Master Program 2013-2015
Electrical Engineering

Distributed Convex Optimization

A Study on the Primal-Dual Method of Multipliers

Delft University of Technology

He Ming Zhang, Guoqiang Zhang, Richard Heusdens

TU Delft Delft University of Technology

Challenge the future

Abstract

The Primal-Dual Method of Multipliers (PDMM) is a new algorithm that solves convex optimization problems in a distributed manner.

This study focuses on the convergence behavior of the PDMM. For a deeper understanding, the PDMM algorithm was applied to distributed averaging and distributed dictionary learning problems. The results were compared to those of other state-of-the-art algorithms. The experiments show that the PDMM algorithm not only has a fast convergence rate but also robust performance against transmission failures in the network.

Furthermore, on the basis of these experiments, the convergence rate of the PDMM was analyzed. Different attempts at proving the linear convergence rate were carried out. As a result, the linear convergence rate has been proven under certain conditions.

Keywords: convex optimization, distributed signal processing, PDMM, ADMM

Technical report:

TU Delft

Delft University of Technology

Contents

List of Figures

Chapter 1

Introduction

The convex optimization problem is a fundamental problem that can be found in divergent applications. Recently, along with the development of large applications in areas such as big data and smart grids, the optimization problems have tended to be decentralized. In a decentralized scenario, a fusion center which collects and spreads information is lacking. Therefore, each node in the network only communicates with adjacent nodes and then updates the local variables accordingly. Through this strategy, all of the nodes jointly optimize the problem in an iterative manner.

Although the alternating direction method of multipliers (ADMM) [1] has been developed over the course of many years, it has become popular recently because of its applicability to decentralized optimization problems. It is favored for solving different large-scale problems such as convolutional sparse coding[2] and model predictive consensus problem [3]. Besides the standard ADMM algorithm, several variants have also been developed, for example [4] which proposes an asynchronous updating scheme for the ADMM. Furthermore in [5], a Jacobi version of the updating scheme is applied to the dictionary learning problem.

In addition to the empirical performance of the ADMM, many studies also focus on the analysis of its convergence rate. In [6], an ADMM with restricted stepsizes is proven to have a linear convergence rate for certain types of non-strongly convex objective functions. While in [7], under the strongly convex assumption, it can be seen that the ADMM has a linear convergence rate for centralized problems. A recent paper [8] extends the method therein to decentralized consensus problems.

In contrast to the ADMM, the primal-dual method of multipliers (PDMM) [9][10][11][1] is a relatively new algorithm that is still under development. To our knowledge, we are still far from a full understanding of this algorithm, compared with other algorithms such as the ADMM. In [11] particularly, the PDMM is modified for optimization over graphs, where each edge between the nodes has a linear equality constraint. Above that, a convergence rate of $O(1/k)$ has also been proved in the same paper.

This study is mainly based on the research in [11] and is focused on two main directions. The first one involves the implementing of the PDMM and comparing it with others. In this connection we studied two cases, namely the distributed averaging problem and the distributed dictionary learning problem. The former is a simple case which can be used as a good indication, while the latter is more complex and has many applications in the distributed signal processing area. Besides illustrating the performance of the PDMM in applications, we aim at gaining more

[1] In these papers, the former name "Bi-Alternating Direction Method of Multipliers" was used.

knowledge on the algorithm. Based on this empirical knowledge, we would like to analyze its convergence rate theoretically, which is the second direction.

This thesis is constructed as follows. In Chapter 2, we define the problem explored in this study. Chapter 3 introduces the background knowledge that is used in this study. Chapters 4 and 5 discuss two applications, namely the distributed averaging problem and the distributed dictionary learning problem, where the PDMM algorithm is compared with other state-of-the-art algorithms. Based on the results shown in Chapters 4 and 5, the convergence rate of the PDMM algorithm is analyzed in Chapter 6. At the end, Chapter 7 concludes the thesis.

Chapter 2

Problem Statement

The main goal of this study is to discover the convergence behavior of the PDMM. The study consists of two steps. In the first step, we focus on its convergence behavior from the application point of view, where two case studies were carried out. In the second step, we attempted to analyze its convergence behavior theoretically, based on the empirical results from the previous step.

2.1 Case Studies

In the case studies, we would like to compare the convergence behavior of the PDMM with that of other algorithms. Since the PDMM has both synchronous and asynchronous updating schemes, we divided the algorithms into two groups and compared them:

1. Synchronous algorithms:
   synchronous PDMM, decentralized ADMM, synchronous Jacobi ADMM

2. Asynchronous algorithms:
   asynchronous PDMM, (modified) asynchronous ADMM, asynchronous Jacobi ADMM

The results of those algorithms are evaluated in connection with the following:

1. Whether an algorithm converges in the application

2. How fast the convergence rate is

3. Whether the convergence is robust

To test the robustness of these algorithms, we assume that each transmission within the network has a package loss rate, which is defined as the possibility that the transmission will fail. Both groups of algorithms were implemented in the package loss scenario and the performances were evaluated in both cases.

2.2 Convergence Analysis

On the basis of the results obtained from case studies, we would like to mathematically prove its convergence rate according to these results. The analysis in this study adheres strictly to synchronous PDMM without package loss.

Chapter 3

Preliminaries

Notations

In this section we introduce the notations used in the remainder of the thesis. Vectors are denoted as bold small letters, while matrices are denoted as capital letters.

Given a real matrix A, $\sigma_{max}(A)$ and $\sigma_{min}(A)$ denote the largest and the smallest singular values of A, respectively. Specifically, $\tilde{\sigma}_{min}(A)$ denotes the smallest non-zero singular values of A. Furthermore, given a real square matrix B, $\lambda_{max}(B)$ and $\lambda_{min}(B)$ stand for the largest and the smallest singular values of B, respectively.

The notation $C \succ 0$ (or $C \succeq 0$) means that C is a positive definite matrix (or a positive semi-definite matrix). The Kronecker product is denoted as $\otimes$.

The Euclidean norm of a vector $\mathbf{a}$ is denoted as $\|\mathbf{a}\|$. Other norms are denoted as subscripts, i.e. $\|\mathbf{a}\|_p$ stands for the p-norm of $\mathbf{a}$, whereas the D-norm ($D \succeq 0$) of $\mathbf{a}$ is denoted as $\|\mathbf{a}\|_D$. The definition of a D-norm is

$$\|\mathbf{a}\|_D = \sqrt{\mathbf{a}^T D \mathbf{a}}. \tag{3.1}$$

For an optimization problem over $\mathbf{x}$, we use $\mathbf{x}^*$ to denote the optimal value. In a graph, $\mathcal{N}(\cdot)$ denotes the neighborhood. A node $j \in \mathcal{N}(i)$ if it is connected to node i. The edge $e = (i, j)$ is the link between node i and node j. We denote $i \in \mathcal{N}(e)$ if node i is connected to edge e.

3.1 Optimization Problems

In this study, we focus on the equality-constrained convex optimization problem. Define closed and proper $f : \mathbb{R}^{MN\times 1} \to \mathbb{R}$, $\mathbf{x} \in \mathbb{R}^{MN\times 1}$, $A \in \mathbb{R}^{P\times MN}$ and $\mathbf{b} \in \mathbb{R}^{P\times 1}$, an equality-constrained convex optimization problem (the primal problem) can be formulated as

$$\begin{aligned} &\min_{\mathbf{x}} f(\mathbf{x}) \\ &\text{s.t. } A\mathbf{x} = \mathbf{b}. \end{aligned} \tag{3.2}$$

The Lagrangian for (3.2) is

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda^T (A\mathbf{x} - \mathbf{b}). \tag{3.3}$$

The dual problem is therefore

$$\begin{aligned}\max_{\lambda} g(\lambda) &= \max_{\lambda} \inf_{\mathbf{x}} L(\mathbf{x},\lambda) \\ &= \max_{\lambda} -f^*(-A^T\lambda) - \mathbf{b}^T\lambda,\end{aligned} \tag{3.4}$$

where $f^*(\cdot)$ is the conjugate function of f [12].

