorem 2.

Proof of Theorem 2. Any KKT point $(\{x_k^\star\}_{k=1}^K, z^\star, \{\lambda_k^\star\}_{k=1}^K)$ of problem (6.1) should satisfy the following conditions

$$\partial F_k(x_k^\star)+A_k^\top\lambda_k^\star=\mathbf{0},\quad\forall k \tag{6.18a}$$

$$\lambda_{k,l}^\star+\lambda_{l,k}^\star=\mathbf{0},\quad\forall(k,l)\in\mathcal{T} \tag{6.18b}$$

$$A_kx_k^\star-z_k^\star=\mathbf{0},\quad\forall k \tag{6.18c}$$

By taking the telescoping sum of (6.16), we obtain

$$\begin{aligned}&\Big(\frac{\rho}{4M_2^2}-\frac{\gamma+CM_1^2}{2}-\frac{CM_1^2}{\rho}\Big)\sum_{\phi=1}^{\nu}\sum_{k\in\mathcal{A}_\phi}\|x_k^{\phi+1}-x_k^{\phi}\|^2\\&\quad+(\rho+\alpha)\sum_{\phi=1}^{\nu}\|z^{\phi+1}-z^{\phi}\|^2-\frac{2\rho M_2^4+1}{2}\sum_{\phi=1}^{\nu}\sum_{k\in\mathcal{A}_\phi}\|z_k^{\bar{\phi}_k+1}-z_k^{\phi}\|^2\\&\leq L(x^1,z^1,\lambda^1)-L(x^{\nu+1},z^{\nu+1},\lambda^{\nu+1})<\infty,\end{aligned} \tag{6.19}$$

where the last inequality holds under Assumption 6.

By substituting (6.17) in Lemma (2) into (6.19), we have

$$\begin{aligned}&\left(\frac{\rho}{4M_2^2}-\frac{\gamma+CM_1^2}{2}-\frac{CM_1^2}{\rho}\right)\sum_{\phi=1}^{\nu}\sum_{k\in\mathcal{A}_\phi}\|x_k^{\phi+1}-x_k^{\phi}\|^2\\&+\left(\rho+\alpha-\frac{(2\rho M_2^4+1)(\omega-1)^2}{2}\right)\sum_{\phi=1}^{\nu}\|z^{\phi+1}-z^{\phi}\|^2<\infty,\end{aligned} \tag{6.20}$$

Then by choosing ρ and α as in (6.15), the left-hand-side (LHS) of (6.20) is positive and increasing with ν. Since the right-hand-side (RHS) of (6.20) is finite, we must have as $\nu\to\infty$,

$$x_k^{\nu+1}-x_k^{\nu}\to\mathbf{0},\quad z_k^{\nu+1}-z_k^{\nu}\to\mathbf{0},\quad\forall k. \tag{6.21}$$

Given (6.21) and (C.12), we also have

$$\lambda_k^{\nu+1}-\lambda_k^{\nu}\to\mathbf{0},\quad\forall k \tag{6.22}$$

Since $\mathcal{X}$ and $\mathcal{Z}$ are both compact and $x \in \mathcal{X}$ and $z \in \mathcal{Z}$, and λ is bounded by projection, $(\{x_k^\star\}_{k=1}^K, \{z_k^\star\}_{k=1}^K, \{\lambda_k^\star\}_{k=1}^K)$ is bounded and has a limit point. Finally, we show that every limit point of the above sequence is a KKT point of problem (6.1); i.e., it satisfies (6.18).

For $k \in \mathcal{A}_\nu$, by applying (6.22) to (6.7) and by (6.21), we obtain

$$A_k x_k^{\nu+1} - z_k^{\nu+1} \to \mathbf{0}, \quad k \in \mathcal{A}_\nu. \tag{6.23}$$

For $k \notin \mathcal{A}_\nu$, let $\bar{\nu}_k$ denote the iteration number of region k's last update, then $k \in \mathcal{A}_{\bar{\nu}_k}$. Then at iteration $\bar{\nu}_k$, we have

$$\lambda_k^{\bar{\nu}_k+1} = \lambda_k^{\bar{\nu}_k} + \rho(A_k x_k^{\bar{\nu}_k+1} - z_k^{\overline{(\bar{\nu}_k)}_k}), \tag{6.24}$$

where $\overline{(\bar{\nu}_k)}_k$ denotes the iteration of k's update before $\bar{\nu}_k$. And since $x_k^{\nu+1} = x_k^{\nu} = \cdots = x_k^{\bar{\nu}_k+2} = x_k^{\bar{\nu}_k+1}$, and by (6.21) and (6.22), we have

$$\begin{aligned}
&\|A_k x_k^{\nu+1} - z_k^{\nu+1}\| \\
&= \|A_k x_k^{\bar{\nu}_k+1} - z_k^{\nu+1}\| \\
&= \|A_k x_k^{\bar{\nu}_k+1} - z_k^{\overline{(\bar{\nu}_k)}_k+1} + z_k^{\overline{(\bar{\nu}_k)}_k+1} - z_k^{\nu+1}\| \\
&\leq \frac{1}{\rho}\|\lambda_k^{\bar{\nu}_k+1} - \lambda_k^{\bar{\nu}_k}\| + \|z_k^{\overline{(\bar{\nu}_k)}_k+1} - z_k^{\nu+1}\| \to \mathbf{0}.
\end{aligned} \tag{6.25}$$

Therefore, we can conclude

$$A_k x_k^{\nu+1} - z_k^{\nu+1} \to \mathbf{0}, \forall k; \tag{6.26}$$

i.e., the KKT condition (6.18c) can be satisfied asymptotically.

For any $z_{[ij]}^{\nu+1}$, where $i \in \mathcal{R}_k$, $j \in \mathcal{R}_l$, $(i,j) \in \mathcal{T}$, and $k \in \mathcal{A}_\nu$, the optimality condition of (6.8) yields (6.11). Since the last three terms on the LHS of (6.11) will asymptotically converge to $\mathbf{0}$ due to (6.26) and (6.21), the KKT condition (6.18b) can be satisfied asymptotically. At last, by applying (6.21) and (6.22) to (C.10), we obtain KKT condition (6.18a). Therefore, we can conclude that $(\{x_k^\nu\}_{k=1}^K, \{z_k^\nu\}_{k=1}^K, \{\lambda_k^\nu\}_{k=1}^K)$ are bounded and converge to the set of KKT points of problem (6.1). □

6.2.3 Simulation and Discussion

The simulations are conducted using two IEEE standard test systems and two large-scale transmission networks. The system configuration and the parameter settings are given in

Table 6.1: Test system configuration and parameter setting for asynchronous ADMM.

System	IEEE-30	IEEE-118	Polish	Great Britain (GB)
buses	30	118	2383	2224
regions	4	8	40	70
initialization	flat	flat	warm	warm
ρ	5×10^4	5×10^5	10^8	10^9

Table 6.1. The partitions of the systems are derived using the proposed spectral partitioning approach that reduces the coupling among regions. A "flat" start initializes x to be at the median of its upper and lower bounds, while a "warm" start is a feasible solution to the power flow equations (2.1b).

Algorithm 3 is conducted in Matlab with p set to 0.1 which simulates the worst case where each worker is allowed to perform a local update with one arrived neighbor. The stopping criterion is that the maximum residue ($\max\{\Gamma_k\}, \forall k$) and constraint mismatch are both smaller than 10^{-3} p.u. We use number of average local iterations and the execution time to measure the performance of Algorithm 3. The execution time records the total time Algorithm 3 takes until convergence including the computation time (measured by CPU time) and the waiting time for neighbors. Here, the waiting time also includes the communication delay, which is estimated by assuming that fiber optical communications is used and exploiting the distributed communications infrastructure with dedicated links proposed in Chapter 5. Therefore, passing message from one worker to the other usually takes a couple of milliseconds, which is very small compared to local computation time.

Figure 6.2 shows the convergence of the maximum residue of the proposed asynchronous ADMM, i.e., Algorithm 3, and its synchronous counterpart with p set to 1, when solving the OPF problem for the considered four test systems. We set α to zero for this experiment and will show later that using a large α is not necessary for the convergence of asynchronous ADMM. As expected, synchronous ADMM takes fewer iterations to converge, especially on large-scale systems, while asynchronous ADMM requires more iterations as each worker keeps performing local computation as long as it receives any new information from neighbors. This result also indicates that asynchronous ADMM injects more communications traffic into the wide area network that connects area control centers as messages are ex-

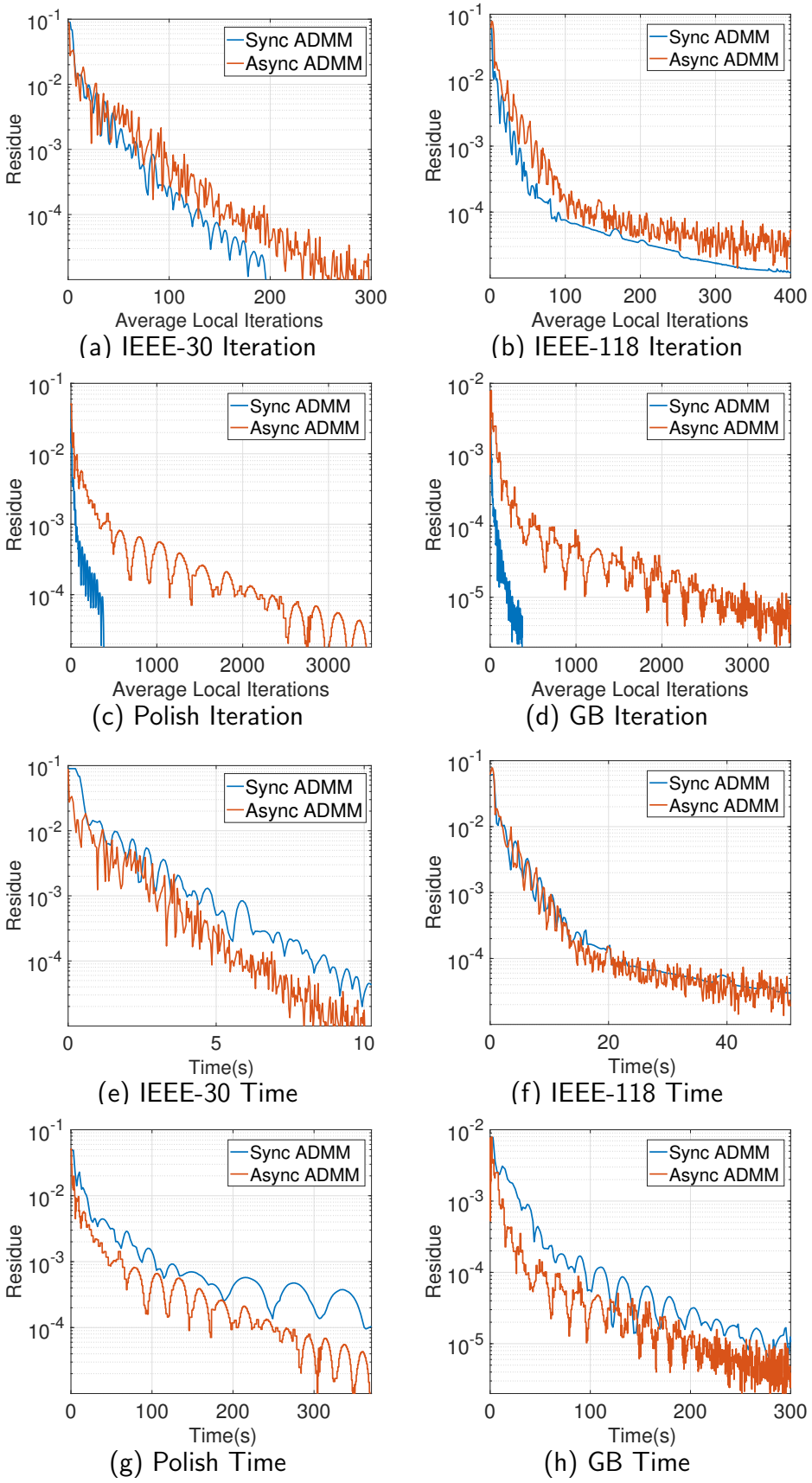


Figure 6.2: Convergence of residue of synchronous and asynchronous ADMM.

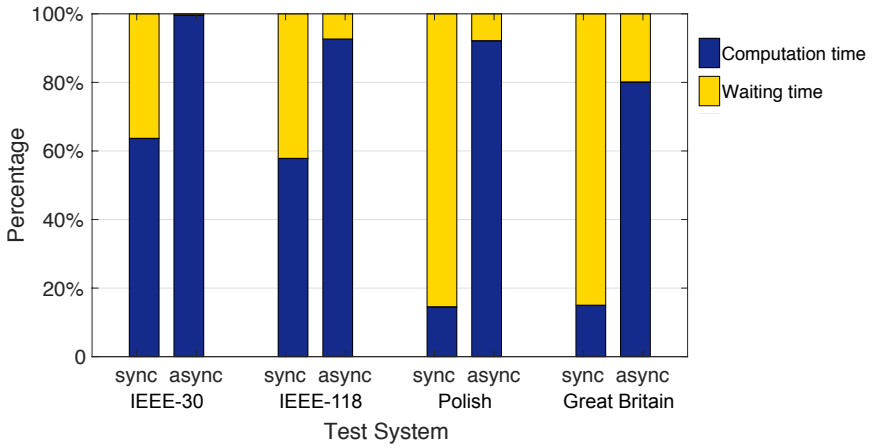


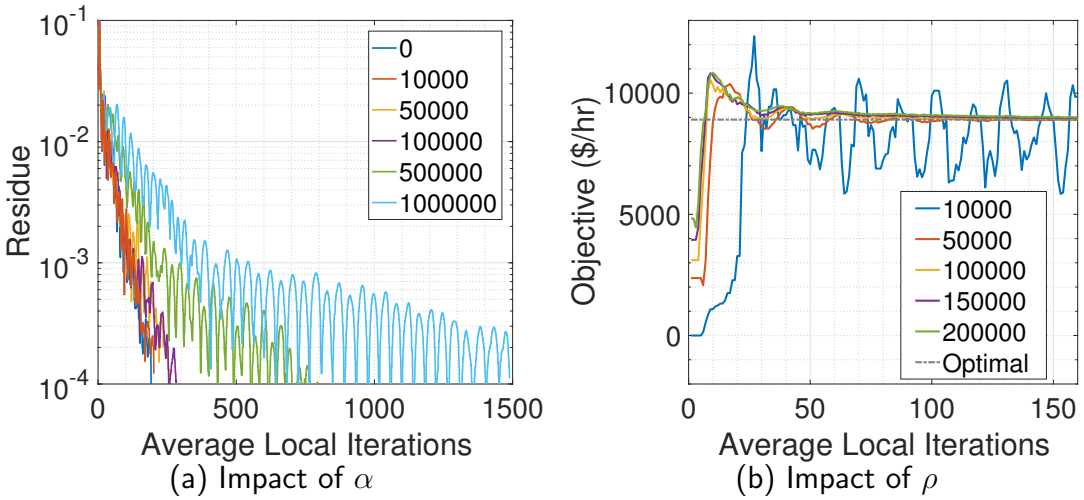
Figure 6.3: Computation/waiting time for synchronous ADMM and asynchronous ADMM on different test systems.

changed every local iteration. Later on, we will show that the number of iterations can be reduced substantially by using an increasing penalty parameter, therefore the increased communications of asynchronous ADMM could be supported if fast fibre optic communications links are used and many direct communications links between ACCs are built. In terms of the execution time of both schemes, synchronous ADMM can be slower than asynchronous ADMM on large-scale networks due to the waiting time for the slowest worker at each iteration. Figure 6.3 illustrates the percentage of the average computation and waiting time experienced by all workers. It is clearly shown that a lot of time is wasted on waiting for all neighbors using a synchronous scheme.