When strong duality holds, this implies that the primal and dual problems have the same optimal value. In other words, we have

$$\sup_{\lambda} \inf_{\mathbf{x}} L(\mathbf{x},\lambda) = \inf_{\mathbf{x}} \sup_{\lambda} L(\mathbf{x},\lambda). \tag{3.5}$$

Consequently, the following inequalities hold:

$$L(\mathbf{x}^*,\lambda) \leq L(\mathbf{x}^*,\lambda^*) \leq L(\mathbf{x},\lambda^*). \tag{3.6}$$

The optimal point $(\mathbf{x}^*,\lambda^*)$ is thus a saddle point.

Assume that the strong duality holds, $\mathbf{x}^*$ is the primal optimal, and λ^* is the dual optimal, then

$$\begin{aligned}f(\mathbf{x}^*) &= g(\lambda^*) \\ &= \min_{\mathbf{x}} L(\mathbf{x},\lambda^*) \\ &\leq L(\mathbf{x}^*,\lambda^*) \\ &= f(\mathbf{x}^*),\end{aligned} \tag{3.7}$$

from which we can conclude that

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} L(\mathbf{x},\lambda^*) \tag{3.8}$$

Therefore, finding the saddle point is equivalent to solving the primal problem.

Let $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N]$, where $\mathbf{x}_i \in \mathbb{R}^{M\times 1}$, then the objective function f is separable if

$$f(\mathbf{x}) = \sum_{i=1}^{N} f_i(\mathbf{x}_i). \tag{3.9}$$

The essence of the distributed optimization problem is the collaboration between the nodes in a network. In this network, each node can only communicate with adjacent nodes. Based on the information obtained from other nodes, each node can update its local variable $\mathbf{x}_i$. The separable objective function is therefore jointly optimized by those nodes in an iterative manner. The network can be defined using a graph $G = (\mathcal{V}, \mathcal{E})$ where the set $\mathcal{V}$ denotes the set of vertices and $\mathcal{E}$ denotes the set of edges (connections) between the nodes in the graph.

The consensus problem which is solved and presented in later chapters can be formulated as

$$\begin{aligned}&\min_{\mathbf{x}} \sum_{i=1}^{N} f_i(\mathbf{x}_i) \\ &\text{s.t. } \mathbf{x}_1 = \mathbf{x}_2 = \cdots = \mathbf{x}_N,\end{aligned} \tag{3.10}$$

3.2 Optimization Algorithms

3.2.1 Dual Ascent and Method of Multipliers

The dual ascent (DA) method aims at finding the saddle point of the Lagrangian and the updating scheme at iteration k is:

$$\mathbf{x}^k = \arg\min_{\mathbf{x}} L(\mathbf{x}, \lambda^{k-1}) \tag{3.11}$$

$$\lambda^k = \lambda^{k-1} + \rho^k (A\mathbf{x}^k - \mathbf{b}), \tag{3.12}$$

where ρ^k is the step size used in the gradient ascent step in (3.12) in the k-th iteration [12].

If the objective function f is separable, then the corresponding Lagrangian is also separable:

$$\begin{aligned} L(\mathbf{x}, \lambda) &= \sum_{i=1}^{N} L_i(\mathbf{x}_i, \lambda) \\ &= \sum_{i=1}^{N} (f_i(\mathbf{x}_i) + \lambda^T A_i \mathbf{x}_i - \frac{1}{N} \lambda^T \mathbf{b}), \end{aligned} \tag{3.13}$$

where A_i is a part of A so that $A\mathbf{x} = \sum_{i=1}^{N} A_i \mathbf{x}_i$. The dual ascent method for this distributed problem is then

$$\mathbf{x}_i^k = \arg\min_{\mathbf{x}_i} L_i(\mathbf{x}_i, \lambda^{k-1}) \tag{3.14}$$

$$\lambda^k = \lambda^{k-1} + \rho^k (A\mathbf{x}^k - \mathbf{b}). \tag{3.15}$$

The update of λ in fact uses the gradient ascent technique, since given $\mathbf{x}^k$ we have

$$g(\lambda) = L(\mathbf{x}^k, \lambda), \tag{3.16}$$

which leads to

$$\nabla g(\lambda) = A\mathbf{x}^k - \mathbf{b}. \tag{3.17}$$

Since in (3.14) the optimization steps can be carried out in parallel for each $\mathbf{x}_i$, $i = 1, 2, \cdots, N$, this dual ascent method is also known as dual decomposition [1].

The dual ascent method does not however always converge. It requires assumptions like strict convexity and finiteness of f. After this, the method of multipliers (MM) was developed to bring robustness and to improve the convergence. The difference between DA and MM is the Lagrangian of the primal problem. In MM, instead of the standard Lagrangian, an augmented Lagrangian is used. The augmented Lagrangian for (3.2) is

$$L_\rho(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda^T (A\mathbf{x} - \mathbf{b}) + \frac{\rho}{2} \|A\mathbf{x} - \mathbf{b}\|_2^2 \tag{3.18}$$

The updating scheme of MM at iteration k is

$$\mathbf{x}^k = \arg\min_{\mathbf{x}} L_\rho(\mathbf{x}, \lambda^{k-1}) \tag{3.19}$$

$$\lambda^k = \lambda^{k-1} + \rho(A\mathbf{x}^k - \mathbf{b}). \tag{3.20}$$

Unlike DA, the augmented Lagrangian is not separable due to the l_2-norm term. Consequently, even if $f(\mathbf{x})$ is separable, $\mathbf{x}$ still has to be optimized jointly.

3.2.2 Standard Alternating Direction Method of Multipliers

As it was the intention to have both the decomposability of DA and the convergence properties of MM, the alternating direction method of multipliers (ADMM) was developed. The problem it solves is described as follows [1]:

$$\begin{aligned} &\min_{\mathbf{x},\mathbf{z}} f(\mathbf{x}) + g(\mathbf{z}) \\ &\text{s.t. } A\mathbf{x} + B\mathbf{z} = \mathbf{c}. \end{aligned} \tag{3.21}$$

The augmented Lagrangian of (3.21) is

$$L_\rho(\mathbf{x},\mathbf{z},\lambda) = f(\mathbf{x}) + g(\mathbf{z}) + \lambda^T(A\mathbf{x} + B\mathbf{z} - \mathbf{c}) + \frac{\rho}{2}\|A\mathbf{x} + B\mathbf{z} - \mathbf{c}\|_2^2. \tag{3.22}$$

The update scheme of ADMM at k-th iteration is

$$\mathbf{x}^k = \arg\min_{\mathbf{x}} L_\rho(\mathbf{x}, \mathbf{z}^{k-1}, \lambda^{k-1}) \tag{3.23}$$

$$\mathbf{z}^k = \arg\min_{\mathbf{z}} L_\rho(\mathbf{x}^k, \mathbf{z}, \lambda^{k-1}) \tag{3.24}$$

$$\lambda^k = \lambda^{k-1} + \rho(A\mathbf{x}^k + B\mathbf{z}^k - \mathbf{c}). \tag{3.25}$$

To minimize the augmented Lagrangian, ADMM uses a Gauss-Seidel step which implies that the updates have to be done in a sequential order. It can be viewed as a version of MM where a single Gauss-Seidel pass over $\mathbf{x}$ and $\mathbf{z}$ is used instead of the usual joint minimization [1].

For a consensus problem, one formulation for ADMM is

$$\begin{aligned} &\min_{\mathbf{x}} \sum_{i=1}^{N} f_i(\mathbf{x}_i) \\ &\text{s.t. } \mathbf{x}_1 = \mathbf{x}_2 = \cdots = \mathbf{x}_N = \mathbf{z}. \end{aligned} \tag{3.26}$$

The updating scheme is therefore

$$\mathbf{x}_i^k = \arg\min_{\mathbf{x}_i} \left(f_i(\mathbf{x}_i) + \lambda_i^{k-1^T}(\mathbf{x}_i - \mathbf{z}^{k-1}) + \frac{\rho}{2}\|\mathbf{x}_i - \mathbf{z}^{k-1}\|_2^2 \right), \ \forall i \in \mathcal{V} \tag{3.27}$$

$$\mathbf{z}^k = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i^k + \frac{1}{\rho}\lambda_i^{k-1}) \tag{3.28}$$

$$\lambda_i^k = \lambda_i^{k-1} + \rho(\mathbf{x}_i^k - \mathbf{z}^k), \ \forall i \in \mathcal{V}, \tag{3.29}$$

where λ_i is the Lagrangian multiplier corresponding to the equality constraint $\mathbf{x}_i = \mathbf{z}$, and $\lambda = [\lambda_1, \lambda_2, \cdots, \lambda_N]^T$. It is worth noting that $\mathbf{x}_i$ and λ_i can be computed locally at node i, while computing $\mathbf{z}$ requires the information from all other nodes. Afterwards, the new update of $\mathbf{z}$ needs to be broadcast back to each node. This means that in the network there is a fusion center which has to be connected to all of the nodes within the same network. As a result, the updating scheme in (3.27) requires a special network topology. All the nodes have to be connected to the fusion center, and they do not need to communicate with each other. This special topology is not always realistic in practical applications. To deal with this disadvantage, a decentralized solution has been conveyed. The problem in (3.26) can be reformulated as:

$$\begin{aligned} &\min_{\mathbf{x}} \sum_{i=1}^{N} f_i(\mathbf{x}_i) \\ &\text{s.t. } \mathbf{x}_i = \mathbf{x}_j = \mathbf{z}_{ij}, \ (i,j) \in \mathcal{E}. \end{aligned} \tag{3.30}$$

The decentralized updating scheme is then:

$$\mathbf{x}_i^k = \arg\min_{\mathbf{x}_i} \left(f_i(\mathbf{x}_i) + \sum_{j \in \mathcal{N}(i)} ({\lambda_{ij}^{k-1}}^T (\mathbf{x}_i - \mathbf{z}_{ij}^{k-1}) + \frac{\rho}{2} \|\mathbf{x}_i - \mathbf{z}_{ij}^{k-1}\|_2^2) \right), \forall i \in \mathcal{V} \tag{3.31}$$

$$\mathbf{z}_{ij}^k = \frac{1}{2} \left(\mathbf{x}_i^k + \mathbf{x}_j^k + \frac{1}{\rho} (\lambda_{ij}^{k-1} + \lambda_{ij}^{k-1}) \right), \forall (i,j) \in \mathcal{E} \tag{3.32}$$

$$\lambda_{ij}^k = \lambda_{ij}^{k-1} + \rho(\mathbf{x}_i^k - \mathbf{z}_{ij}^k), \forall (i,j) \in \mathcal{E} \tag{3.33}$$

In (3.30), a local variable $\mathbf{z}_{ij}$ shared by node i and j is used instead of a global variable $\mathbf{z}$. Consequently, each $\mathbf{z}_{ij}$ can be computed by either node i or j. The special topology with the fusion center is thus unnecessary.

3.2.3 Variants of Alternating Direction Method of Multipliers

Many variants of ADMM have been developed. Here we introduce two variants which are related to our study.

Jacobi version of Alternating Direction Method of Multipliers

A variant of the ADMM is to use a Jacobi iteration to update the primal variables instead of using a Gauss-Seidel iteration. The updating scheme for a separable objective function is:

$$\mathbf{x}_i^k = \arg\min_{\mathbf{x}_i} L_a(\mathbf{x}_1^{k-1}, \mathbf{x}_2^{k-1}, \cdots, \mathbf{x}_{i-1}^{k-1}, \mathbf{x}_i, \mathbf{x}_{i-1}^{k+1}, \cdots, \mathbf{x}_N^{k-1}, \lambda^{k-1}), \forall i \in \mathcal{V} \tag{3.34}$$

$$\lambda^k = \lambda^{k-1} + \rho(A\mathbf{x}^k - \mathbf{b}). \tag{3.35}$$

Due to the Jacobi iteration, the step that updates the common variable $\mathbf{z}$ is not needed. However, its convergence is not guaranteed. In [13, 14] it was shown that these Jacobi updates may not converge.

In [5], the consensus problem as shown in (3.10) is reformulated for Jacobi ADMM.

Definition 3.1. Given a graph with E as the number of edges and N as the number of nodes, the incidence matrix $C \in \mathbb{R}^{E \times N}$ is defined as

$$[C]_{ek} = \begin{cases} +1, & k \text{ is the node connected to edge e with a lower index} \\ -1, & k \text{ is the node connected to edge e with a higher index} \\ 0, & k \text{ is not connected to edge e} \end{cases} \tag{3.36}$$

The problem can therefore be rewritten as

$$\begin{aligned} &\min_{\mathbf{x}_i} \sum_{i=1}^N f_i(\mathbf{x}_i) \\ &\text{s.t. } \mathcal{C}\mathbf{x} = \mathbf{0}, \end{aligned} \tag{3.37}$$

where

$$\mathcal{C} \triangleq C \otimes I_M, \quad \mathbf{x} \triangleq \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_N \end{bmatrix}. \tag{3.38}$$

Asynchronous Alternating Direction Method of Multipliers

Unlike in the standard ADMM where in each iteration all the nodes update their variables synchronously, an asynchronous updating scheme only has one or a few selected nodes activated within one iteration. Recently, an asynchronous version of the ADMM was proposed in [4]. In this work, an edge-based selection strategy was proposed, where at each iteration one edge was chosen. The two nodes connected by this edge are required to update, and the rest of the nodes remain unchanged within the iteration.

The algorithm reformulates the problem in (3.10) as

$$\begin{aligned} &\min_{\mathbf{x}} \sum_{i=1}^{N} f_i(\mathbf{x}_i) \\ &\text{s.t. } C_{ei}\mathbf{x}_i = \mathbf{z}_{ei},\ \forall e \in \mathcal{E}, i \in \mathcal{N}(e), \end{aligned} \tag{3.39}$$

where $\mathbf{z}_{ei}$ is the variable associated with the node i in the edge e. For an edge $e=(p,q)$, the constraint can be written as $\mathbf{x}_p = \mathbf{z}_{ep}$ and $-\mathbf{x}_q = \mathbf{z}_{eq}$, with $\mathbf{z}_{ep} + \mathbf{z}_{eq} = 0$.