To measure the optimality of the solution found by asynchronous ADMM, we also calculate the gap in the objective value achieved by synchronous and asynchronous ADMM with respect to the objective value obtained by a centralized method when the maximum residue Γ_k of all workers reaches 10^{-3}. This gap is shown in Table 6.2 which is fairly small for all the considered systems with both schemes. A solution can be considered of good quality if this gap is smaller than 1%. Surprisingly, asynchronous ADMM achieves a slightly smaller gap compared with synchronous ADMM. This is due to the fact that in asynchronous ADMM a worker uses the most updated information of its neighbors in a more timely manner than in the synchronous case and updates its local variables more frequently, which, as a trade-off, results in more local iterations especially on large systems. Note that for both schemes, this

Table 6.2: Gap of objective function achieved by synchronous and asynchronous ADMM.

Scheme	IEEE-30	IEEE-118	Polish	Great Britain (GB)
synchronous	0.025%	0.122%	0.060%	0.575%
asynchronous	0.005%	0.098%	0.031%	0.416%



(a) Impact of α (b) Impact of ρ

Figure 6.4: Impact of parameters on the convergence of asynchronous ADMM.

gap should asymptotically approach zero. However, for many engineering applications, only a mild level of accuracy is needed. Therefore ADMM is usually terminated when it reaches the required accuracy with a suboptimal solution. These results indeed validate that asynchronous ADMM could find a good solution faster than its synchronous counterpart and is more fault-tolerable for delayed or missing information. But we should mention the large number of iterations taken by asynchronous ADMM incurs more communications load because updated information is sent at each iteration. And this might raise more requirements on the communications system used for deploying distributed algorithms.

Now, we evaluate the impact of parameter values on the performance of asynchronous ADMM using the IEEE 30-bus system. Interestingly, as shown in Fig. 6.4a, even though a large α is a sufficient condition for asynchronous ADMM to converge, it is not necessary in practice. This observation is consistent with the observation made in [75] as the proof there and our proof are both derived for the worst case. In fact, the purpose of using α is to make sure that the Augmented Lagrangian (6.2) decreases at each iteration which may not

be the case due to the existence of the term $\rho \sum_{k \in \mathcal{A}_\nu} (z_k^{\bar{\nu}_k+1} - z_k^\nu)^\top (A_k x_k^{\nu+1} - A_k x_k^\nu)$ in (C.9). However, as z_k is generally a value between $A_k x_k$ and $A_l x_l, \forall l \in \mathcal{N}_k$ that worker k and l try to approach, $(z_k^{\bar{\nu}_k+1} - z_k^\nu)^\top (A_k x_k^{\nu+1} - A_k x_k^\nu)$ is likely to be a negative value, which makes α unnecessary. In fact, as shown in Fig. 6.4a, a large α will slow down the convergence as the proximal term in (6.5) forces local updates to take very small steps. Finally, Fig. 6.4b shows that a large ρ is indeed necessary for the convergence of asynchronous ADMM. With a larger ρ, asynchronous ADMM tends to stabilize around the final solution more quickly, which, however, may lead to a slightly less optimal solution.

6.3 Parameter Tuning

In Chapter 6.1, we have proposed an asynchronous ADMM approach and established its convergence for the partition-based distributed optimization problem in Chapter 6.2. However, we have not investigated what factors affect the *convergence rate* of asynchronous ADMM. In fact, the performance of distributed methods is strongly dependent on the problem of interest, which calls for empirical work to characterize their practical performance for various power system optimization problems [15].

Therefore, in this section and the two subsequent sections, we aim to provide practical guidance on how to better implement the ADMM algorithm in large-scale transmission networks in a distributed and asynchronous manner. By studying the impact of algorithm parameter, we make suggestions on how to tune the penalty parameter based on the feasibility and optimality requirements especially for large-scale networks. By studying the impact of communication delays, we help the utilities identify whether their existing communications infrastructure could support asynchronous distributed optimization algorithms. And by studying the impact of system partitioning, we further validate the importance of partitioning as a good partition generally leads to higher optimality of the solution and faster convergence.

6.3.1 Asynchronous ADMM with Increasing Penalty

While fixed penalty guarantees the convergence of asynchronous ADMM to a local optimum of problem (6.1), it can be very slow when applied to large-scale power systems. For example, as shown in Fig. 6.3, Algorithm 3 takes thousands of iterations and hundreds of seconds to converge. Therefore, to speed up the convergence of Algorithm 3, we consider the penalty update schemes used in the synchronous ADMM method, i.e., Algorithm 1, and integrate them into the asynchronous Algorithm 3. This improved asynchronous ADMM algorithm is presented in Algorithm 5 with ν_k denoting the local iteration counter. Note that in Algorithm 5, we replace the fixed penalty ρ in the Lagrangian function (6.2) with local penalty ρ_k which can vary across regions. Again, $\Gamma_k^{\nu_k+1} = \|A_k x_k^{\nu_k+1} - z_k^{\nu_k+1}\|_\infty$ is defined as the local primal residue which can be treated as a measure for feasibility [5]. The update of penalty involves

Algorithm 5 Asynchronous ADMM in region k with increasing penalty

1: **Initialization**

Given x_k^0, set $\lambda_k^0 = 0$, $\rho_k = \rho_0$, $\nu_k = 0, z_k^0 \in \mathcal{Z}$

2: **Repeat**

3: **Update** x_k by

$$x_k^{\nu_k+1} = \underset{x_k}{\operatorname{argmin}}\ L_{\rho_k}(x_k, z_k^{\nu_k}, \lambda_k^{\nu_k}) \tag{6.27}$$

4: **Update** λ_k by

$$\lambda_k^{\nu_k+1} = \lambda_k^{\nu_k} + \rho_k^{\nu_k}(A_k x_k^{\nu_k+1} - z_k^{\nu_k}) \tag{6.28}$$

5: **Update** ρ_k by

$$\rho_k^{\nu_k+1} = \begin{cases} \rho_k^{\nu_k} & \text{if } \Gamma_k^{\nu_k+1} \leq \xi \Gamma_k^{\nu_k} \\ \tau \rho_k^{\nu_k} & \text{otherwise} \end{cases} \tag{6.29}$$

with constants $0 < \xi < 1$ and $\tau \geq 1$.

6: Send $\{A_{k,l} x_k^{\nu_k+1}, \lambda_{k,l}^{\nu_k+1}, \rho_k^{\nu_k+1}\}$ to region l, $\forall l \in \mathcal{N}_k$

7: **Repeat**

8: **Wait** until receiving message from at least 1 neighbor

9: **Update** z_k associated with arrived neighbors

$$z_k^{\nu_k+1} = \underset{z_k}{\operatorname{argmin}}\ \ L_{\rho_k}(x_k^{\nu_k}, z_k, \lambda_k^{\nu_k}) + \frac{\alpha}{2}\|z_k - z_k^{\nu_k}\|^2 \tag{6.30}$$

10: **Choose** ρ_k to deploy in next iteration of x-update

$$\rho_k^{\nu_k+1} = \max\{\rho_l^{\nu_k+1}, \forall l \in \{k\} \cup \mathcal{N}_k\} \tag{6.31}$$

11: Set $\nu_k \leftarrow \nu_k + 1$

12: **Until** a predefined stopping criterion is satisfied

two steps. First, after local x and λ updates, the local penalty ρ_k should be increased if the local primal residue does not decrease sufficiently in the most recent local computation, as shown in (6.29). Then, after receiving the penalty parameters from neighbors, the maximum penalty used by the neighbors and the region itself will be chosen to be used in the next local iteration to mimic the penalty increasing method used in centralized optimization [10], as

shown in (6.31). For a variable belonging to neighboring regions such as $\tilde{\rho}_l$, its subscription $\nu_k + 1$ denotes its most recent value received by region k at or before iteration $\nu_k + 1$. Note that in Algorithm 5 we consider the worst case where each worker only waits for at least one neighbor before each local update. The rest of Algorithm 5 is similar to Algorithm 3.

6.3.2 Convergence Analysis

The asynchronous ADMM method with fixed penalty parameter is shown to converge under the assumptions of bounded delay, bounded primal and dual variables, smoothness of the feasible region, regularity of the objective function and the existence of a local minimum at local x-updates. With an increasing penalty parameter, an additional assumption for the convergence of Algorithm 5 is that all ρ_k's should be upper-bounded. As the convergence of Algorithm 5 can be proved following the same procedure as discussed in details in Chapter 6.2, we refer to the assumptions made in Chapter 6.2 and only state the main convergence result of Algorithm 5 below:

Theorem 3 *Suppose that Assumptions 1 to 6 in Chapter 6.2 hold. Additionally, the penalty parameter ρ is updated by (6.29) and (6.31) with $\rho_0 \leq \rho_k^{\nu_k} \leq \rho_{\max}, \forall k, \nu_k$. Moreover, assume*

$$\begin{aligned} \rho_0 &> (\gamma + CM_1^2)M_2^2 + \sqrt{(\gamma + CM_1^2)M_2^2 + 2CM_1^2M_2^2(\tau^{\omega-1}+1)}, \\ \alpha &> \frac{\big((2M_2^4 + \tau^{\omega-1} - 1)\rho_{\max} + 1\big)(\omega-1)^2}{2} - \rho_0. \end{aligned} \tag{6.32}$$

Then, $(\{x_k^{\nu_k}\}_{k=1}^K, \{z_k^{\nu_k}\}_{k=1}^K, \{\lambda_k^{\nu_k}\}_{k=1}^K)$ generated by Algorithm 5 are bounded and have limit points that satisfy the KKT conditions of problem (6.1).

The notations in Theorem 3 can be found in the assumptions stated in Chapter 6.2. Theorem 3 states that the penalty parameter ρ and the weight for the proximal term α should be set to be sufficiently large. However, through numerical simulations, we observed that the proximal term is not necessary for convergence of Theorem 3, which we drop in the subsequent numerical experiments. Here, we would like to elaborate more on the role that the penalty parameter ρ plays in the convergence of Algorithm 5. According to the update rule (6.29), an increasing penalty enforces sufficient decrease of the primal residue. In other words, by increasing the penalty, Algorithm 5 would satisfy the KKT condition (6.18c) first

which demonstrates that a feasible solution is found if satisfied. Now we take a closer look at how the other KKT conditions are affected by the penalty parameter. The optimality condition of (6.27) yields

$$\partial F_k(x_k^{\nu_k+1}) + A_k^\top \lambda_k^{\nu_k} + \rho_k^{\nu_k} A_k^\top (A_k x_k^{\nu_k+1} - z_k^{\nu_k}) = 0, \tag{6.33}$$

and the optimality condition of (6.30) with respect to any $z_{k,l}$ yields

$$\lambda_{k,l}^{\nu_k+1} + \lambda_{l,k}^{\nu_k+1} + {\rho_k}^{\nu_k+1}(A_{k,l}x_k^{\nu_k+1} - z_{k,l}^{\nu_k+1}) + {\rho_l}^{\nu_k+1}(A_{l,k}x_l^{\nu_k+1} - z_{k,l}^{\nu_k+1}) - \alpha(z_{k,l}^{\nu_k+1} - z_{k,l}^{\nu_k}) = 0. \tag{6.34}$$

We observe that if the primal residue reaches a certain level of feasibility tolerance ϵ, the third term on the left-hand-side (LHS) of (6.33) and the third and fourth terms on the LHS of (6.34) will be larger with a larger penalty parameter ρ. In other words, while measured at the same level of feasibility, it is harder to fulfill KKT conditions (6.18a) and (6.18b), which makes the solution less optimal. Furthermore, if ρ increases to infinity, (6.18a) and (6.18b) can not be fulfilled and Algorithm 5 can only converge to a feasible but suboptimal solution. Therefore, the choice of penalty parameter involves the tradeoff between feasibility and optimality, and using a large penalty parameter or increasing the penalty aggressively results in finding a feasible solution faster which however can be less optimal. The increasing penalty method is usually adopted for large-scale systems because finding a feasible solution in a limited amount of time is more important in practice, and a solution can be regarded as a good solution as long as the achieved final objective value is within 1% of the true local minimum.

6.3.3 Impact of Penalty Parameter

In this section, we conduct numerical experiments to better understand the impact of the penalty parameter on the convergence performance of Algorithm 5. The test systems used are the IEEE 118-bus system, the Polish 2383-bus system and the Great Britain (GB) 2224-bus system. A flat start is used for the 118-bus system, while a warm start is used for the other two large systems. The feasibility of the solution is measured by maximum infeasibility which includes the maximum primal residue of all regions and the maximum constraint violation of constraints (2.1b) to (2.1f). Line limits are not included. The optimality of the solution is

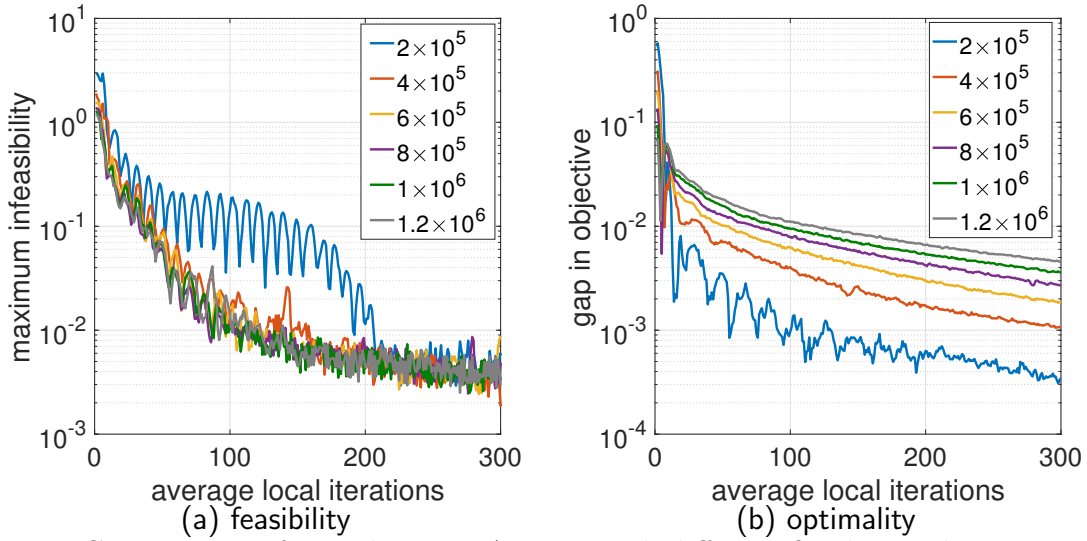
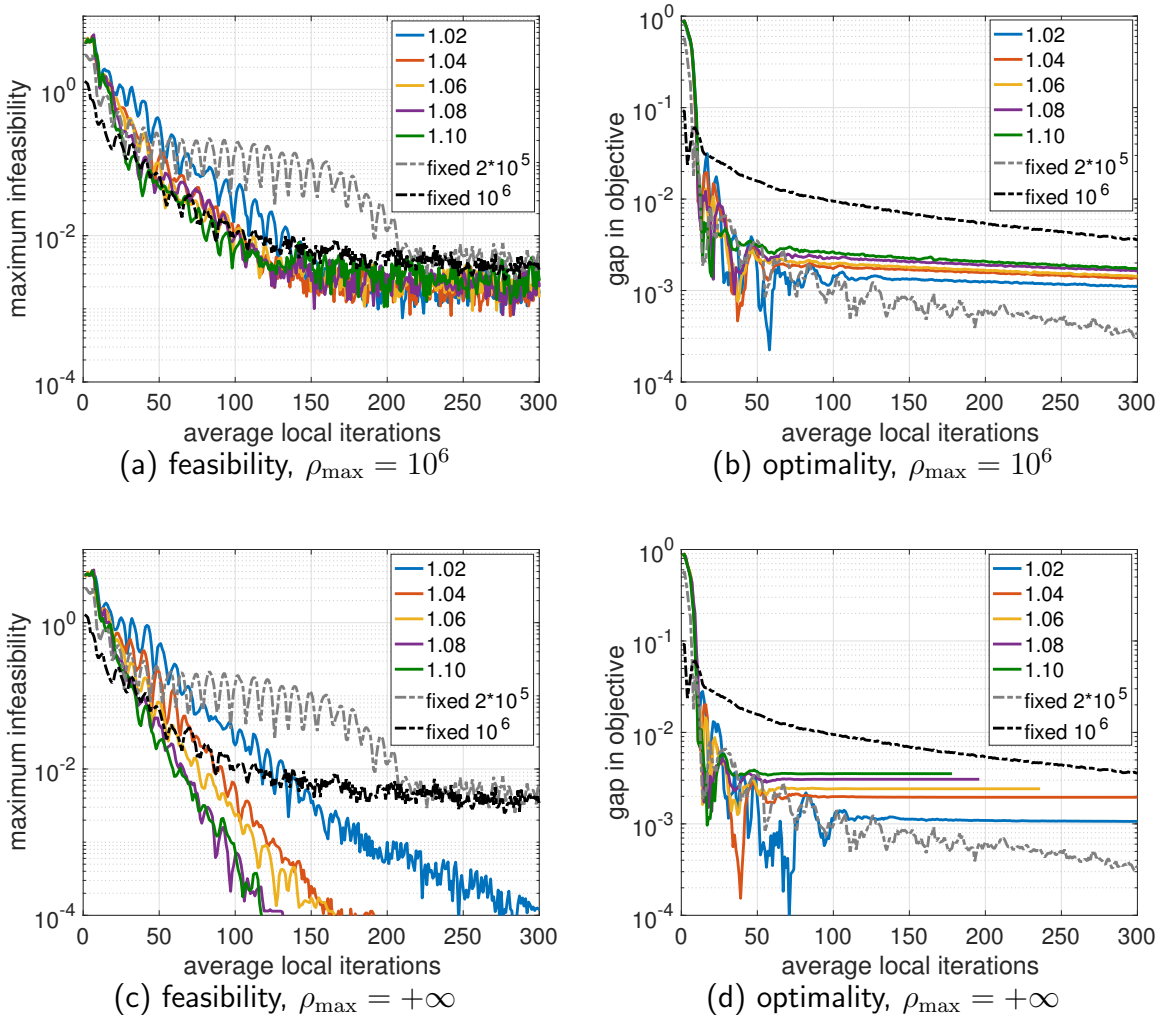


Figure 6.5: Convergence of asynchronous ADMM with different fixed ρ on the IEEE 118-bus system.

measured by the relative absolute difference between the objective function value achieved by Algorithm 5 and the optimal objective function value using the centralized Optimal Power Flow solver provided by MATPOWER [79], which will be denoted as "gap in objective" in the following analysis. We evaluate the convergence of Algorithm 5 to a feasible or optimal point in terms of both average number of local ADMM iterations and total execution time. The solver used to solve the local subproblems is the Tomlab SNOPT solver [86] implemented in Matlab. The total execution time includes the local computation time measured by CPU time and the communications delay of passing messages among regions. While studying the impact of penalty parameter and system partitioning, the communications delays are estimated using the same approach as in Chapter 6.2.3. The partitions of the systems are derived using the proposed spectral partitioning approach. Specifically, the 118-bus system, the Polish system and the GB system are divided into 8, 40 and 70 regions, respectively.

Before we use the increasing penalty method, we first investigate the impact of using a fixed penalty parameter on Algorithm 5; i.e., $\tau = 1$ in (6.29) on the IEEE 118-bus system. Figure 6.5 shows the convergence of Algorithm 5 on this system with ρ_0 varying from 2×10^5 to 1.2×10^6. With a larger penalty parameter, Algorithm 5 approaches feasibility faster especially during the first tens of iterations but approaches optimality slower. This indeed illustrates the tradeoff between feasibility and optimality related to the penalty parameter



(a) feasibility, $\rho_{\max} = 10^6$ (b) optimality, $\rho_{\max} = 10^6$

(c) feasibility, $\rho_{\max} = +\infty$ (d) optimality, $\rho_{\max} = +\infty$

Figure 6.6: Convergence of asynchronous ADMM with different increasing rate τ of ρ on the IEEE 118-bus system.

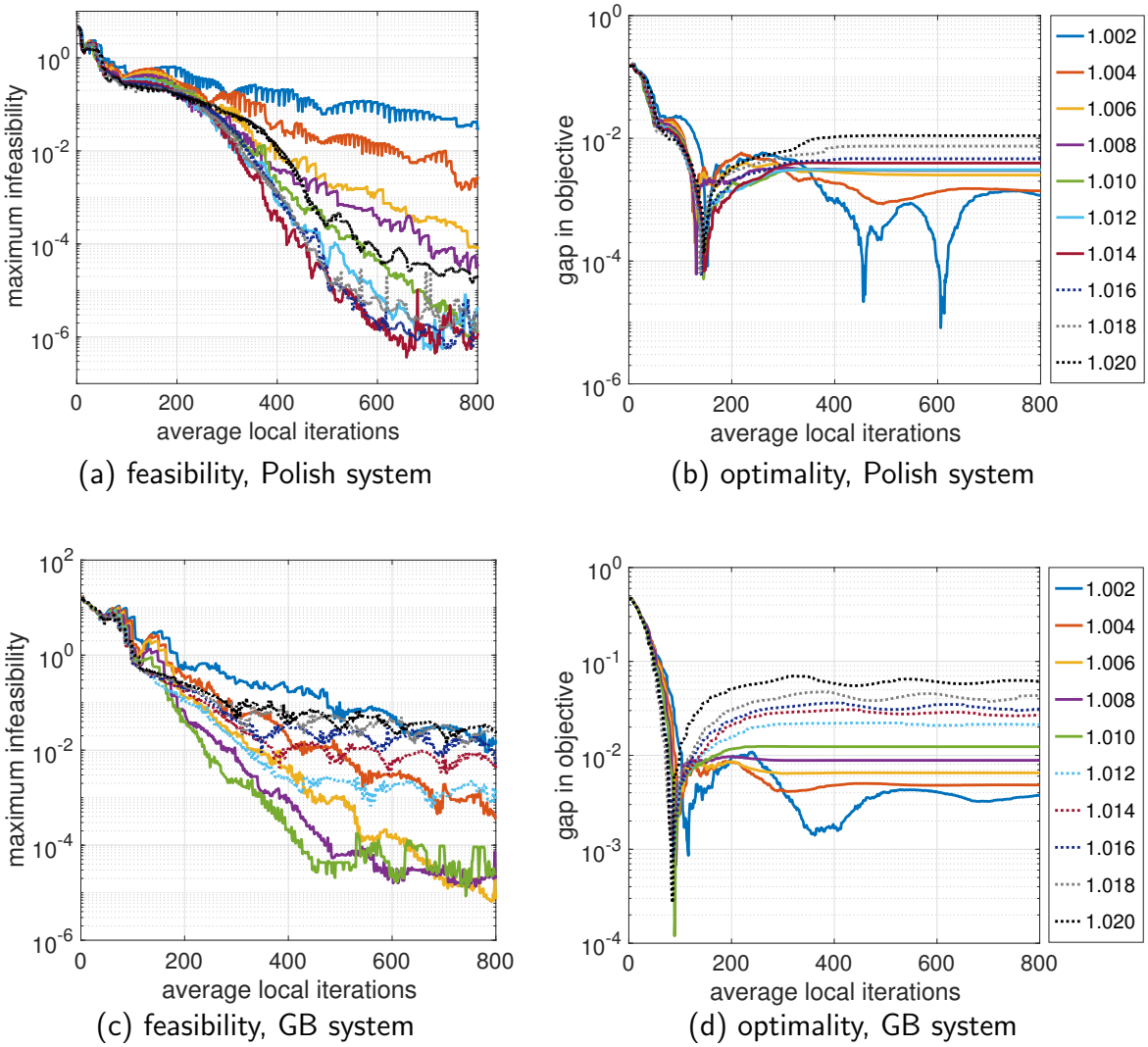
where a larger penalty biases towards finding a feasible but less optimal solution in fewer iterations.

In Fig. 6.6, we demonstrate the convergence of Algorithm 5 using the increasing penalty method but with various increasing rate τ from 1.02 to 1.10. ρ_0 is set to 8000, and $\rho_{\max}$ is set to 10^6 and $+\infty$, respectively to evaluate the impact of an unbounded penalty parameter. To compare the increasing penalty with the fixed penalty approach, we also replot the convergence curves of using fixed penalties 2×10^5 and 10^6 in grey and black dotted lines as references. As shown in Fig. 6.6a and Fig. 6.6b, using increasing penalty enables finding a solution with both high feasibility and optimality in relatively few iterations. Recall that with fixed penalty parameter, Algorithm 5 can only approach either feasibility or optimality

faster at the sacrifice of the other. However, with increasing penalty, Algorithm 5 approaches feasibility fast as if a large fixed penalty is used and the solution found is equally good as that found by using a small fixed penalty. Furthermore, with a larger increasing rate τ, Algorithm 5 tends to approach feasibility faster but the acceleration is rather limited as τ reaches 1.06. This shows that increasing the penalty parameter using a large stepsize is not necessary as it does not bring significant speed-up.

Figure 6.6c and Fig. 6.6d are presented to validate the fact that the penalty should be upper-bounded for the convergence of Algorithm 5 to an optimal point. As shown in Fig. 6.6c, infeasibility can decrease very fast with ρ increasing to infinity, but the gap in objective will stop to decrease as shown in Fig. 6.6d, which shows that Algorithm 5 at best reaches a suboptimal solution. Note that Algorithm 5 is terminated if the maximum infeasibility falls below 10^{-6}, which explains why in Fig. 6.6d some curves end before 300 local iterations.

For large-scale systems with thousands of buses, finding a feasible solution of the OPF problem quickly is always desired. Therefore, for the Polish and GB systems, we only consider using increasing penalty because using a fixed penalty parameter results in thousands of local iterations to converge which is too slow in practice. The impact of increasing rate τ on these two systems are illustrated in Fig. 6.7. ρ_0 is set to 10^7 and $\rho_{\max}$ is set to $+\infty$ for both systems. We first look at the Polish system (Fig. 6.7a and 6.7b) as an example, and we expect that the larger the increasing rate of penalty, the faster Algorithm 5 approaches feasibility. While this is the case when τ increases from 1.002 to 1.014, the convergence to feasibility slows down while keeping increasing τ beyond 1.014, as shown in Fig. 6.7a. A similar trend is also observed for the GB system as shown in Fig. 6.7c where the convergence of Algorithm 5 becomes slower as τ becomes larger than 1.010. This shows that for large and complex networks, increasing the penalty too fast harms convergence. The reason for this is that enforcing agreement of neighboring regions at the beginning of iterations while lacking complete information from all neighbors could lead the updates of variables into a wrong direction, which causes difficulty in finding local optima during local updates in later iterations. Interestingly, this observation is in line with some of the discussions made in [76] and [64] regarding parameter tuning of synchronous and asynchronous ADMM for solving the consensus problem, respectively, which show that the penalty parameter should be cho-



(a) feasibility, Polish system (b) optimality, Polish system

(c) feasibility, GB system (d) optimality, GB system

Figure 6.7: Convergence of asynchronous ADMM with different increasing rate τ of ρ on the Polish power system and the Great Britain power system.

sen neither too small nor too large. As shown in Fig. 6.7b and Fig. 6.7d, increasing the penalty faster leads to a less optimal solution. Therefore, for large-scale systems, one should increase the penalty parameter of Algorithm 5 at a slow pace. The best parameter to use for a specific system can be chosen via parameter sweeping, as the experiments conducted here. How to derive the best parameter settings from analytical studies could be an interesting topic for future research.

6.4 Impact of Partitioning

We have demonstrated in Chapter 3 and Chapter 4.1 that system partitioning can affect the convergence of the OCD and ADMM algorithms, respectively, which are both deployed in a synchronous manner. Therefore, it remains to be studied how system partitioning affects the proposed asynchronous ADMM scheme and whether using the proposed spectral partitioning technique is critical in an asynchronous setting as well.

In this section, we conduct experiments to show the importance of good system partitioning and that the spectral partitioning approach can also improve the performance of asynchronous ADMM. We first decompose the IEEE-118 bus system into 8, 12, 20 and 40 regions with the spectral partitioning approach and arbitrary partitioning, respectively. As previously discussed, the spectral partitioning method could generate more than one partitioning solution because the last step in spectral clustering is to use the Kmeans method to cluster the buses which can generate different results due to different initialization of centroids. While the approach in Chapter 3 and Chapter 4.1 is to choose one best partition out of the set of solutions, we keep all the partitioning solutions here and study their statistics. The motivation is to investigate if one chooses a partition out of the set of solutions spectral partitioning finds, whether this partition can perform better compared with an arbitrarily chosen partition. The arbitrary partition is generated by deploying the spectral clustering approach [84] using the connectivity matrix of buses as the affinity matrix; i.e., two buses will have affinity of 1 if they are directly connected by a transmission line, otherwise 0. The affinity matrix derived in this way contains no information regarding the different levels of coupling among different pairs of buses, which in other words, does not show any clusters of buses. Due to the fact that spectral clustering performs very bad when there are no identifiable clusters, we can regard the results generated by this method as arbitrary partitions of the system. For each number of regions, we compute 80 partitions each with the spectral partitioning approach proposed in [88] and the arbitrary partitioning approach described above, respectively. Then for each partition, we run Algorithm 5 three times and record the average of number of local iterations, execution time, and gap in the objective function at convergence. We say Algorithm 5 has converged when the maximum infeasibility falls below 10^{-3}. The increasing penalty method is deployed, and $\rho_0 = 8000$ and $\tau = 1.1$ are used in

Table 6.3: Convergence of asynchronous ADMM with increasing penalty ρ on the IEEE 118-bus system using spectral partitioning and arbitrary partitioning.

Regions	Partition	Iterations				Time (s)				Gap in Objective (%)			
		Mean	Median	Min	Max	Mean	Median	Min	Max	Mean	Median	Min	Max
8	Spectral	88	88	67	116	7.1	6.9	4.9	10.4	0.62	0.52	0.28	1.50
	Arbitrary	106	106	77	134	9.3	9.2	4.6	13.1	0.63	0.51	0.23	3.76
12	Spectral	90	88	75	117	5.0	4.9	4.1	7.2	0.93	0.91	0.56	1.55
	Arbitrary	120	119	95	218	5.8	5.7	4.6	10.1	1.48	1.11	0.43	7.45
20	Spectral	101	101	84	128	3.5	3.6	2.8	4.5	1.52	1.49	1.01	2.16
	Arbitrary	109	103	83	300	3.5	3.4	2.6	10.1	2.62	2.34	1.22	7.96
40	Spectral	126	125	110	146	2.6	2.5	2.2	3.0	2.58	2.63	1.56	3.42
	Arbitrary	706	728	123	1250	12.8	14.0	2.3	24.5	6.16	6.35	2.65	9.97

the simulations.

Table 6.3 summarizes the results from these experiments. Each row presents the statistics of the three performance metrics for all 80 partitions. The first observation here is that with an increasing number of regions, the number of iterations and the gap in objective increase but the execution time decreases with both partitioning methods. This shows that the asynchronous ADMM approach is scalable in terms of a large number of regions but the solution is less optimal. One way to reduce the gap is to reduce the penalty increasing rate, which on the other hand, will result in more iterations. The second observation is that Algorithm 5 generally has a better performance with spectral partitioning than that with arbitrary partitioning. Particularly, the maximum value of iterations, time and gap of using an arbitrary partition can be much larger than that using a partition from the set of solutions spectral partitioning finds. Furthermore, with severe partitioning of the system into 40 regions, Algorithm 5 fails to converge with 54 out of the 80 arbitrary partitions within the maximum local iteration number which is set to 1250. But with spectral clustering, Algorithm 5 successfully converged with all 80 partitions within 150 local iterations. Therefore, it is fair to say that using the spectral partitioning technique, there is a better chance of identifying a partition that boosts the performance of the asynchronous ADMM approach.