The update scheme at the k-th iteration where edge $e=(p,q)$ is activated is described as follows:

$$\mathbf{x}_i^k = \begin{cases} \arg\min_{\mathbf{x}_i}(f_i(\mathbf{x}_i) - (\lambda_{ei}^{k-1})^T A_{ei}\mathbf{x}_i + \frac{\rho}{2}\|A_{ei}\mathbf{x}_i - \mathbf{z}_{ei}^{k-1}\|_2^2), & i = p,q \\ \mathbf{x}_i^{k-1}, & i \neq p,q \end{cases} \tag{3.40}$$

$$\mathbf{z}_{ei}^k = \begin{cases} -\frac{1}{\rho}(\lambda_{ei}^{k-1} - \nu) + A_{ei}\mathbf{x}_i^k, & i = p,q \\ \mathbf{z}_{ei}^{k-1}, & i \neq p,q \end{cases} \tag{3.41}$$

$$\lambda_{ei}^k = \begin{cases} -\nu, & i = p,q \\ \lambda_{ei}^{k-1}, & i \neq p,q \end{cases}, \tag{3.42}$$

where

$$\nu = -\frac{1}{2}(\lambda_{ep}^{k-1} + \lambda_{eq}^{k-1}) + \frac{\rho}{2}(A_{ep}\mathbf{x}_p^k + A_{eq}\mathbf{x}_q^k). \tag{3.43}$$

Although [4] claims that the edge-based asynchronous ADMM method as shown in (3.40) to (3.42) has a convergence rate of O(1/k), we did not observe such a convergence rate. In our experiments, it does not converge to the optimal point. A modified version of this algorithm for the consensus problem is introduced in [11] and converges to the optimal point in the experiments. It is described here as follows:

$$\mathbf{x}_i^k = \begin{cases} \arg\min_{\mathbf{x}_i}(f_i(\mathbf{x}_i) + \sum_{j \in \mathcal{N}(i)} (\lambda_{ij}^{k-1^T}(\mathbf{x}_i - \mathbf{z}_{ij}^{k-1}) + \frac{\rho}{2}\|\mathbf{x}_i - \mathbf{z}_{ij}^{k-1}\|_2^2)), & i = p,q \\ \mathbf{x}_i^{k-1}, & i \neq p,q \end{cases} \tag{3.44}$$

$$\mathbf{z}_{ij}^k = \begin{cases} \frac{1}{2}(\mathbf{x}_i^k + \mathbf{x}_j^k + \frac{1}{\rho}(\lambda_{ij}^{k-1} + \lambda_{ij}^{k-1})), & (i,j) = (p,q) \\ \mathbf{z}_{ij}^{k-1}, & (i,j) \neq (p,q) \end{cases} \tag{3.45}$$

$$\lambda_{ij}^k = \begin{cases} \lambda_{ij}^{k-1} + \rho(\mathbf{x}_i^k - \mathbf{z}_{ij}^k), & (i,j) = (p,q) \\ \lambda_{ei}^{k-1}, & (i,j) \neq (p,q) \end{cases} \tag{3.46}$$

3.2.4 Primal Dual Method of Multipliers

The PDMM is the algorithm that has been recently introduced [9][10][11]. As explained below, it is a node-based algorithm which means that an asynchronous updating scheme can be easily derived. Although there is a lack of

deep research and understanding on this algorithm, it is proved in [11] to have a $O(1/K)$ linear convergence rate for both synchronous and asynchronous updating schemes.

The PDMM solves the problem in the form of

$$\begin{aligned} &\min_{\mathbf{x}} \sum_{i\in\mathcal{V}} f_i(\mathbf{x_i}) \\ &\text{s.t. } A_{ij}\mathbf{x_i} + A_{ji}\mathbf{x_j} = \mathbf{c_{ij}},\ \forall (i,j)\in\mathcal{E}, \end{aligned} \tag{3.47}$$

where $A_{ij}\in\mathbb{R}^{P\times MN}$ and $\mathbf{c}_{ij}\in\mathbb{R}^{P\times 1}$. The arrow means that A_{ij} and A_{ji} may not be equal. If one takes the consensus problem as an example, the equality constraint on edge (i,j) is then $\mathbf{x}_i-\mathbf{x}_j=\mathbf{0}$, i.e. $A_{ij}=I$ and $A_{ji}=-I$.

For convenience, we denote δ as the Lagrangian multiplier, and the Lagrangian of this primal problem can be constructed as

$$L_p(\mathbf{x},\delta) = \sum_{(i,j)\in\mathcal{E}} \delta_{ij}^T(\mathbf{c_{ij}} - A_{ij}\mathbf{x}_i - A_{ji}\mathbf{x_j}) + \sum_{i\in\mathcal{V}} f_i(\mathbf{x_i}), \tag{3.48}$$

where δ_{ij} is the Lagrangian multiplier for the corresponding edge constraint in (3.47), and δ is the vector consists of all the δ_{ij}, $\forall(i,j)\in\mathcal{E}$.

By minimizing $L_p(\mathbf{x},\delta)$, the dual problem of (3.47) is derived as:

$$\max_{\delta} -\sum_{i\in\mathcal{V}} f_i^*\Big(\sum_{j\in\mathcal{N}(i)} A_{ij}^T\delta_{ij}\Big) + \sum_{(i,j)\in\mathcal{E}} \delta_{ij}^T\mathbf{c}_{ij}. \tag{3.49}$$

It is noted in (3.49) that for $(i,j)\in\mathcal{E}$, δ_{ij} is shared by two conjugate functions $f_i^*(\cdot)$ and $f_j^*(\cdot)$, which results in the entanglement between the conjugate functions in the graph. In order to decouple the entanglement, two auxiliary variables $\lambda_{i|j}$ and $\lambda_{j|i}$ are introduced, where $\lambda_{i|j}$ is a variable owned by node i that corresponds to node j. Therefore two node variables with equality constraint are used instead of an edge variable λ_{ij} shared by two nodes. As a result, (3.49) can be reformulated as:

$$\begin{aligned} &\max_{\delta,\lambda} -\sum_{i\in\mathcal{V}} f_i^*\Big(\sum_{j\in\mathcal{N}(i)} A_{ij}^T\lambda_{i|j}\Big) + \sum_{(i,j)\in\mathcal{E}} \delta_{ij}^T\mathbf{c}_{ij} \\ &\text{s.t. } \lambda_{i|j} = \lambda_{j|i} = \delta_{ij},\ \forall(i,j)\in\mathcal{E}, \end{aligned} \tag{3.50}$$

Let A_i^T be the matrix obtained by horizontally stacking A_{ij}^T, $j\in\mathcal{N}(i)$, λ_i be the vector obtained by vertically stacking $\lambda_{i|j}$, $j\in\mathcal{N}(i)$, and λ is the matrix obtained by horizontally concatenating λ_i, $i\in\mathcal{V}$. If node i is connected to nodes $1,2,3$, then we have

$$A_i^T = \begin{bmatrix} A_{i1}^T & A_{i2}^T & A_{i3} \end{bmatrix},\ \lambda_i = \begin{bmatrix} \lambda_{i|1} \\ \lambda_{i|2} \\ \lambda_{i|3} \end{bmatrix} \tag{3.51}$$

It can be easily shown that

$$\sum_{j\in\mathcal{N}(i)} A_{ij}^T\lambda_{i|j} = A_i^T\lambda_i. \tag{3.52}$$

The Lagrangian of the dual problem is therefore:

$$\begin{aligned} L_d(\delta,\lambda,\mathbf{z}) = &-\sum_{i\in\mathcal{V}} f_i^*(A_i^T\lambda_i) + \sum_{(i,j)\in\mathcal{E}} \delta_{ij}^T\mathbf{c}_{ij} \\ &+ \sum_{(i,j)\in\mathcal{E}} \mathbf{z}_{i|j}^T(\delta_{ij}-\lambda_{i|j}) + \sum_{(i,j)\in\mathcal{E}} \mathbf{z}_{j|i}^T(\delta_{ij}-\lambda_{j|i}). \end{aligned} \tag{3.53}$$