While using spectral partitioning may not seem very beneficial on the IEEE 118-bus

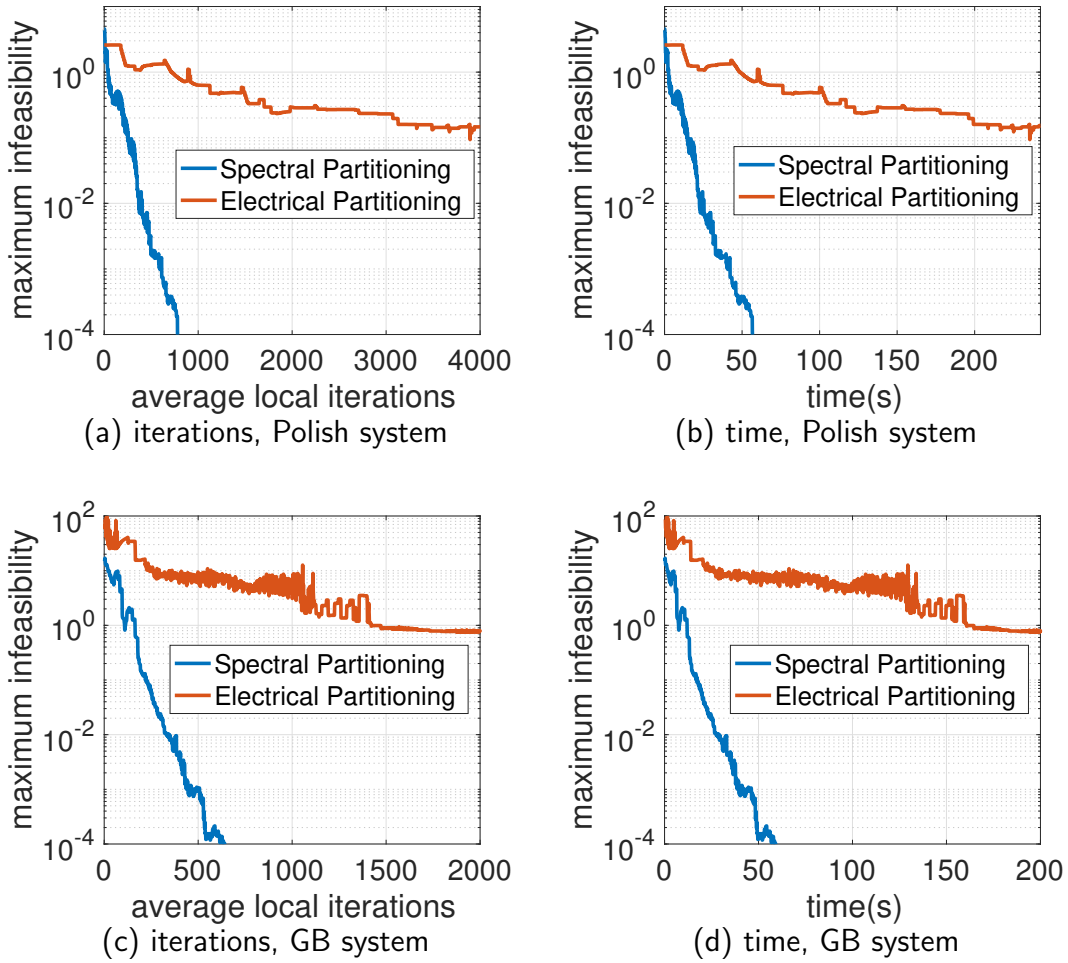


Figure 6.8: Convergence of asynchronous ADMM with spectral partitioning and electrical partitioning.

system, it is much more critical when employing Algorithm 5 on large-scale transmission networks. For the Polish and GB systems, we compare the most balanced partition computed by spectral partitioning with the partition generated by an electrical partitioning approach used in [10]. The electrical partitioning method assigns a bus to the same region as its closest generator bus in terms of electrical distance represented by admittance. Again, the increasing penalty method is used with $\rho_0 = 10^7$ used for both systems. τ is set to 1.012 and 1.006 respectively for the Polish and GB systems based on the results in Fig. 6.7. Figure 6.8 shows the convergence of Algorithm 5 with these two partitioning method, and it is clearly shown that using spectral partitioning, Algorithm 5 converges to a feasible solution significantly faster than using electrical partitioning. We should mention that the gaps in the objective function of the solutions found using the spectral partitioning of these two

systems are both below 1%. These findings demonstrate that if a partition is not chosen intelligently, Algorithm 5 is likely to fail to find a feasible solution in time if the AC OPF is expected to be solved every five minutes, which shows the importance of good system partitioning for distributed optimization on large-scale systems.

6.5 Impact of Communication Delays

At last, we investigate the impact of communication delays using the considered three power systems. Here the delay denotes the time from sending out a message from the source region until it is received by the destination region. The communication delays affect the actual number of arrived neighbors before each local iteration. The larger the delay, the fewer neighbors a region could possibly receive a message from before it starts a new iteration during the first couple of iterations of Algorithm 5. Consequently, the region will update fewer z variables, namely, the ones that are associated with arrived neighbors. Therefore, the local update makes less progress in a single iteration, which then could lead to an increased number of iterations.

We first conduct a study on the 118-bus system and then consider the Polish and the GB systems. For the 118-bus system, the number of neighbors each region has and the average computation time of solving the local OPF problem at one iteration are shown in Table 6.4. Five ranges of delays are considered varying from milliseconds if fiber optical communications are used with direct dedicated links among regions to 1 or 2 seconds if no direct link exists such that the message has to be routed via multiple hops or the traffic load of the network is high. For each pair of neighboring regions, its associated communication delay is randomly generated within the range listed in the second row of Table 6.5. The rest of Table 6.5 shows the average number of arrived neighbors of region k (denoted by na_k)

Table 6.4: Number of neighbors and local computation time of the 118-bus system.

Region	1	2	3	4	5	6	7	8
$\lvert\mathcal{N}_k\rvert$	3	2	3	1	4	3	3	1
Computation time (s)	0.31	0.13	0.09	0.12	0.15	0.14	0.13	0.11

Table 6.5: Communication delays and number of arrived neighbors on the 118-bus system.

Case	I	II	III	IV	V
Delay (s)	0.003-0.005	0.03-0.05	0.3-0.5	0.6-1.0	1.2-2.0
na_1	3.0	3.0	2.6	2.4	1.8
na_2	1.6	1.5	1.3	1.2	1.0
na_3	1.5	1.4	1.4	1.2	1.0
na_4	1.0	1.0	1.0	1.0	1.0
na_5	3.3	3.2	2.9	2.0	1.1
na_6	2.7	2.6	1.7	1.5	1.2
na_7	2.8	2.7	2.4	1.8	1.1
na_8	1.0	1.0	1.0	1.0	1.0

before its each local update. We calculate this statistic using the first 20 local iterations for each region because the first few iterations are critical in ADMM that determines the final point it converges to. It is shown in Table 6.5 that with increasing communication delays, the number of arrived neighbor decreases as expected. With small delays such as in Case I and II, all regions can receive the updated information from the majority of their neighbors, while with large delays such as in Case V, most regions have to wait until one neighbor arrives. This indicates that larger communication delays lead to more severe asynchronism.

The severity of asynchronism has a strong impact on the value for the penalty parameter that achieves a solution of good quality. Figure 6.9 shows the gap of the objective function between Algorithm 5 and the centralized solution with respect to the increasing rate τ of the penalty ρ. For comparison, we also simulated the synchronous counterpart of Algorithm 5, i.e., Algorithm 1 where each region has to wait for all neighbors before it carries out its local update. In the synchronous case, the faster the penalty increases, the larger is the gap. But as all regions need to wait for all neighbors regardless of the communication delay, similar trends can be observed for all cases with various communication delays. In contrast, in the asynchronous case, the sensitivity of Algorithm 5 to the increasing rate of penalty highly depends on the communication delays. For example, with large communication delays such as in Case V, one needs to increase the penalty at a very slow pace to reach a solution with a gap smaller than 1%.

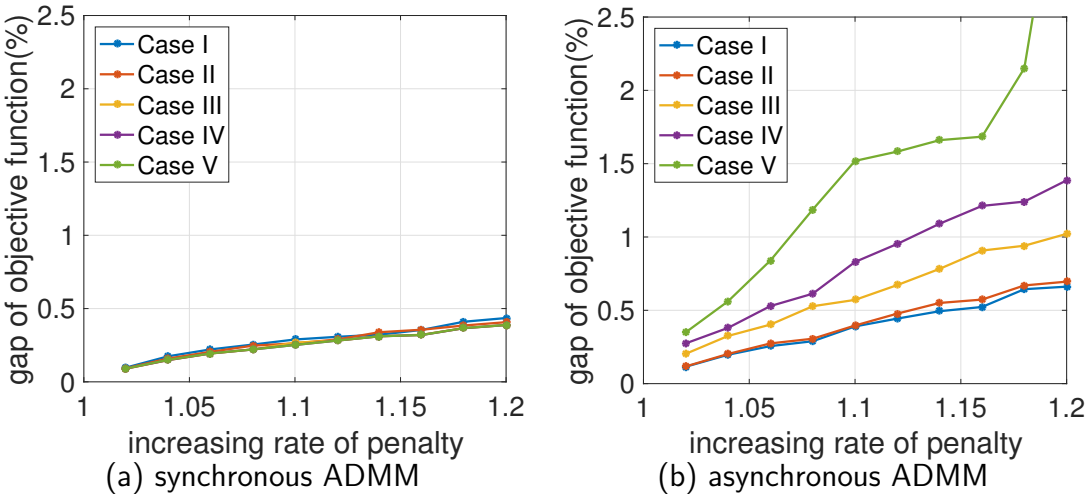


Figure 6.9: Solution quality and the penalty increasing rate.

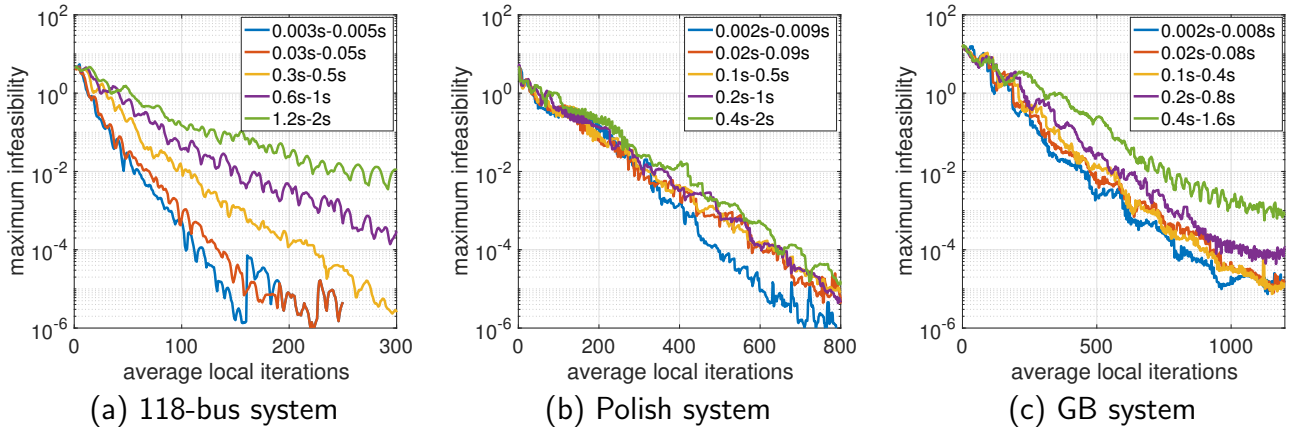


Figure 6.10: Convergence of asynchronous ADMM with various communications delays.

Now we evaluate the impact of the level of asynchrony on the convergence rate of Algorithm 5 using all three test systems. ρ_0 is set to 8000, 10^7 and 10^7, and τ is set to 1.1, 1.012 and 1.006, respectively for the 118-bus, Polish and GB systems. Fig. 6.10 shows that Algorithm 5 also needs more iterations to converge as the communications delay increases. However, for the Polish system, the increase of delay seems to only affect the convergence of Algorithm 5 slightly, as illustrated in Fig. 6.12b. The reason for this is that as discussed in Chapter 6.3.2, the communication delay will only have an impact if the number of received messages decreases significantly due to an increase in the delay, which is not the case for the Polish system, as discussed next.

To visualize the level of asynchrony on large-scale systems, we also plot the average

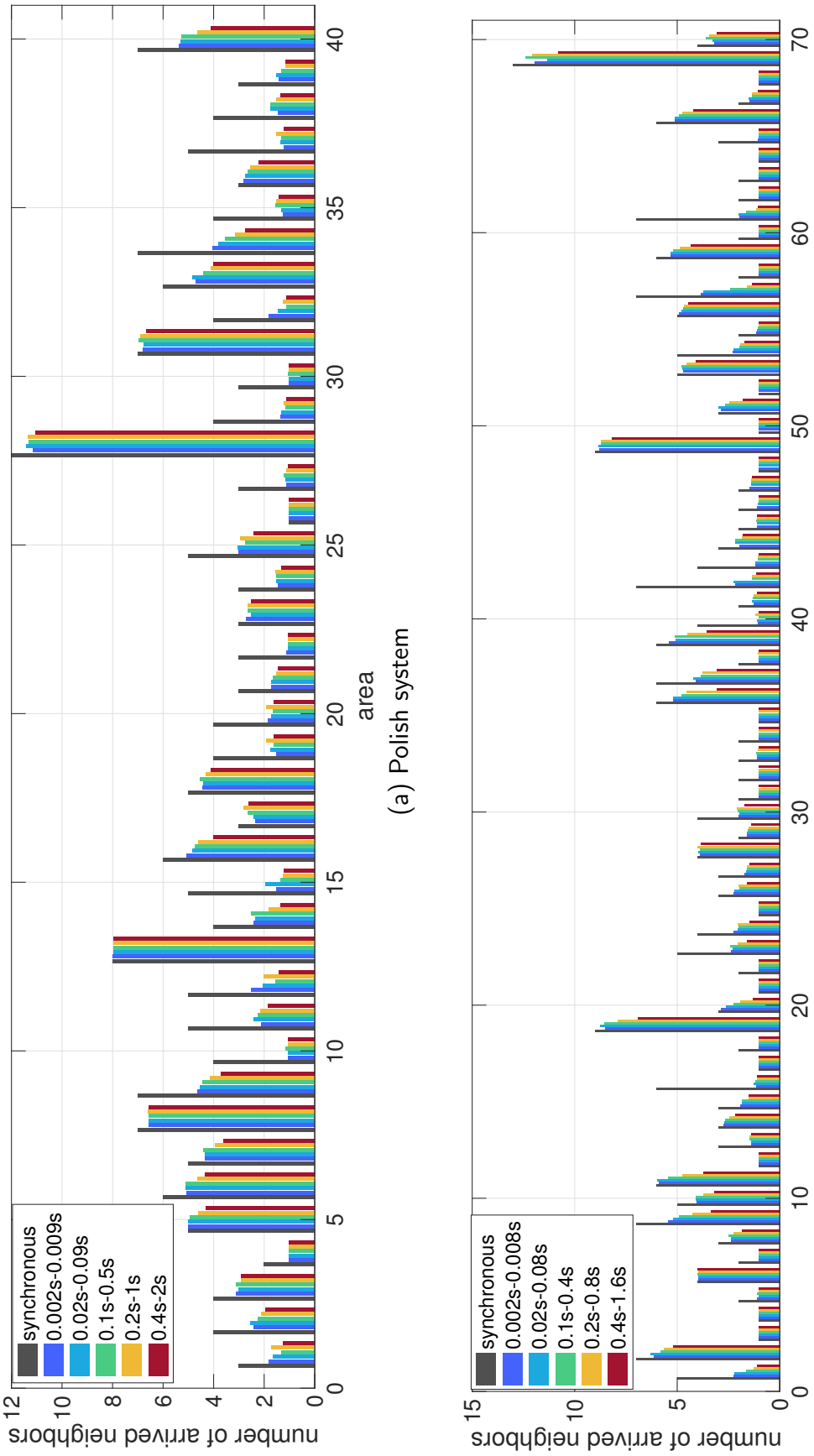


Figure 6.11: Average number of arrived neighbors during the first 20 local iterations at each area.

number of arrived neighbors at each region during the first 20 local ADMM iterations in Fig. 6.11. The x-axis of Fig. 6.11 is the index of regions. The grey bar denotes the total number of neighbors one regions has, which corresponds to the number of arrived neighbors if synchronous ADMM is used. From Fig. 6.11, one can observe that even though Algorithm 5 allows a region to perform local updates after receiving message from one neighbor, the actual number of arrived neighbors during the time when this region is doing computation is usually more than one. This shows that the asynchronism among the regions are dependent on the system topology and may not be as severe as one might have conjectured. As shown in Fig. 6.11a, the number of arrived neighbors doe not necessarily decrease significantly with increasing delays for the Polish system, which explains why the iterations of Algorithm 5 are not affected much by varying delays for this specific system. Therefore, even though larger delays generally lead to more iterations of Algorithm 5, how significant that increase could be is system dependent and should be analyzed case by case.