The newly introduced variable $\mathbf{z}_{i|j}$ can be replaced by observing that at a saddle point of L_d we have

$$0 = \partial_{\lambda_{i|j}} L_d(\delta^*, \lambda^*, \mathbf{z}^*) = \partial_{\lambda_{i|j}} f_i^*(\sum_{j\in\mathcal{N}(i)} A_{ij}^T \lambda_{i|j}^*) + \mathbf{z}_{i|j}^*. \tag{3.54}$$

On the other hand, Frenchel's inequality leads to

$$f_i(\mathbf{x}_i) + f_i^*(\sum_{j\in\mathcal{N}(i)} A_{ij}^T \lambda_{i|j}) \geq \sum_{j\in\mathcal{N}(i)} \lambda_{i|j}^T A_{ij} \mathbf{x}_i. \tag{3.55}$$

The equality holds at the saddle point, from which we can derive that

$$0 = \partial_{\lambda_{i|j}} f_i^*(\sum_{j\in\mathcal{N}(i)} A_{ij}^T \lambda_{i|j}^*) - A_{ij}\mathbf{x}_i^* = \partial_{\lambda_{i|j}} f_i^*(\sum_{j\in\mathcal{N}(i)} A_{ij}^T \lambda_{i|j}^*) + A_{ji}\mathbf{x}_j^* - \mathbf{c}_{ij}. \tag{3.56}$$

These result in $\mathbf{z}_{i|j} = A_{ji}\mathbf{x}_j^* - \mathbf{c}_{ij}$.

By restricting the Lagrangian multiplier $\mathbf{z}_{i|j}$ equal to $A_{ji}\mathbf{x}_j - \mathbf{c}_{ij}$, the augmented primal-dual Lagrangian function is then defined as

$$\begin{aligned} L_P(\{\mathbf{x}_i\}, \{\lambda_i\}) &= L_p(\mathbf{x}, \delta) + L_d(\delta, \lambda, \mathbf{x}) + h(\{\mathbf{x}_i\}, \{\lambda_i\}) \\ &= \sum_i [f_i(\mathbf{x}_i) - \sum_{j\in\mathcal{N}(i)} \lambda_{j|i}^T (A_{ij}\mathbf{x}_i - \mathbf{c}_{ij}) - f_i^*(A_i^T \lambda_i)] + h(\{\mathbf{x}_i\}, \{\lambda_i\}), \end{aligned} \tag{3.57}$$

where $\lambda_{j|i}$ is those elements of λ_j which are related to node i, and $h(\{\mathbf{x}_i\}, \{\lambda_i\})$ is defined as

$$h(\{\mathbf{x}_i\}, \{\lambda_i\}) = \sum_{(i,j)\in\mathcal{E}} (\frac{1}{2}\|A_{ij}\mathbf{x}_i + A_{ji}\mathbf{x}_j + \mathbf{c}_{ij}\|_{P_{p,ij}}^2 - \frac{1}{2}\|\lambda_{i|j} - \lambda_{j|i}\|_{P_{d,ij}}^2. \tag{3.58}$$

Here the $P_{p,ij}$ and $P_{d,ij}$ are positive-definite matrices for regularization for each edge $(i, j) \in \mathcal{E}$.

In (3.57), δ is cancelled out, and the variables $\mathbf{x}_i$ and λ_i corresponding to the same node i, $i \in \mathcal{V}$, are not directly related, i.e. optimization over $\mathbf{x}$ and λ is separable.

At iteration k, the synchronous update scheme of PDMM is therefore:

$$\mathbf{x}_i^k = \arg\min_{\mathbf{x}_i} L_P(\{\mathbf{x}_1^{k-1}, \mathbf{x}_2^{k-1}, \cdots, \mathbf{x}_i, \cdots, \mathbf{x}_N^{k-1}\}, \{\lambda_j^{k-1}\}),\ \forall i \in \mathcal{V},\ j \in \mathcal{N}(i) \tag{3.59}$$

$$\lambda_i^k = \arg\max_{\lambda_i} L_P(\{x_j^{k-1}\}, \{\lambda_1^{k-1}, \lambda_2^{k-1}, \cdots, \lambda_i, \cdots, \lambda_N^{k-1}\}),\ \forall i \in \mathcal{V},\ j \in \mathcal{N}(i), \tag{3.60}$$

where updating $\mathbf{x}$ and λ only requires the result from the previous iteration.

In contrast to the synchronous update scheme, the asynchronous update scheme of PDMM at iteration k is:

$$\mathbf{x}_i^k = \begin{cases} \arg\min_{\mathbf{x}_i} L_P(\{\mathbf{x}_i, x_j^{k-1}\}, \{\lambda_i^{k-1}\}), & \forall i \in \mathcal{V},\ i \text{ is activated} \\ \mathbf{x}_i^{k-1}, & \forall i \in \mathcal{V},\ i \text{ is not activated} \end{cases} \tag{3.61}$$

$$\lambda_i^k = \begin{cases} \arg\max_{\lambda_i} L_P(\{x_i^{k-1}\}, \{\lambda_i, \lambda_j^{k-1}\}), & \forall i \in \mathcal{V},\ i \text{ is activated} \\ \lambda_i^{k-1}, & \forall i \in \mathcal{V},\ i \text{ is not activated} \end{cases}. \tag{3.62}$$

Unlike the edge-based asynchronous update scheme where the edge is the basic unit to activate, the asynchronous update scheme of PDMM is node-based. This means that each node can update independently within the iteration and only needs the information computed in previous iterations from other nodes.

As shown in (3.57), in the augmented primal-dual Lagrangian, conjugate functions are involved. It could therefore be difficult to obtain the updates of λ's. Nevertheless, by properly choosing the matrices $P_{p,ij}$ and $P_{d,ij}$. If the matrix $P_{d,ij}$ for each neighbor $j \in \mathcal{N}(i)$ is chosen to be the inverse of $P_{p,ij}$, then there is

$$\lambda_{i|j}^k = \lambda_{j|i}^{k-1} + P_{p,ij}(\mathbf{c}_{ij} - A_{ji}\mathbf{x}_j^{k-1} - A_{ij}\mathbf{x}_i^K),\ \forall i \in \mathcal{V},\ j \in \mathcal{N}(i). \tag{3.63}$$

3.3 Background Concepts

In this section, we introduce the background knowledge that is used in our study.

3.3.1 The Karush-Kuhn-Tucker conditions

The Karush-Kuhn-Tucker(KKT) conditions are the necessary conditions for an optimal solution in nonlinear optimization problems. The nonlinear optimization problem can be formulated as

$$\begin{aligned} &\min_{\mathbf{x}} f_0(\mathbf{x}) \\ &\text{s.t. } f_i(\mathbf{x}) \le 0,\ i = 1,2,\cdots,m \\ &\quad h_j(\mathbf{x}) = 0,\ j = 1,2,\cdots,p. \end{aligned} \tag{3.64}$$

Its Lagrangian is

$$L(\mathbf{x},\lambda,\nu) = f_0(\mathbf{x}) + \sum_{i=1}^m \lambda_i^T f_i(\mathbf{x}) + \sum_{j=1}^p \nu_i^T h_j(\mathbf{x}). \tag{3.65}$$

Assume that f_i and h_j are differentiable, then the KKT conditions for this problem are

$$\begin{aligned} f_i(\mathbf{x}) \le 0, h_j(\mathbf{x}) = 0,\ \forall i,j, \\ \lambda \succeq 0, \\ \lambda_i^T f_i(\mathbf{x}) = 0,\ \forall i, \\ \nabla f_0(\mathbf{x}) + \sum_{i=1}^m \lambda_i^T \nabla f_i(\mathbf{x}) + \sum_{j=1}^p \nu_i^T \nabla h_j(\mathbf{x}) = 0. \end{aligned} \tag{3.66}$$

For convex problems, the KKT conditions are sufficient.