Finally, we would like to conduct a comparison of the asynchronous ADMM method with increasing penalty, i.e., Algorithm 5 with its synchronous counterpart, i.e., Algorithm 1, under various communication delays. Table 6.6 demonstrates the performance of these two algorithms on the 118-bus system with $\bar{\nu}_k$ denoting the average local iterations among all regions. The following observations can be made: 1) With the same penalty increasing rate, asynchronous ADMM on average takes more iterations than synchronous ADMM but with less time spent on each iteration. Therefore, asynchronous ADMM generally takes less time to converge but the solution quality is worse. 2) To achieve a similar level of solution quality, asynchronous ADMM needs to adopt a smaller penalty increasing rate that leads to more iterations and possibly longer execution time. 3) With the increase of communication delays, the number of iterations of asynchronous ADMM increases even with a fixed penalty increasing rate due to the more severe asynchronism among regions. 4) On this test system, asynchronous ADMM achieves comparable performance of its synchronous counterpart for Cases I to IV where the delay is small or comparable with the computation time, but slows down substantially if the delay is dominant such as in Case V. Note that with the synchronous ADMM algorithm using the same penalty increasing rate, the number of iterations could be different under different delays. This is due to the implementation of the synchronous

Table 6.6: Performances of synchronous and asynchronous ADMM with increasing penalty under various communication delays on the 118-bus system.

Delay (s)	Method	τ	$\bar{\nu}_k$	Gap(%)	Time(s)
0.003-0.005	sync	1.18	35	0.40	9.7
	async	1.18	56	0.60	7.8
	async	1.10	91	0.40	11.5
0.03-0.05	sync	1.20	31	0.38	12.3
	async	1.20	61	0.70	10.6
	async	1.10	96	0.40	16.0
0.3-0.5	sync	1.20	48	0.39	28.8
	async	1.20	89	1.02	14.0
	async	1.06	233	0.40	33.6
0.6-1.0	sync	1.20	48	0.39	52.1
	async	1.20	126	1.39	20.9
	async	1.04	377	0.40	57.1
1.2-2.0	sync	1.16	57	0.32	118.0
	async	1.16	238	1.52	40.0
	async	1.02	1119	0.35	169.4

Table 6.7: Local computation time using asynchronous ADMM and communication delays with different 40-region partitions of the Polish system.

Partition	Computation (s)			Small delays (s)			Medium delays (s)			Large delays (s)		
	Min	Max	Mean	Min	Max	Mean	Min	Max	Mean	Min	Max	Mean
P1	0.05	1.95	0.43	0.0025	0.0088	0.0041	0.25	0.88	0.41	1.25	4.42	2.03
P2	0.04	2.47	0.48	0.0025	0.0072	0.0040	0.25	0.72	0.40	1.25	3.58	2.00
P3	0.05	2.69	0.45	0.0029	0.0092	0.0041	0.29	0.92	0.41	1.46	4.58	2.06
P4	0.06	3.78	0.46	0.0026	0.0058	0.0039	0.26	0.58	0.39	1.28	2.91	1.97

algorithm here which lets a region start its local update after receiving information from all neighbors without any global synchronization. Such implementation is slightly different from a strictly synchronous implementation coordinated by a master node and could lead to difference in the sequence of regional updates under different delays.

Table 6.8: Parameter setting and solution quality of asynchronous ADMM on the Polish system under different delays.

Scheme	Small delays (s)					Medium delays (s)					Large delays (s)				
	τ	Gap (%)				τ	Gap (%)				τ	Gap (%)			
	All	P1	P2	P3	P4	All	P1	P2	P3	P4	All	P1	P2	P3	P4
Sync	1.060	0.41	0.42	0.49	0.42	1.060	0.38	0.41	0.40	0.36	1.100	0.56	0.54	0.57	0.50
Async	1.012	0.39	0.34	0.33	0.34	1.012	0.41	0.37	0.51	0.39	1.006	0.53	0.40	0.50	0.35

While for the 118-bus system, Algorithm 5 seems to converge slightly slower than its synchronous counterpart, it converges faster when applied to large-scale systems such as the Polish system. For the Polish system, we also consider four 40-region partitions which are all found by the spectral partitioning approach but are different in terms of sizes of regions and their associated local computation time spent on solving the subproblems at one iteration. For each partition, we carry out both synchronous and asynchronous ADMM under three levels of delays. In the small delay scenario, the delays of passing messages among pairs of neighboring areas are calculated using (5.7) based on the assumptions and schemes proposed in Chapter 5. While for the scenarios of medium and large delays, we multiply the delays calculated in the small delay scenario by 100 and 500, respectively. The minimum, maximum, and average value of these pair-wise delays are presented in Table 6.7. Note that the local computation time is only dependent on the solver used but not on the delays, and the average values of the minimum, maximum and mean local computation time of all regions over all experiments associated with a particular partition are shown in Table 6.7. Here, P1 is the most balanced partition found by the spectral partitioning method in terms of the difference in the numbers of buses the largest and smallest regions have. In fact, it can be difficult to search for a very balanced partition in a large-scale system and more importantly, even if a more balanced partition exists, it may not be a partition that has weaker couplings among the regions and could result in lower efficiency while deploying distributed optimization. Comparing the local computation time and the delays, we consider the delay to be small if the average delay is more than one magnitude smaller than the average computation time, medium if the average delay is around the same magnitude as the average computation time, and large if the average delay is more than five

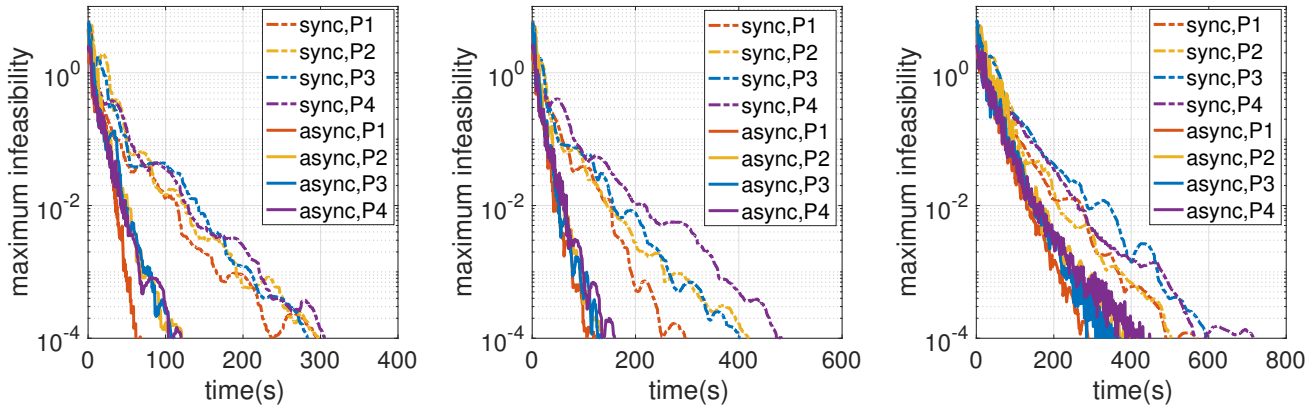


Figure 6.12: Convergence of synchronous/asynchronous ADMM on the Polish system with various communication delays and partitions.

times the average computation time.

For all the experiments, the initial penalty parameter is set to 10^7. However, the increasing rates of the penalty parameter are chosen differently under different levels of delays and algorithms. The settings of the penalty increasing rate τ are given in Table 6.8, which are chosen in a way that the synchronous and asynchronous ADMM can achieve similar accuracy measured by gap in the objective function at convergence. Specifically, the gap is measured when the maximum infeasibility falls below 10^{-3}. Similar to the findings presented in Table 6.6, to achieve comparable accuracy, one can increase the penalty faster for synchronous ADMM compared with asynchronous ADMM.

Figure 6.12 shows the convergence of synchronous and asynchronous ADMM with the chosen four partitions under various delays. As shown, regardless of how large the communication delays are, asynchronous ADMM always converges faster than its synchronous counterpart. While synchronous ADMM tends to converge slower with partitions that are more unbalanced such as P4, asynchronous ADMM is not affected as much by the heterogeneity of the subproblems, which indeed validates that asynchronous schemes are more suitable for solving subproblems with different sizes. Compared with the performance of asynchronous ADMM on the 118-bus system presented in Table 6.6, the asynchronous approach is more beneficial for the Polish system because the control regions here are more unbalanced and the synchronous scheme wastes a lot of time waiting for slow regions, which

is generally the case in large-scale systems. Therefore, the proposed asynchronous ADMM can be an efficient alternative to synchronous ADMM when the partition is unbalanced, at the tradeoff of increasing communications traffic. Hopefully, such communications requirement will be supported by the communications network of the smart grid that is currently incorporating more direct data communications links among the substations and control centers.

A final remark here is that even though asynchronous ADMM converges fast on the Polish system under small or medium delays, it might not converge under large delays if the AC OPF problem need to be solved every five minutes. In fact, large delays suggest that the communications infrastructure of the system is not suitable for deploying distributed optimization algorithms because the delays will be dominant and slow down these algorithms drastically. This observation also suggests that out of the candidate partitions found by the spectral partitioning approach, one should consider the conceivable communications infrastructures associated with these partitions and estimate communication delays. If different partitions incur delays that vary across several magnitudes, then one should narrow down the choice of partitions to the candidates with delays in the smallest range. Then out of the partitions with relatively small delays, one can choose the partition according to other operational or legacy constraints.

Chapter 7

Closure

7.1 Conclusion

This thesis has presented methods that enable the implementation of distributed optimization methods in large-scale power systems to solve non-convex optimal power flow problems. Particularly, we have investigated three issues, namely, system partitioning, communications, and synchronization, that should be carefully considered when applying any distributed algorithm to a real-world system.

First, we proposed a spectral partitioning method which determines how a power system should be partitioned to achieve the fastest convergence when applying the OCD method. The proposed partitioning method determines the best partition by grouping the computationally strongly coupled buses into one area and assigning weakly coupled buses to different areas using the spectral clustering technique. Simulation results on multiple IEEE test systems have shown that by using the proposed partitioning approach, OCD converges significantly faster than using other partitions of the system and the convergence time could be shorter than the convergence time of the centralized approach.

We then applied the partitioning approach in conjunction with the distributed ADMM method. By using the partitions computed by the partitioning method, ADMM can find a solution very close to the local optimum very efficiently, which demonstrates the applicability of distributed optimization in large-scale real power systems. Additionally, it was checked and verified that the partitions found are robust with respect to the starting point,

the parameters used in ADMM, and the inclusion of thermal line constraints. We also applied the partitioning approach with OCD to solve a multi-step OPF problem and showed that the best partition of the system is applicable to a wide range of time steps. These results show that the spectral partitioning method can be generalized to different distributed methods, e.g., ADMM, and problems that span multiple time steps, which provides a structured approach to decompose a large-scale system such that an efficient implementation of distributed optimization methods is possible.

Next, we have presented a generic approach that models the communications network architectures and simulates the *communications process* for distributed optimization using the OPNET modeler. In particular, we have considered the centralized and the distributed communications infrastructures and compared the convergence performance of OCD and ADMM with communications delay taken into account. Our results suggest that the centralized infrastructure might be prohibitive for deploying methods with frequent information exchange or a large number of control areas, and a distributed communications plane and infrastructure is more suitable for reaping the full benefits of distributed optimization, especially in large-scale power systems. The findings reported in this part, particularly the impact of the communications plane on distributed optimization, bridges the gap that exists between the communications research community and the distributed optimization research community in how these two research communities see the problem of deploying distributed methods in power systems.

Finally, we proposed an asynchronous distributed ADMM approach to solve the nonconvex OPF problem. The proposed method does not require any centralized coordination and allows any worker to perform local updates with information received from a subset of its physically connected neighbors. We provided the sufficient conditions under which asynchronous ADMM asymptotically satisfies the first-order optimality conditions. Through numerical studies on three large-scale systems, we have investigated the impact of parameter tuning, system partitioning, and communications delay on the proposed asynchronous ADMM method. Specifically, we have shown that the penalty parameter used in ADMM is highly related to the tradeoff of the feasibility and optimality of the solution, which needs to be increased slowly for large-scale networks such as the Polish and Great Britain systems.

We have also demonstrated the importance of partitioning and that the spectral partitioning method works well with the asynchronous ADMM approach. In terms of communications delay, we have shown that asynchronous ADMM needs more iterations to converge if the severity of asynchrony among the regions increases due to large delays. These findings have shown that asynchronous ADMM could be more efficient than its synchronous counterpart if it is implemented with proper system partitioning, parameter tuning and communications technologies.

In conclusion, the findings presented in this thesis suggest that the developed approaches in the aspects of system partitioning, communications, and synchronization could provide a roadmap for making distributed optimization practical and viable in large-scale power systems. Figure 7.1 provides a flowchart outlining the steps one could follow if one is considering to deploy distributed optimization for a specific system. The first step is to determine possible ways to decompose the system. To apply the proposed spectral partitioning method, the assumption made in this thesis is that a specific application should be considered. However, to apply the distributed optimization approach for multiple applications, one can apply the partitioning method to all considered problems and choose a partition that works for many applications or the most important applications. We also assume that the number of regions is predetermined because it is dependent on the computational capabilities available and other legacy constraints. Then, one should choose a partition whose communications infrastructures support message passing among neighboring regions and the associated communication delays are relatively small among all the partition candidates. Next, according to the actual level of delays associated with the chosen partition and the heterogeneity of regions, one can choose a proper distributed optimization algorithm and tune the parameters if necessary. This flowchart is built upon the findings presented throughout this thesis and can be used as a guidance for the implementation of distributed optimization on large-scale systems.

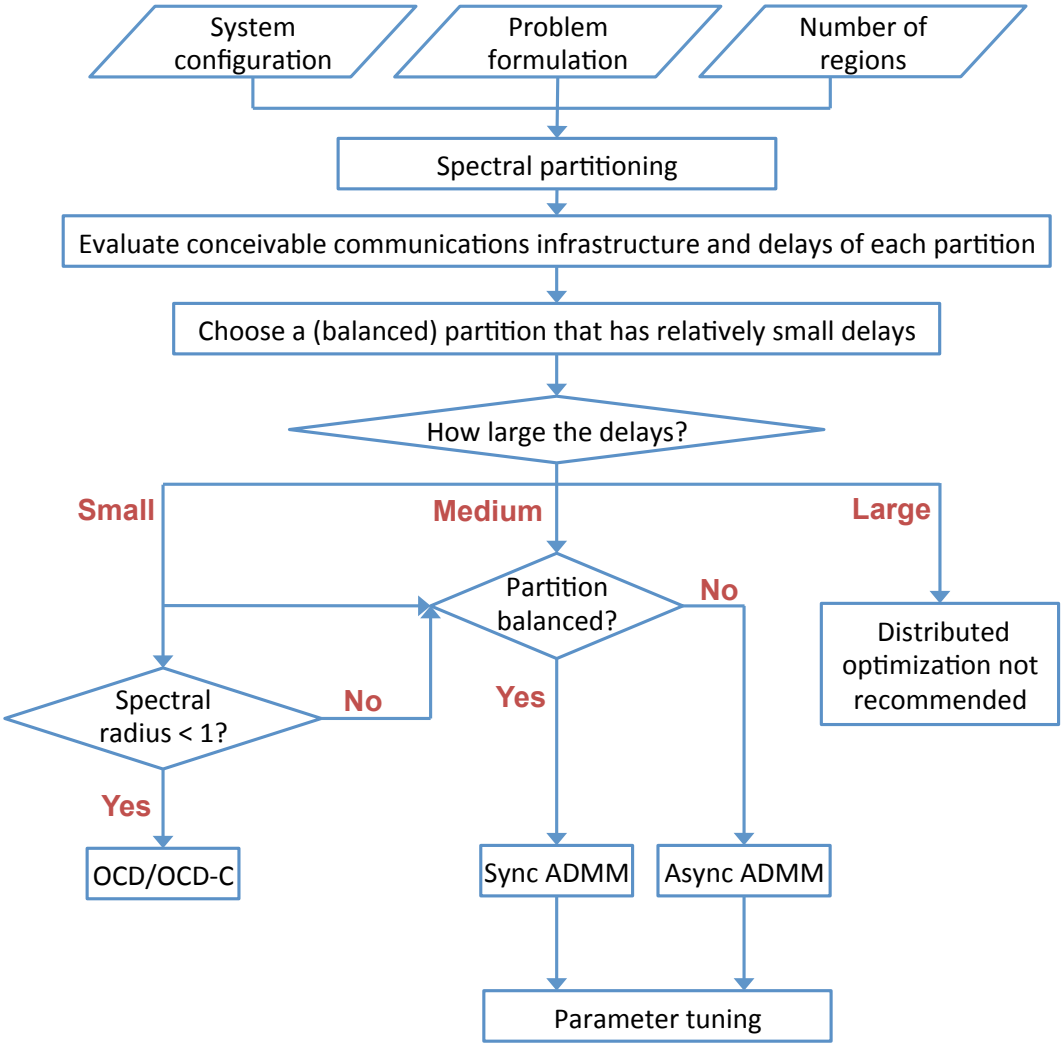


Figure 7.1: Steps of implementation of distributed optimization on a large-scale system.