3.3.2 Strong convexity

Strong convexity is a common assumption that is used to prove linear convergence rate [7][8][12]. A function is strongly convex on S [12] if there exists an $m > 0$ so that

$$\nabla^2 f(\mathbf{x}) \succeq mI,\ \forall \mathbf{x} \in S. \tag{3.67}$$

One of the consequences of strong convexity is

$$f(\mathbf{y}) \ge f(\mathbf{x}) + \nabla f(\mathbf{x})^T(\mathbf{y}-\mathbf{x}) + \frac{m}{2}\|\mathbf{y}-\mathbf{x}\|^2,\ \mathbf{x},\mathbf{y} \in S. \tag{3.68}$$

The right-hand side of (3.68) is a convex function of $\mathbf{y}$ and $\tilde{\mathbf{y}} = \mathbf{x} - \frac{1}{m}\nabla f(\mathbf{x})$ minimizes it. Therefore we obtain

$$\begin{aligned} f(\mathbf{y}) &\geq f(\mathbf{x}) + \nabla f(\mathbf{x})^T(\tilde{\mathbf{y}} - \mathbf{x}) + \frac{m}{2}\|\tilde{\mathbf{y}} - \mathbf{x}\|^2 \\ &= f(\mathbf{x}) - \frac{1}{2m}\|\nabla f(\mathbf{x})\|^2. \end{aligned} \tag{3.69}$$

Since (3.69) holds for all $\mathbf{y} \in S$, it also holds for $\mathbf{x}^*$, which leads to

$$f(\mathbf{x}^*) \geq f(\mathbf{x}) - \frac{1}{2m}\|\nabla f(\mathbf{x})\|^2. \tag{3.70}$$

Besides the lower bound for $\nabla^2 f(x)$ in (3.67), there is also an upper bound, i.e. there exists a constant M such that

$$\nabla^2 f(x) \preceq MI, \ \forall \mathbf{x} \in S. \tag{3.71}$$

Similarly, we can derive

$$f(\mathbf{x}^*) \leq f(\mathbf{x}) - \frac{1}{2M}\|\nabla f(\mathbf{x})\|^2. \tag{3.72}$$

If we interchange $\mathbf{x}$ and $\mathbf{y}$ in (3.68), and then take the sum of the new inequality with (3.68), then we have

$$(\nabla f(\mathbf{x}) - \nabla f(\mathbf{y}))^T(\mathbf{x} - \mathbf{y}) \geq m\|\mathbf{x} - \mathbf{y}\|^2. \tag{3.73}$$

3.3.3 Inequality with eigenvalues and singular values

Let $D \in \mathbb{R}^{K\times K}$ be a symmetric positive-definite matrix and $\mathbf{u} \in \mathbb{R}^{K\times 1}$, then the following inequality holds:

$$\lambda_{min}(D)\|\mathbf{u}\|^2 \leq \|\mathbf{u}\|_D^2 \leq \lambda_{max}(D)\|\mathbf{u}\|^2 \tag{3.74}$$

This inequality can be easily shown via the eigenvalue decomposition. The matrix D can be decomposed as $D = Q\Lambda Q^T$, where Q is a $K \times K$ orthogonal matrix that contains the eigenvectors and Λ is a $K \times K$ diagonal matrix whose entries are the eigenvalues of D. Denote $\mathbf{q}_i$ as the q-th column of Q for $i = 1, 2, \cdots, K$ and λ_i as the i-th entry on the diagonal of Λ for $i = 1, 2, \cdots, K$, then we have

$$D = \begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 & \cdots & \mathbf{q}_K \end{bmatrix} \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_K \end{bmatrix} \begin{bmatrix} \mathbf{q}_1^T \\ \mathbf{q}_2^T \\ \vdots \\ \mathbf{q}_K^T \end{bmatrix} = \sum_{i=1}^{K} \lambda_i \mathbf{q}_i \mathbf{q}_i^T \tag{3.75}$$

Accordingly,

$$\begin{aligned} \|\mathbf{u}\|_D^2 &= \mathbf{u}^T D \mathbf{u} \\ &= \mathbf{u}^T (\sum_{i=1}^{K} \lambda_i \mathbf{q}_i \mathbf{q}_i^T) \mathbf{u} \\ &= \sum_{i=1}^{K} \lambda_i \mathbf{u}^T \mathbf{q}_i \mathbf{q}_i^T \mathbf{u} \end{aligned} \tag{3.76}$$

Since D is positive-definite, all its eigenvalues are positive. As a result,

$$\begin{aligned}\|\mathbf{u}\|_D^2 &= \sum_{i=1}^{K} \lambda_i \mathbf{u}^T \mathbf{q}_i \mathbf{q}_i^T \mathbf{u} \\ &\leq \lambda_{max}(D) \sum_{i=1}^{K} \mathbf{u}^T \mathbf{q}_i \mathbf{q}_i^T \mathbf{u} \\ &= \lambda_{max}(D) \mathbf{u}^T (\sum_{i=1}^{K} \mathbf{q}_i \mathbf{q}_i^T) \mathbf{u} \\ &= \lambda_{max}(D) \mathbf{u}^T Q Q^T \mathbf{u} \\ &= \lambda_{max}(D) \|\mathbf{u}\|^2\end{aligned} \tag{3.77}$$

In a similar way, we can obtain the other half of (3.74).

Let E be a $KL \times J$ matrix constructed as

$$E = \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_K, \end{bmatrix} \tag{3.78}$$

where E_i, $i = 1, 2, \cdots, K$ has the dimension of $L \times J$. Furthermore, let $\mathbf{w}$ be a $KL \times 1$ vector which is

$$\mathbf{w} = \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \vdots \\ \mathbf{w}_K, \end{bmatrix} \tag{3.79}$$

where each $\mathbf{w}_i$, $i = 1, 2, \cdots, K$ is a $L \times 1$ vector. Similarly, we have

$$\sigma_{min}^2(E) \|\mathbf{w}\|^2 \leq \|E^T \mathbf{w}\|^2 \leq \sigma_{max}^2(E) \|\mathbf{w}\|^2. \tag{3.80}$$

If $\mathbf{w}_i$ lies in the column space of E_i, for all $i = 1, 2, \cdots, K$, then the following inequality holds:

$$\|E^T \mathbf{w}\|^2 \geq \tilde{\sigma}_{min}^2(E) \|\mathbf{w}\|^2 \tag{3.81}$$

This inequality can be proven using the singular value decomposition of a similar method used for proving (3.74). Let the singular value decomposition of E be

$$E = U \Sigma V^T, \tag{3.82}$$

then we have

$$\begin{aligned}\|E^T \mathbf{w}\|^2 &= \mathbf{w}^T E E^T \mathbf{w} \\ &= \mathbf{w}^T U \Sigma V^T V \Sigma^T U^T \mathbf{w} \\ &= \mathbf{w}^T U \Sigma \Sigma^T U^T \mathbf{w}.\end{aligned} \tag{3.83}$$

Denoting $\mathbf{u}_i$, $i = 1, 2, \cdots, KL$ as the i-th column of U and σ_i^2, $i = 1, 2, \cdots, KL$ as the i-th entry on the diagonal of $\Sigma \Sigma^T$ leads to

$$\|E^T \mathbf{w}\|^2 = \sum_{i=1}^{KL} \sigma_i^2 \mathbf{w}^T \mathbf{u}_i \mathbf{u}_i^T \mathbf{w} \tag{3.84}$$

Denote the rank of E as R. If $R = KL$, then all σ_i^2, $i = 1, 2, \cdots, KL$ is non-zero. As a consequence,

$$\begin{aligned} \|E^T \mathbf{w}\|^2 &\geq \tilde{\sigma}_{min}(E) \sum_{i=1}^{KL} \mathbf{w}^T \mathbf{u}_i \mathbf{u}_i^T \mathbf{w} \\ &= \tilde{\sigma}_{min}(E) \|\mathbf{w}\|^2. \end{aligned} \tag{3.85}$$

In the case that $R < KL$, then the following equation holds for $i > R$:

$$E^T \mathbf{u}_i = 0. \tag{3.86}$$

By decomposing $\mathbf{u}_i$ as

$$\mathbf{u}_i = \begin{bmatrix} \mathbf{u}_{i1} \\ \mathbf{u}_{i2} \\ \vdots \\ \mathbf{u}_{iK}, \end{bmatrix} \tag{3.87}$$

where each $\mathbf{u}_{ij}$, $j = 1, 2, \cdots, K$ is a $L \times 1$ vector, (3.86) is equivalent to

$$\sum_{j=1}^{K} E_j^T \mathbf{u}_{ij} = 0. \tag{3.88}$$

Splitting the right-hand side of (3.84) yields

$$\begin{aligned} \|E^T \mathbf{u}\|^2 &= \sum_{i=1}^{R} \sigma_i^2 \mathbf{w}^T \mathbf{u}_i \mathbf{u}_i^T \mathbf{w} + \sum_{i=R+1}^{KL} \sigma_i^2 \mathbf{w}^T \mathbf{u}_i \mathbf{u}_i^T \mathbf{w} \\ &\geq \tilde{\sigma}_{min}(E) \|\mathbf{w}\|^2. \end{aligned} \tag{3.89}$$

Chapter 4

Case Study: Distributed Averaging Problem

Firstly, a simple averaging problem is conveyed for illustration purposes. Given a network where each node owns a number (the result of a measurement, for example), we would like to compute the mean of all these numbers by fully distributed method.

4.1 Problem Formulation

For a given connected graph consisting of N nodes, where each node owns a_i, $i = 1, 2, \cdots, N$, the averaging problem can be formulated as:

$$\min_x \sum_{i=1}^{N} \frac{1}{2}(x - a_i)^2. \tag{4.1}$$

This formulation can easily be verified by the fact that at the optimal point, the derivative of the objective function is equal to zero, thus

$$0 = \partial_x \left(\sum_{i=1}^{N} \frac{1}{2}(x - a_i)^2 \right) = Nx - \sum_{i=1}^{N} a_i. \tag{4.2}$$

Rewriting the problem with local variables leads to

$$\begin{aligned} &\min_{x_i} \sum_{i=1}^{N} f_i(x_i) = \sum_{i=1}^{N} \frac{1}{2}(x_i - a_i)^2, \\ &\text{s.t. } x_1 = x_2 = \cdots = x_N, \end{aligned} \tag{4.3}$$

where x_i is the variable owned by the ith node.

During the transmission in the nodes, there may be some package loss, i.e. transmission failure. The package loss rate r is defined as the ratio between the number of the transmission failure and the number of total transmissions. In this study, we applied those algorithms to scenarios with different package loss rates and compared their performances.

4.2 Algorithms

As described in Chapter 2, we implemented six algorithms in total. The updating schemes are derived as follows.

4.2.1 Decentralized ADMM

By substituting (4.3) into (3.31) to (3.33), we obtain the update scheme as follow:

$$x_i^k = \frac{1}{\rho|\mathcal{N}(i)|+1}(a_i + \sum_{j\in\mathcal{N}(i)}(\rho z_{ij}^{k-1} - \lambda_{ij}^{k-1})),\ \forall i \in \mathcal{V} \tag{4.4}$$

$$z_{ij}^k = \frac{1}{2}(x_i^k + x_j^k + \frac{1}{\rho}(\lambda_{ij}^{k-1} + \lambda_{ji}^{k-1})),\ \forall (i,j) \in \mathcal{E} \tag{4.5}$$

$$\lambda_{ij}^k = \lambda_{ij}^{k-1} + \rho(x_i^k - z_{ij}^k),\ \forall (i,j) \in \mathcal{E}. \tag{4.6}$$

4.2.2 Synchronous Jacobi ADMM

According to (3.34), the update scheme is

$$x_i^k = a_i + \sum_{\substack{j\in\mathcal{N}(i)\\ j>i}} \lambda_{e_{ij}}^{k-1} - \sum_{\substack{j\in\mathcal{N}(i)\\ j<i}} \lambda_{e_{ij}}^{k-1} + \sum_{j\in\mathcal{N}(i)} \rho x_j^{k-1},\ \forall i \in \mathcal{V} \tag{4.7}$$

$$\lambda_{e_{ij}}^k = \lambda_{e_{ij}}^{k-1} + \rho(x_i - x_j),\ \forall (i,j) \in \mathcal{E}, i < j, \tag{4.8}$$

where $\lambda_{e_{ij}}$ denotes the dual variable corresponding to the edge between nodes i and j. It is worth noting that since $\lambda_{e_{ij}}^k$ is a variable shared by two nodes, only one of the two nodes needs to compute update of $\lambda_{e_{ij}}^k$.

4.2.3 Asynchronous Jacobi ADMM

Similar to the synchronous updating scheme, the asynchronous updating scheme for Jacobi ADMM is

$$x_i^k = \begin{cases} a_i + \sum_{\substack{j\in\mathcal{N}(i)\\ j>i}} \lambda_{e_{ij}}^{k-1} - \sum_{\substack{j\in\mathcal{N}(i)\\ j<i}} \lambda_{e_{ij}}^{k-1} + \sum_{j\in\mathcal{N}(i)} \rho x_j^{k-1}, & i = p,q, \\ x_i^{k-1}, & i \neq p,q, \end{cases} \tag{4.9}$$

$$\lambda_{ij}^k = \begin{cases} \lambda_{e_{ij}}^{k-1} + \rho(x_i - x_j), & (i,j) = (p,q),\quad i < j, \\ \lambda_{ei}^{k-1}, & (i,j) \neq (p,q). \end{cases} \tag{4.10}$$

4.2.4 Asynchronous ADMM

If we substitute (4.3) into (3.44) to (3.46), then the updating scheme for averaging problem is:

$$x_i^k = \begin{cases} \frac{1}{\rho|\mathcal{N}(i)|+1}(a_i + \sum_{j\in\mathcal{N}(i)}(\rho z_{ij}^{k-1} - \lambda_{ij}^{k-1})), & i = p,q \\ x_i^{k-1}, & i \neq p,q \end{cases} \tag{4.11}$$

$$z_{ij}^k = \begin{cases} \frac{1}{2}(x_i^k + x_j^k + \frac{1}{\rho}(\lambda_{ij}^{k-1} + \lambda_{ij}^{k-1})), & (i,j) = (p,q) \\ z_{ij}^{k-1}, & (i,j) \neq (p,q) \end{cases} \tag{4.12}$$

$$\lambda_{ij}^k = \begin{cases} \lambda_{ij}^{k-1} + \rho(x_i^k - z_{ij}^k), & (i,j) = (p,q) \\ \lambda_{ei}^{k-1}, & (i,j) \neq (p,q) \end{cases} \tag{4.13}$$