7.2 Future Directions

There are three directions in which the work presented in this thesis could be extended. First, while this thesis has shown preliminary results that asynchronous ADMM could be a viable approach to tackle non-convex OPF problems in large-scale systems, its convergence behavior needs to be further investigated. This can be done by either theoretical analysis that aims to derive the best algorithm parameter or more comprehensive empirical studies that applies the proposed asynchronous approach to a large number of real-world systems.

The second direction lies in the application field. While this thesis studies the OPF problem for the transmission network, the methods developed could also be useful for dis-

tribution network management. As many renewable and distributed energy resources are integrated and managed at the distribution level, distributed optimization will play a key role in the coordination of these resources. It is worth studying how the partitioning approach and the modeling of the communications plane should be modified to better adapt to the characteristics of distribution networks. Furthermore, it will also be interesting to study how to solve the optimization problems of transmission and distribution networks in a collaborative and interactive manner by treating them as one integrated system instead of two decoupled systems.

The third direction is more oriented towards implementation. While this thesis studies distributed optimization algorithms, the experiments are conducted on a single machine in a sequential manner which simulates the process of distributed algorithms to show proofs of concepts. When the algorithms are ready to be applied in practice, it will be necessary to build a distributed computation environment that utilizes a cluster of computers to run these algorithms in a truly distributed manner. In fact, we have made some efforts towards this direction by implementing the proposed asynchronous ADMM in Python using Python's built-in parallel computing module. The source code using IEEE test systems can be found at https://github.com/guojunyao419/OPF-ADMM. This parallel implementation can serve as a starting point, and collaborative efforts from computer and network engineers and researchers are needed to build a distributed computing platform which will be beneficial for both academic research and industrial applications.

Overall, it is our hope that the methods and findings presented in this thesis could establish the much needed interconnection between the research communities of optimization, power systems and communications and spur up more interdisciplinary research in this area. As the power systems today are evolving towards a more distributed architecture, we believe that distributed optimization approaches will provide important tools to tackle the upcoming challenges in the transition to a smarter grid.

Appendix A

Convergence Criterion for OCD-C

In OCD-C, the update of variables associated with area k is calculated by

$$\Delta y_k = H_{kk}^{-1} \cdot (-r_k + \hat{r}_k), \tag{A.1}$$

where

$$\hat{r}_k = \sum_{l=1, l \neq k}^{K} H_{kl} H_{ll}^{-1} \cdot r_l. \tag{A.2}$$

Now, consider the simple case given in Chapter 2.3.2 where the overall problem is divided into two subproblems resulting in solving

$$\begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \cdot \begin{bmatrix} \Delta y_1 \\ \Delta y_2 \end{bmatrix} = - \begin{bmatrix} r_1 \\ r_2 \end{bmatrix} \tag{A.3}$$

If we use g^ν to denote the value of the right-hand-side vector of (A.3) which are the KKT conditions evaluated at iteration ν, then

$$g^\nu = g(y^\nu) = \begin{bmatrix} r_1^\nu - \hat{r}_1^\nu \\ r_2^\nu - \hat{r}_2^\nu \end{bmatrix} \tag{A.4}$$

$$\Delta y^\nu = - \begin{bmatrix} H_{11}^\nu & 0 \\ 0 & H_{22}^\nu \end{bmatrix}^{-1} \cdot g^\nu = -\overline{H}^{\nu-1} \cdot g^\nu. \tag{A.5}$$

If we use y^* to denote the optimal solution, then the Taylor series expansion of g^ν yields:

$$g^\nu = g(y^*) + \nabla g(y^*)(y^\nu - y^*) + o(||y^\nu - y^*||)$$

$$
\begin{aligned}
&= \nabla \begin{bmatrix} r_1^* - H_{12}^* {H_{22}^*}^{-1} \cdot r_2^* \\ r_2^* - H_{21}^* {H_{11}^*}^{-1} \cdot r_1 \end{bmatrix} (y^\nu - y^*) + o(||y^\nu - y^*||) \\
&= \begin{bmatrix} H_{11}^* - H_{12}^* {H_{22}^*}^{-1} H_{21}^* & 0 \\ 0 & H_{22}^* - H_{21}^* {H_{11}^*}^{-1} H_{12}^* \end{bmatrix} \cdot (y^\nu - y^*) + o(||y^{\nu)} - y^*||) \\
&= \widetilde{H}^* (y^\nu - y^*) + o(||y^\nu - y^*||). \qquad \text{(A.6)}
\end{aligned}
$$

Then, for the next iteration

$$
\begin{aligned}
y^{\nu+1} - y^* &= y^\nu + \Delta y^\nu - y^* \qquad \text{(A.7)} \\
&= (I - \overline{H}^{\nu-1} \widetilde{H}^*)(y^\nu - y^*) + o(||y^\nu - y^*||).
\end{aligned}
$$

Using the triangular inequality, one can get

$$
||y^{\nu+1} - y^*|| \leq ||I - \overline{H}^{\nu-1} \widetilde{H}^*||\ ||y^\nu - y^*|| + o(||y^\nu - y^*||). \qquad \text{(A.8)}
$$

Dividing both side of (A.8) by $||y^\nu - y^*||$ and taking limits of ν, one can see that in order for OCD-C to converge, the following has to hold:

$$
lim_{\nu \to \infty} ||I - \overline{H}^{*-1} \widetilde{H}^*||^\nu = 0. \qquad \text{(A.9)}
$$

Based on the theorem on spectral radius which states that $\rho(A) < 1$ if and only if $lim_{\nu \to \infty} A^\nu = 0$, OCD-C will converge if $\rho(I - \overline{H}^{*-1} \widetilde{H}^*) < 1$.

Appendix B

Optimality Conditions of ADMM Subproblems

For the x-update in ADMM at the ν-th iteration, the k-th region solves the following subproblem

$$\underset{x_k}{\text{minimize}} \quad f_k(x_k) + \lambda_k^{\nu-1\top}(A_k x_k - z_k^{\nu-1}) + \frac{1}{2}\|A_k x_k - z_k^{\nu-1}\|^2_{\rho_k^{\nu-1}} \tag{B.1a}$$

$$\text{subject to} \quad g(x_k) = 0. \tag{B.1b}$$

For the simplicity of presentation, only equality constraints are considered in the following analysis which can be readily extended to include inequality constraints. The optimality conditions of subproblem (B.1) are

$$\nabla f_k(x_k) + \mu_k^\top \nabla g(x_k) + A_k^\top \lambda_k^{\nu-1} + A_k^\top \text{diag}(\rho_k^{\nu-1})(A_k x_k - z_k^{\nu-1}) = 0 \tag{B.2a}$$

$$g(x_k) = 0, \tag{B.2b}$$

where μ_k are the Lagrange multipliers of (B.1b). In (B.2a), the terms $A_k^\top \lambda_k^{\nu-1}$, $A_k^\top \text{diag}(\rho_k^{\nu-1}) A_k x_k$ and $A_k^\top \text{diag}(\rho_k^{\nu-1}) z_k^{\nu-1}$ only contain non-zero entries in the rows that correspond to the variables (x_k, μ_k) associated with the boundary buses. Hence, for the non-boundary buses, (B.2a) reduces to $\nabla f_k(x_k) + \mu_k \nabla g(x_k) = 0$, which combined with (B.2b) are the centralized optimality conditions to be fulfilled at those non-boundary buses.

Appendix C

Proof of Lemma 1

Proof of Lemma 1.

$$\begin{aligned} & L(x^{\nu+1}, z^{\nu+1}, \lambda^{\nu+1}) - L(x^{\nu}, z^{\nu}, \lambda^{\nu}) \\ & = L(x^{\nu+1}, z^{\nu}, \lambda^{\nu}) - L(x^{\nu}, z^{\nu}, \lambda^{\nu}) \\ & \quad + L(x^{\nu+1}, z^{\nu}, \lambda^{\nu+1}) - L(x^{\nu+1}, z^{\nu}, \lambda^{\nu}) \\ & \quad + L(x^{\nu+1}, z^{\nu+1}, \lambda^{\nu+1}) - L(x^{\nu+1}, z^{\nu}, \lambda^{\nu+1}). \end{aligned} \tag{C.1}$$

We bound the three pairs of differences on the right-hand-side (RHS) of (C.1) as follows. First, due to the optimality of $x_k^{\nu+1}$ in (6.6), we introduce the general subgradient $d_k^{\nu+1}$:

$$d_k^{\nu+1} := -\big(A_k^\top \lambda_k^\nu + \rho A_k^\top (A_k x_k^{\nu+1} - z_k^{\bar{\nu}_k+1})\big) \in \partial F_k(x_k^{\nu+1}). \tag{C.2}$$

Then, since only x_k for $k \in \mathcal{A}_\nu$ is updated, we have

$$
\begin{aligned}
&L(x^\nu, z^\nu, \lambda^\nu) - L(x^{\nu+1}, z^\nu, \lambda^\nu) \\
&= \sum_{k\in\mathcal{A}_\nu} \Big(F_k(x_k^\nu) - F_k(x_k^{\nu+1}) + \lambda_k^{\nu\top} A_k(x_k^\nu - x_k^{\nu+1}) + \frac{\rho}{2}\|A_k x_k^\nu - z_k^\nu\|^2 - \frac{\rho}{2}\|A_k x_k^{\nu+1} - z_k^\nu\|^2 \Big) \\
&= \sum_{k\in\mathcal{A}_\nu} \Big(F_k(x_k^\nu) - F_k(x_k^{\nu+1}) + \lambda_k^{\nu\top} A_k(x_k^\nu - x_k^{\nu+1}) + \frac{\rho}{2}\|A_k x_k^\nu - A_k x_k^{\nu+1}\|^2 \\
&\quad + \rho\langle A_k x_k^{\nu+1} - z_k^{\bar\nu_k+1} + z_k^{\bar\nu_k+1} - z_k^\nu, A_k x_k^\nu - A_k x_k^{\nu+1}\rangle \Big) \\
&= \sum_{k\in\mathcal{A}_\nu} \Big(F_k(x_k^\nu) - F_k(x_k^{\nu+1}) + \frac{\rho}{2}\|A_k x_k^\nu - A_k x_k^{\nu+1}\|^2 + \langle A_k^\top \lambda_k^\nu + \rho A_k^\top (A_k x_k^{\nu+1} - z_k^{\bar\nu_k+1}), x_k^\nu - x_k^{\nu+1}\rangle \\
&\quad + \rho(z_k^{\bar\nu_k+1} - z_k^\nu)^\top (A_k x_k^\nu - A_k x_k^{\nu+1}) \Big) \\
&= \sum_{k\in\mathcal{A}_\nu} \Big(F_k(x_k^\nu) - F_k(x_k^{\nu+1}) - \langle d_k^{\nu+1}, x_k^\nu - x_k^{\nu+1}\rangle + \rho(z_k^{\bar\nu_k+1} - z_k^\nu)^\top (A_k x_k^\nu - A_k x_k^{\nu+1}) \Big) \\
&\quad + \frac{\rho}{2}\|A_k x_k^\nu - A_k x_k^{\nu+1}\|^2 \\
&\geq -\frac{\gamma}{2}\sum_{k\in\mathcal{A}_\nu} \|x_k^{\nu+1} - x_k^\nu\|^2 + \frac{\rho}{2}\sum_{k\in\mathcal{A}_\nu} \|A_k x_k^\nu - A_k x_k^{\nu+1}\|^2 + \rho \sum_{k\in\mathcal{A}_\nu} (z_k^{\bar\nu_k+1} - z_k^\nu)^\top (A_k x_k^\nu - A_k x_k^{\nu+1}),
\end{aligned}
\tag{C.3}
$$

where the second equality follows from the cosine rule:
$\|b+c\|^2 - \|a+c\|^2 = \|b-a\|^2 + 2\langle a+c, b-a\rangle$with $a = A_k x_k^{\nu+1}$, $b = A_k x_k^\nu$, and $c = -z_k^\nu$, and the fourth equality is due to the definition of $d_k^{\nu+1}$in (C.2). The last inequality is derived from Definition 1 under Assumption 4. Then, by Assumption 2, we have

$$
\begin{aligned}
&L(x^{\nu+1}, z^\nu, \lambda^\nu) - L(x^\nu, z^\nu, \lambda^\nu) \\
&\leq \frac{\gamma}{2}\sum_{k\in\mathcal{A}_\nu} \|x_k^{\nu+1} - x_k^\nu\|^2 - \frac{\rho}{2}\sum_{k\in\mathcal{A}_\nu} \|A_k x_k^\nu - A_k x_k^{\nu+1}\|^2 - \rho \sum_{k\in\mathcal{A}_\nu} (z_k^{\bar\nu_k+1} - z_k^\nu)^\top (A_k x_k^\nu - A_k x_k^{\nu+1}) \\
&\leq \frac{\gamma M_2^2 - \rho}{2M_2^2}\sum_{k\in\mathcal{A}_\nu} \|x_k^{\nu+1} - x_k^\nu\|^2 + \rho \sum_{k\in\mathcal{A}_\nu} (z_k^{\bar\nu_k+1} - z_k^\nu)^\top (A_k x_k^{\nu+1} - A_k x_k^\nu).
\end{aligned}
\tag{C.4}
$$

Secondly, for the λ-update, we have

$$\begin{aligned}
&L(x^{\nu+1}, z^{\nu}, \lambda^{\nu+1}) - L(x^{\nu+1}, z^{\nu}, \lambda^{\nu}) \\
&= \sum_{k\in\mathcal{A}_\nu} (\lambda_k^{\nu+1} - \lambda_k^{\nu})^\top (A_k x_k^{\nu+1} - z_k^{\nu}) \\
&= \sum_{k\in\mathcal{A}_\nu} (\lambda_k^{\nu+1} - \lambda_k^{\nu})^\top (A_k x_k^{\nu+1} - z_k^{\bar{\nu}_k+1}) + \sum_{k\in\mathcal{A}_\nu} (\lambda_k^{\nu+1} - \lambda_k^{\nu})^\top (z_k^{\bar{\nu}_k+1} - z_k^{\nu}) \\
&= \frac{1}{\rho} \sum_{k\in\mathcal{A}_\nu} \|\lambda_k^{\nu+1} - \lambda_k^{\nu}\|^2 + \sum_{k\in\mathcal{A}_\nu} (\lambda_k^{\nu+1} - \lambda_k^{\nu})^\top (z_k^{\bar{\nu}_k+1} - z_k^{\nu}).
\end{aligned} \tag{C.5}$$

The first equality is due to the fact that $\lambda_k^{\nu+1} = \lambda_k^{\nu}, \forall k \notin \mathcal{A}_\nu$, and the last equality is obtained by applying (6.7) for $k \in \mathcal{A}_\nu$.

Thirdly, for the z-update, we first look at the difference caused by updating $z_{k,l}$ (or $z_{l,k}$ for one pair of neighboring workers k and l. By Definition 2.1, (6.10) is strongly convex with respect to (w.r.t.) $z_{k,l}$ with modulus $2\rho + \alpha$. Then by Definition 2.2, we have

$$\begin{aligned}
&L'(x^{\nu+1}, z_{k,l}^{\nu}, \lambda^{\nu+1}) - L'(x^{\nu+1}, z_{k,l}^{\nu+1}, \lambda^{\nu+1}) \\
&= \left(- (\lambda_{k,l}^{\nu+1} + \lambda_{l,k}^{\nu+1}) z_{k,l}^{\nu} + \frac{\rho}{2} \|A_{k,l} x_k^{\nu+1} - z_{k,l}^{\nu}\|^2 + \frac{\rho}{2} \|A_{l,k} x_l^{\nu+1} - z_{k,l}^{\nu}\|^2 + \frac{\alpha}{2} \|z_{k,l}^{\nu} - z_{k,l}^{\nu}\|^2 \right) \\
&\quad - \left(- (\lambda_{k,l}^{\nu+1} + \lambda_{l,k}^{\nu+1}) z_{k,l}^{\nu+1} + \frac{\rho}{2} \|A_{k,l} x_k^{\nu+1} - z_{k,l}^{\nu+1}\|^2 + \frac{\rho}{2} \|A_{l,k} x_l^{\nu+1} - z_{k,l}^{\nu+1}\|^2 + \frac{\alpha}{2} \|z_{k,l}^{\nu+1} - z_{k,l}^{\nu}\|^2 \right) \\
&\geq \left(\frac{\partial L'(x^{\nu+1}, z_{k,l}^{\nu+1}, \lambda^{\nu+1})}{\partial z_{k,l}^{\nu+1}} (z_{k,l}^{\nu} - z_{k,l}^{\nu+1}) + (\rho + \frac{\alpha}{2}) \|z_{k,l}^{\nu} - z_{k,l}^{\nu+1}\|^2 \right) \\
&= (\rho + \frac{\alpha}{2}) \|z_{k,l}^{\nu} - z_{k,l}^{\nu+1}\|^2.
\end{aligned} \tag{C.6}$$

The last equality in (C.6) holds because $\partial L'(z_{k,l}^{\nu+1}) = 0$ due to the optimality condition of (6.8). For any $z_{k,l} \in z$, we have

$$L(x^{\nu+1}, z_{k,l}^{\nu+1}, \lambda^{\nu+1}) - L(x^{\nu+1}, z_{k,l}^{\nu}, \lambda^{\nu+1}) \leq -(\rho + \alpha) \|z_{k,l}^{\nu+1} - z_{k,l}^{\nu}\|^2. \tag{C.7}$$

Summing up the LHS of (C.7), we have

$$L(x^{\nu+1}, z^{\nu+1}, \lambda^{\nu+1}) - L(x^{\nu+1}, z^{\nu}, \lambda^{\nu+1}) \leq -(\rho + \alpha) \|z^{\nu+1} - z^{\nu}\|^2. \tag{C.8}$$

By substituting (C.4), (C.5) and (C.8) into (C.1), we obtain

$$\begin{aligned}
&L(x^{\nu+1}, z^{\nu+1}, \lambda^{\nu+1}) - L(x^{\nu}, z^{\nu}, \lambda^{\nu}) \\
&\leq \frac{\gamma M_2^2 - \rho}{2M_2^2} \sum_{k\in\mathcal{A}_\nu} \|x_k^{\nu+1} - x_k^{\nu}\|^2 - (\rho+\alpha)\|z^{\nu+1} - z^{\nu}\|^2 \\
&\quad + \frac{1}{\rho} \sum_{k\in\mathcal{A}_\nu} \|\lambda_k^{\nu+1} - \lambda_k^{\nu}\|^2 + \sum_{k\in\mathcal{A}_\nu} (\lambda_k^{\nu+1} - \lambda_k^{\nu})^\top (z_k^{\bar{\nu}_k+1} - z_k^{\nu}) \\
&\quad + \rho \sum_{k\in\mathcal{A}_\nu} (z_k^{\bar{\nu}_k+1} - z_k^{\nu})^\top (A_k x_k^{\nu+1} - A_k x_k^{\nu}) \\
&\leq \frac{\gamma M_2^2 - \rho}{2M_2^2} \sum_{k\in\mathcal{A}_\nu} \|x_k^{\nu+1} - x_k^{\nu}\|^2 - (\rho+\alpha)\|z^{\nu+1} - z^{\nu}\|^2 \\
&\quad + (\frac{1}{\rho} + \frac{1}{2}) \sum_{k\in\mathcal{A}_\nu} \|\lambda_k^{\nu+1} - \lambda_k^{\nu}\|^2 + \frac{1}{2} \sum_{k\in\mathcal{A}_\nu} \|z_k^{\bar{\nu}_k+1} - z_k^{\nu}\|^2 \\
&\quad + \rho M_2^4 \sum_{k\in\mathcal{A}_\nu} \|z_k^{\bar{\nu}_k+1} - z_k^{\nu}\|^2 + \frac{\rho}{4M_2^4} \sum_{k\in\mathcal{A}_\nu} \|A_k x_k^{\nu+1} - A_k x_k^{\nu}\|^2 \\
&\leq (\frac{\gamma}{2} - \frac{\rho}{4M_2^2}) \sum_{k\in\mathcal{A}_\nu} \|x_k^{\nu+1} - x_k^{\nu}\|^2 - (\rho+\alpha)\|z^{\nu+1} - z^{\nu}\|^2 \\
&\quad + (\frac{1}{\rho} + \frac{1}{2}) \sum_{k\in\mathcal{A}_\nu} \|\lambda_k^{\nu+1} - \lambda_k^{\nu}\|^2 + \frac{2\rho M_2^4 + 1}{2} \sum_{k\in\mathcal{A}_\nu} \|z_k^{\bar{\nu}_k+1} - z_k^{\nu}\|^2.
\end{aligned} \tag{C.9}$$

The second inequality is obtained by applying Young's inequality; i.e., $ab \leq \frac{a^2}{2\epsilon} + \frac{\epsilon b^2}{2}$, with $\epsilon = 1$ and $\epsilon = \frac{1}{2M_2^4}$, to term $(\lambda_k^{\nu+1} - \lambda_k^{\nu})^\top (z_k^{\bar{\nu}_k+1} - z_k^{\nu})$and $\rho(z_k^{\bar{\nu}_k+1} - z_k^{\nu})^\top (A_k x_k^{\nu+1} - A_k x_k^{\nu})$, respectively. The last inequality holds under Assumption 2.

We further bound $\|\lambda_k^{\nu+1} - \lambda_k^{\nu}\|^2$as follows. From the optimality condition of (6.6) and (6.7), for $k \in \mathcal{A}_\nu$, we have

$$\begin{aligned}
\mathbf{0} &= \partial F_k(x_k^{\nu+1}) + A_k^\top \left(\lambda_k^{\nu} + \rho_k (A_k x_k^{\nu+1} - z_k^{\bar{\nu}_k+1}) \right) \\
&= \partial F_k(x_k^{\nu+1}) + A_k^\top \lambda_k^{\nu+1}.
\end{aligned} \tag{C.10}$$

For $\forall k \notin \mathcal{A}_\nu$, let $\bar{\nu}_k < \nu$ denote the last iteration number for which region k updated. Then, $\partial F_k(x_k^{\bar{\nu}_k}) + A_k^\top \lambda_k^{\bar{\nu}_k+1} = \mathbf{0}$. Since $x_k^{\bar{\nu}_k+1} = \cdots = x_k^{\nu} = x_k^{\nu+1}$and $\lambda_k^{\bar{\nu}_k+1} = \cdots = \lambda_k^{\nu} = \lambda_k^{\nu+1}$, we can conclude that

$$\partial F_k(x_k^{\nu+1}) + A_k^\top \lambda_k^{\nu+1} = \mathbf{0}, \qquad \forall k \text{ and } \forall \nu. \tag{C.11}$$

By Assumption 3, Assumption 5 and (C.11), we can bound

$$\begin{aligned}
&\|\lambda_k^{\nu+1} - \lambda_k^{\nu}\|^2 = \|(A_k A_k^\top)^{-1} A_k (A_k^\top \lambda_k^{\nu+1} - A_k^\top \lambda_k^{\nu})\|^2 \\
&= \|(A_k A_k^\top)^{-1} A_k \big(\partial F_k(x_k^{\nu+1}) - \partial F_k(x_k^{\nu})\big)\|^2 \\
&= \|B_k \big(\partial F_k(x_k^{\nu+1}) - \partial F_k(x_k^{\nu})\big)\|^2 \\
&\leq \sigma_{\max}(B_k^\top B_k) \|\partial F_k(x_k^{\nu+1}) - \partial F_k(x_k^{\nu})\|^2 \\
&\leq C M_1^2 \|x_k^{\nu+1} - x_k^{\nu}\|^2.
\end{aligned} \tag{C.12}$$

Substituting (C.12) into (C.9), we obtain Lemma 1. □

Appendix D

Proof of Lemma 2

Proof of Lemma 2.

$$\begin{aligned}
&\sum_{\phi=1}^{\nu}\sum_{k\in\mathcal{A}_\nu}\|z_k^{\phi}-z_k^{\bar{\phi}_k+1}\|^2=\sum_{\phi=1}^{\nu}\sum_{k\in\mathcal{A}_\nu}\Big\|\sum_{\iota=\bar{\phi}_k+1}^{\phi-1}(z_k^{\iota+1}-z_k^{\iota})\Big\|^2\\
&\leq\sum_{\phi=1}^{\nu}\sum_{k\in\mathcal{A}_\nu}(\phi-\bar{\phi}_k-1)\sum_{\iota=\bar{\phi}_k+1}^{\phi-1}\|z_k^{\iota+1}-z_k^{\iota}\|^2\\
&\leq\sum_{\phi=1}^{\nu}\sum_{k\in\mathcal{A}_\nu}(\omega-1)\sum_{\iota=\phi-\omega+1}^{\phi-1}\|z_k^{\iota+1}-z_k^{\iota}\|^2 \qquad \text{(D.1)}\\
&\leq(\omega-1)^2\sum_{\phi=1}^{\nu}\sum_{k=1}^{K}\|z_k^{\phi+1}-z_k^{\phi}\|^2\\
&\leq(\omega-1)^2\sum_{\phi=1}^{\nu}\|z^{\phi+1}-z^{\phi}\|^2,
\end{aligned}$$

where for the second inequality, we apply (6.9); i.e., $\phi-\omega\leq\bar{\phi}_k<\phi$. The third inequality holds due to the observation that in the summation $\sum_{\phi=1}^{\nu}\sum_{\iota=\phi-\omega+1}^{\phi-1}\|z_k^{\iota+1}-z_k^{\iota}\|^2$, each $\|z_k^{\iota+1}-z_k^{\iota}\|$ with $\iota=1,2,...,\nu$ appears no more than $\omega-1$ times. □

Bibliography

[1] M. Shahidehpour and Y. Wang, *Communication and control in electric power systems: applications of parallel and distributed processing.* John Wiley & Sons, 2004.

[2] M. B. Cain, R. P. O'neill, and A. Castillo, "History of optimal power flow and formulations," *Federal Energy Regulatory Commission*, 2012.

[3] A. Lam, B. Zhang, and D. Tse, "Distributed algorithms for optimal power flow problem," in *IEEE 51st Annual Conference on Decision and Control (CDC)*, 2012, pp. 430–437.

[4] A. J. Conejo, E. Castillo, R. García-Bertrand, and R. Mínguez, *Decomposition techniques in mathematical programming: engineering and science applications.* Springer Berlin, 2006.

[5] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[6] A. M. Geoffrion, *Lagrangean relaxation for integer programming.* Springer, 1974.

[7] G. Cohen, "Auxiliary problem principle and decomposition of optimization problems," *Journal of optimization Theory and Applications*, vol. 32, no. 3, pp. 277–305, 1980.

[8] R. Baldick, B. H. Kim, C. Chase, and Y. Luo, "A fast distributed implementation of optimal power flow," *IEEE Transactions on Power Systems*, vol. 14, no. 3, pp. 858–864, 1999.

[9] B. H. Kim and R. Baldick, "A comparison of distributed optimal power flow algorithms," *IEEE Transactions on Power Systems*, vol. 15, no. 2, pp. 599–604, 2000.

[10] T. Erseghe, "A distributed approach to the opf problem," *EURASIP Journal on Advances in Signal Processing*, vol. 2015, no. 1, pp. 1–13, 2015.

[11] A. J. Conejo, F. J. Nogales, and F. J. Prieto, "A decomposition procedure based on approximate newton directions," *Mathematical programming*, vol. 93, no. 3, pp. 495–515, 2002.

[12] F. J. Nogales, F. J. Prieto, and A. J. Conejo, "A decomposition methodology applied to the multi-area optimal power flow problem," *Annals of operations research*, vol. 120, no. 1-4, pp. 99–116, 2003.

[13] G. Hug-Glanzmann and G. Andersson, "Decentralized optimal power flow control for over-

lapping areas in power systems," *IEEE Transactions on Power Systems*, vol. 24, no. 1, pp. 327–336, 2009.

[14] M. Shahidehopour and Y. Fu, "Benders decomposition: applying benders decomposition to power systems," *IEEE Power and Energy Magazine*, vol. 3, no. 2, pp. 20–21, 2005.

[15] D. K. Molzahn, F. Drfler, H. Sandberg, S. H. Low, S. Chakrabarti, R. Baldick, and J. Lavaei, "A survey of distributed optimization and control algorithms for electric power systems," *IEEE Transactions on Smart Grid*, vol. 8, no. 6, pp. 2941–2962, Nov 2017.

[16] D. Phan and J. Kalagnanam, "Some efficient optimization methods for solving the security-constrained optimal power flow problem," *IEEE Transactions on Power Systems*, vol. 29, no. 2, pp. 863–872, 2014.

[17] M. Kraning, E. Chu, J. Lavaei, and S. Boyd, "Dynamic network energy management via proximal message passing," *Foundations and Trends in Optimization*, vol. 1, no. 2, pp. 73–126, 2014.

[18] T. Erseghe and S. Tomasin, "Power flow optimization for smart microgrids by sdp relaxation on linear networks," *IEEE Transactions on Smart Grid*, vol. 4, no. 2, pp. 751–762, 2013.

[19] E. Dall'Anese, H. Zhu, and G. Giannakis, "Distributed optimal power flow for smart microgrids," *IEEE Transactions on Smart Grid*, vol. 4, no. 3, pp. 1464–1475, 2013.

[20] S. Magnusson, P. Weeraddana, and C. Fischione, "A distributed approach for the optimal power-flow problem based on ADMM and sequential convex approximations," *IEEE Transactions on Control of Network Systems*, vol. 2, no. 3, pp. 238–253, 2015.

[21] J. Lavaei and S. H. Low, "Zero duality gap in optimal power flow problem," *IEEE Transactions on Power Systems*, vol. 27, no. 1, pp. 92–107, 2012.

[22] R. Madani, S. Sojoudi, and J. Lavaei, "Convex relaxation for optimal power flow problem: Mesh networks," *IEEE Transactions on Power Systems*, vol. 30, no. 1, pp. 199–211, 2015.

[23] T. Erseghe, "Distributed optimal power flow using ADMM," *IEEE Transactions on Power Systems*, vol. 29, no. 5, pp. 2370–2380, 2014.

[24] P. Šulc, S. Backhaus, and M. Chertkov, "Optimal distributed control of reactive power via the alternating direction method of multipliers," *IEEE Transactions on Energy Conversion*, vol. 29, no. 4, pp. 968–977, 2014.

[25] Q. Peng and S. H. Low, "Distributed algorithm for optimal power flow on a radial network," in *IEEE 53rd Annual Conference on Decision and Control (CDC)*, 2014, pp. 167–172.

[26] J. Undrill and H. Happ, "Automatic sectionalization of power system networks for network solutions," *IEEE Transactions on Power Apparatus and Systems*, no. 1, pp. 46–53, 1971.

[27] A. Sangiovanni-Vincentelli, L.-K. Chen, and L. Chua, "An efficient heuristic cluster algorithm

for tearing large-scale networks," *IEEE Transactions on Circuits and Systems*, vol. 24, no. 12, pp. 709–717, 1977.

[28] M. Irving and M. Sterling, "Optimal network tearing using simulated annealing," in *Generation, Transmission and Distribution, IEE Proceedings C*, vol. 137, no. 1, 1990, pp. 69–72.

[29] H. Ding, A. El-Keib, and R. Smith, "Optimal clustering of power networks using genetic algorithms," *Electric Power Systems Research*, vol. 30, no. 3, pp. 209–214, 1994.

[30] M. Mori and O. Matsuzaki, "A rule-based tabu search technique for power system decomposition," in *IEEE Power Engineering Society Summer Meeting*, vol. 4, 2000, pp. 1990–1995.

[31] G. Angeline Ezhilarasi and K. Swarup, "Network partitioning using harmony search and equivalencing for distributed computing," *Journal of Parallel and Distributed Computing*, vol. 72, no. 8, pp. 936–943, 2012.

[32] S. Yusof, G. Rogers, and R. Alden, "Slow coherency based network partitioning including load buses," *IEEE Transactions on Power Systems*, vol. 8, no. 3, pp. 1375–1382, 1993.

[33] E. Cotilla-Sanchez, P. D. Hines, C. Barrows, S. Blumsack, and M. Patel, "Multi-attribute partitioning of power networks based on electrical distance," *IEEE Transactions on Power Systems*, vol. 28, no. 4, pp. 4979–4987, 2013.

[34] Q. Zhao, K. Sun, D.-Z. Zheng, J. Ma, and Q. Lu, "A study of system splitting strategies for island operation of power system: a two-phase method based on obdds," *IEEE Transactions on Power Systems*, vol. 18, no. 4, pp. 1556–1565, 2003.

[35] H. Mehrjerdi, S. Lefebvre, D. Asber, and M. Saad, "Graph partitioning of power network for emergency voltage control," in *IEEE Asian Control Conference (ASCC)*, 2013.

[36] J. T. Allison, M. Kokkolaras, and P. Y. Papalambros, "Optimal partitioning and coordination decisions in decomposition-based design optimization," *Journal of Mechanical Design*, vol. 131, p. 081008, 2009.

[37] S. Cvijic and M. D. Ilic, "Optimal clustering for efficient computations of contingency effects in large regional power systems," in *IEEE Power and Energy Society General Meeting*, 2012.

[38] Z. Fan, P. Kulkarni, S. Gormus, C. Efthymiou, G. Kalogridis, M. Sooriyabandara, Z. Zhu, S. Lambotharan, and W. H. Chin, "Smart grid communications: Overview of research challenges, solutions, and standardization activities," *Communications Surveys & Tutorials, IEEE*, vol. 15, no. 1, pp. 21–38, 2013.

[39] Q. Yang, J. A. Barria, and T. C. Green, "Communication infrastructures for distributed control of power distribution networks," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 2, pp. 316–327, 2011.

[40] C.-H. Lo and N. Ansari, "Decentralized controls and communications for autonomous distri-

bution networks in smart grid," 2013.

[41] M. Shahraeini, M. H. Javidi, and M. S. Ghazizadeh, "Comparison between communication infrastructures of centralized and decentralized wide area measurement systems," *IEEE Transactions on Smart Grid*, vol. 2, no. 1, pp. 206–211, 2011.

[42] A. Ghasemkhani, H. Monsef, A. Rahimi-Kian, and A. Anvari-Moghaddam, "Optimal design of a wide area measurement system for improvement of power network monitoring using a dynamic multiobjective shortest path algorithm," *IEEE Systems Journal*, vol. PP, no. 99, pp. 1–12, 2015.

[43] X. Xiong, J. Tan, and X. Lin, "Study on communication architecture design of wide-area measurement system," *IEEE Transactions on Power Delivery*, vol. 28, no. 3, pp. 1542–1547, 2013.

[44] M. Chenine and L. Nordström, "Modeling and simulation of wide-area communication for centralized pmu-based applications," *IEEE Transactions on Power Delivery*, vol. 26, no. 3, pp. 1372–1380, 2011.

[45] Y. Deng, H. Lin, A. G. Phadke, S. Shukla, J. S. Thorp, and L. Mili, "Communication network modeling and simulation for wide area measurement applications," in *IEEE Innovative Smart Grid Technologies (ISGT) Conference*, 2012.

[46] I. Ali and S. Hussain, "Communication design for energy management automation in microgrid," *IEEE Transactions on Smart Grid*, vol. PP, no. 99, pp. 1–1, 2016.

[47] Z. Cai, M. Yu, M. Steurer, H. Li, and Y. Dong, "A network model for the real-time communications of a smart grid prototype," *Journal of Network and Computer Applications*, vol. 59, pp. 264–273, 2016.

[48] K. Hopkinson, X. Wang, R. Giovanini, J. Thorp, K. Birman, and D. Coury, "Epochs: a platform for agent-based electric power and communication simulation built from commercial off-the-shelf components," *IEEE Transactions on Power Systems*, vol. 21, no. 2, pp. 548–558, 2006.

[49] V. Liberatore and A. Al-Hammouri, "Smart grid communication and co-simulation," in *Energytech, IEEE*, 2011, pp. 1–5.

[50] H. Lin, S. S. Veda, S. S. Shukla, L. Mili, and J. Thorp, "Geco: Global event-driven co-simulation framework for interconnected power system and communication network," *IEEE Transactions on Smart Grid*, vol. 3, no. 3, pp. 1444–1456, 2012.

[51] Z. Zhang and M.-Y. Chow, "Convergence analysis of the incremental cost consensus algorithm under different communication network topologies in a smart grid," *IEEE Transactions on Power Systems*, vol. 27, no. 4, pp. 1761–1768, 2012.

[52] J. Wang and N. Elia, "A control perspective for centralized and distributed convex optimization," in *IEEE Decision and Control and European Control Conference (CDC-ECC)*, 2011.

[53] K. Tsianos, M. G. Rabbat *et al.*, "Distributed consensus and optimization under communication delays," in *2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2011, pp. 974–982.

[54] A. Nedić and A. Ozdaglar, "Convergence rate for consensus with delays," *Journal of Global Optimization*, vol. 47, no. 3, pp. 437–456, 2010.

[55] Y. P. Tian and C. L. Liu, "Consensus of multi-agent systems with diverse input and communication delays," *IEEE Transactions on Automatic Control*, vol. 53, no. 9, pp. 2122–2128, Oct 2008.

[56] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*. Prentice hall Englewood Cliffs, NJ, 1989, vol. 23.

[57] C. Dwork, N. Lynch, and L. Stockmeyer, "Consensus in the presence of partial synchrony," *Journal of the ACM (JACM)*, vol. 35, no. 2, pp. 288–323, 1988.

[58] Z. Peng, Y. Xu, M. Yan, and W. Yin, "Arock: an algorithmic framework for asynchronous parallel coordinate updates," *SIAM Journal on Scientific Computing*, vol. 38, no. 5, pp. A2851–A2879, 2016.

[59] I. Aravena and A. Papavasiliou, "A distributed asynchronous algorithm for the two-stage stochastic unit commitment problem," in *2015 IEEE Power Energy Society General Meeting*, July 2015, pp. 1–5.

[60] A. Abboud, R. Couillet, M. Debbah, and H. Siguerdidjane, "Asynchronous alternating direction method of multipliers applied to the direct-current optimal power flow problem," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 7764–7768.

[61] H. K. Nguyen, A. Khodaei, and Z. Han, "Distributed algorithms for peak ramp minimization problem in smart grid," in *IEEE International Conference on Smart Grid Communications (SmartGridComm)*, 2016, pp. 174–179.

[62] C.-Y. Chang, J. Cortés, and S. Martínez, "A scheduled-asynchronous distributed optimization algorithm for the optimal power flow problem," in *American Control Conference (ACC), 2017*. IEEE, 2017, pp. 3968–3973.

[63] D. Davis and W. Yin, "Convergence rate analysis of several splitting schemes," in *Splitting Methods in Communication, Imaging, Science, and Engineering*. Springer, 2016, pp. 115–163.

[64] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, "On the linear convergence of the ADMM in decentralized consensus optimization," *IEEE Transactions on Signal Processing*, vol. 62, no. 7, pp. 1750–1761, 2014.

[65] M. Hong and Z.-Q. Luo, "On the linear convergence of the alternating direction method of multipliers," *Mathematical Programming*, vol. 162, no. 1-2, pp. 165–199, 2017.

[66] P. Giselsson and S. Boyd, "Linear convergence and metric selection for douglas-rachford splitting and admm," *IEEE Transactions on Automatic Control*, vol. 62, no. 2, pp. 532–544, Feb 2017.

[67] D. Davis and W. Yin, "Faster convergence rates of relaxed peaceman-rachford and admm under regularity assumptions," *Mathematics of Operations Research*, 2017.

[68] A. U. Raghunathan and S. Di Cairano, "Alternating direction method of multipliers for strictly convex quadratic programs: Optimal parameter selection," in *American Control Conference (ACC), 2014*. IEEE, 2014, pp. 4324–4329.

[69] R. Zhang and J. Kwok, "Asynchronous distributed admm for consensus optimization," in *International Conference on Machine Learning*, 2014, pp. 1701–1709.

[70] T. H. Chang, W. C. Liao, M. Hong, and X. Wang, "Asynchronous distributed admm for large-scale optimization – part ii: Linear convergence analysis and numerical performance," *IEEE Transactions on Signal Processing*, vol. 64, no. 12, pp. 3131–3144, June 2016.

[71] E. Wei and A. Ozdaglar, "On the o (1= k) convergence of asynchronous distributed alternating direction method of multipliers," in *Global conference on signal and information processing (GlobalSIP), 2013 IEEE*. IEEE, 2013, pp. 551–554.

[72] M. Hong, Z.-Q. Luo, and M. Razaviyayn, "Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems," *SIAM Journal on Optimization*, vol. 26, no. 1, pp. 337–364, 2016.

[73] Y. Wang, W. Yin, and J. Zeng, "Global convergence of admm in nonconvex nonsmooth optimization," *arXiv preprint arXiv:1511.06324*, 2016.

[74] S. Magnússon, P. C. Weeraddana, M. G. Rabbat, and C. Fischione, "On the convergence of alternating direction lagrangian methods for nonconvex structured optimization problems," *IEEE Transactions on Control of Network Systems*, vol. 3, no. 3, pp. 296–309, 2016.

[75] T. H. Chang, M. Hong, W. C. Liao, and X. Wang, "Asynchronous distributed admm for large-scale optimization – part i: Algorithm and convergence analysis," *IEEE Transactions on Signal Processing*, vol. 64, no. 12, pp. 3118–3130, June 2016.

[76] S. Kumar, R. Jain, and K. Rajawat, "Asynchronous optimization over heterogeneous networks via consensus admm," *IEEE Transactions on Signal and Information Processing over*

Networks, vol. 3, no. 1, pp. 114–129, 2017.

[77] M. Hong, "A distributed, asynchronous and incremental algorithm for nonconvex optimization: An admm approach," *IEEE Transactions on Control of Network Systems*, vol. PP, no. 99, pp. 1–1, 2017.

[78] L. Cannelli, F. Facchinei, V. Kungurtsev, and G. Scutari, "Asynchronous parallel algorithms for nonconvex big-data optimization. part ii: Complexity and numerical results," *arXiv preprint arXiv:1701.04900*, 2017.

[79] R. D. Zimmerman, C. E. Murillo-Sanchez, and R. J. Thomas, "Matpower: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Transactions on Power Systems*, vol. 26, no. 1, pp. 12–19, Feb 2011.

[80] E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson, "Optimal parameter selection for the alternating direction method of multipliers (ADMM): quadratic problems," *IEEE Transactions on Automatic Control*, vol. 60, no. 3, pp. 644–658, 2015.

[81] M. L. Overton and R. S. Womersley, "On minimizing the special radius of a nonsymmetric matrix function: Optimality conditions and duality theory," *SIAM Journal on Matrix Analysis and Applications*, vol. 9, no. 4, pp. 473–498, 1988.

[82] A. Pothen, H. D. Simon, and K.-P. Liou, "Partitioning sparse matrices with eigenvectors of graphs," *SIAM Journal on Matrix Analysis and Applications*, vol. 11, no. 3, pp. 430–452, 1990.

[83] A. R. Gourlay and G. A. Watson, *Computational methods for matrix eigenproblems*. Wiley London, 1973.

[84] A. Y. Ng, M. I. Jordan, Y. Weiss *et al.*, "On spectral clustering: Analysis and an algorithm," *Advances in neural information processing systems*, vol. 2, pp. 849–856, 2002.

[85] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "Matpower: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Transactions on power systems*, vol. 26, no. 1, pp. 12–19, 2011.

[86] K. Holmstrm, "Tomlab – an environment for solving optimization problems in matlab," in *Proceedings for the Nordic MATLAB Conference*, 1997, pp. 27–28.

[87] S. Magnsson, P. C. Weeraddana, M. G. Rabbat, and C. Fischione, "On the convergence of alternating direction lagrangian methods for nonconvex structured optimization problems," *IEEE Transactions on Control of Network Systems*, vol. 3, no. 3, pp. 296–309, 2016.

[88] J. Guo, G. Hug, and O. K. Tonguz, "Intelligent partitioning in distributed optimization of electric power systems," *IEEE Transactions on Smart Grid*, vol. 7, no. 3, pp. 1249–1258, 2016.

[89] L. Xie and M. D. Ilić, "Model predictive dispatch in electric energy systems with intermittent resources," in *IEEE International Conference on Systems, Man and Cybernetics*, 2008.

[90] K. Baker, G. Hug, and X. Li, "Optimal integration of intermittent energy sources using distributed multi-step optimization," in *Power and Energy Society General Meeting*, 2012.

[91] P. D. Christofides, R. Scattolini, D. M. de la Pena, and J. Liu, "Distributed model predictive control: A tutorial review and future research directions," *Computers & Chemical Engineering*, vol. 51, pp. 21–41, 2013.

[92] R. R. Negenborn and J. Maestre, "Distributed model predictive control: An overview of features and research opportunities," in *IEEE 11th International Conference on Networking, Sensing and Control (ICNSC)*, 2014.

[93] A. Elaiw, X. Xia, and A. Shehata, "Application of model predictive control to optimal dynamic dispatch of generation with emission limitations," *Electric power systems research*, vol. 84, no. 1, pp. 31–44, 2012.

[94] A. N. Venkat, I. A. Hiskens, J. B. Rawlings, and S. J. Wright, "Distributed mpc strategies with application to power system automatic generation control," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 6, pp. 1192–1206, 2008.

[95] A. G. Beccuti, T. Geyer, and M. Morari, "Temporal lagrangian decomposition of model predictive control for hybrid systems," in *IEEE Conference on Decision and Control*, 2004.

[96] Y. Wakasa, M. Arakawa, K. Tanaka, and T. Akashi, "Decentralized model predictive control via dual decomposition," in *IEEE Conference on Decision and Control*, 2008.

[97] M. Kuzlu and M. Pipattanasomporn, "Assessment of communication technologies and network requirements for different smart grid applications," in *IEEE Innovative Smart Grid Technologies (ISGT) Conference*, 2013.

[98] "Standards and technology adoption case study: Inter-control center protocol (iccp/tase.2)," *EPRI Technical Report*, 2012.

[99] Y.-J. Kim, M. Thottan, V. Kolesnikov, and W. Lee, "A secure decentralized data-centric information infrastructure for smart grid," *IEEE Communications Magazine*, vol. 48, no. 11, pp. 58–65, 2010.

[100] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.