4.2.5 Synchronous PDMM

By choosing $P_{p,ij} = \rho I$ and $P_{d,ij} = P_{p,ij}^{-1}$, the updating scheme of the synchronous PDMM is

$$x_i^k = a_i + \sum_{j \in \mathcal{N}(i)} A_{ij} \lambda_{j|i}^{k-1} + \rho \sum_{j \in \mathcal{N}(i)} x_j^{k-1}, \ \forall i \in \mathcal{V} \tag{4.14}$$

$$\lambda_{i|j}^k = \lambda_{j|i}^{k-1} - \rho(A_{ij} x_i^k + A_{ji} x_j^{k-1}), \ \forall (i,j) \in \mathcal{E}, \tag{4.15}$$

where

$$A_{ij} = \begin{cases} 1, & i < j \\ -1, & j < i \end{cases}. \tag{4.16}$$

4.2.6 Asynchronous PDMM

Using the same setting as for the synchronous PDMM, we obtain

$$x_i^k = \begin{cases} a_i + \sum_{j \in \mathcal{N}(i)} A_{ij} \lambda_{j|i}^{k-1} + \rho \sum_{j \in \mathcal{N}(i)} x_j^{k-1}, & i \text{ is activated} \\ x_i^{k-1}, & i \text{ is not activated} \end{cases} \tag{4.17}$$

$$\lambda_{i|j}^k = \begin{cases} \lambda_{j|i}^{k-1} - \rho(A_{ij} x_i^k + A_{ji} x_j^{k-1}), & i \text{ is activated,} \\ \lambda_{i|j}^{k-1}, & i \text{ is not activated.} \end{cases} \tag{4.18}$$

4.3 Experimental Results

For each group of algorithms, we conducted six experiments for comparison. There were two sorts of graphs being used. Both of these graphs contains 100 nodes. One graph is a 10×10 grid, where each node has 2 to 4 neighbors. While the others are randomly generated, where the possibility that each two nodes are connected is 0.5. Besides two sorts of graphs, we also set three package loss rates, namely 0, 0.2, 0.4. When a package loss occurs, the corresponding node will use the variable received previously. The local variable a_i, $i = 1, 2, \cdots, N$ is randomly generated with a uniform distribution between 0 and 100. The error is defined as

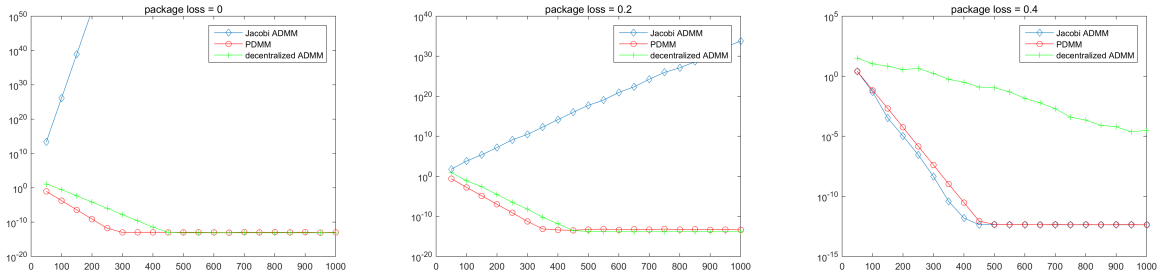
$$\varepsilon = \sum_{i \in \mathcal{V}} (x_i - a_i)^2 \tag{4.19}$$

4.3.1 Synchronous algorithms

The results for the grid are shown in Figures 4.1(a),4.1(b) and 4.1(c). In addition, the results for the random graph are illustrated in Figures 4.2(a), 4.2(b) and 4.2(c). In these experiments, the step size is 1 for all algorithms.

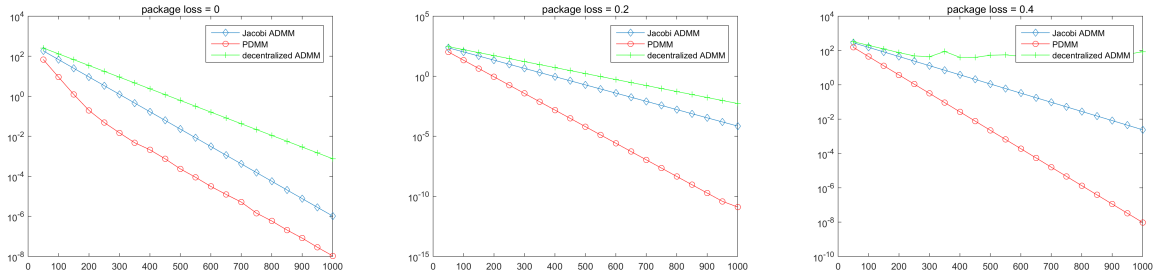
4.3.2 Asynchronous algorithms

Figures 4.1(a),4.1(b) and 4.1(c) show the results for the grid. Besides, Figures 4.2(a), 4.2(b) and 4.2(c) illustrate the results for the random graph. In these experiments, the step size is 1 for all algorithms.



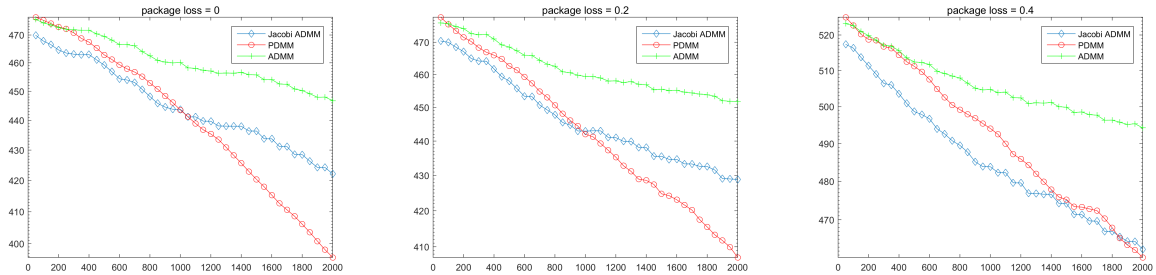
(a) (b) (c)

Figure 4.1: *Performance of synchronous algorithms on the* 10×10 *grid*



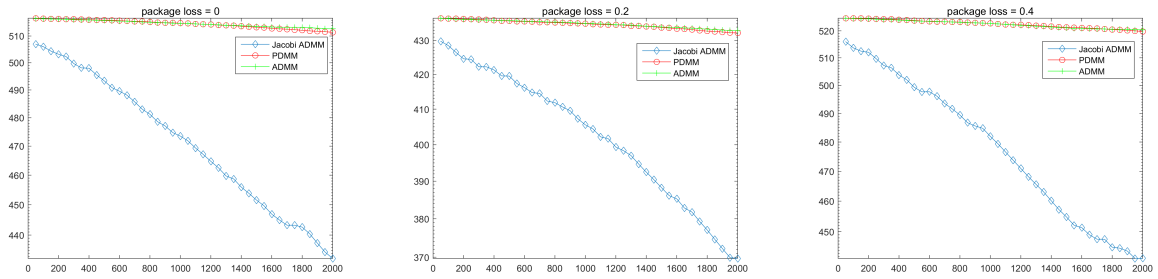
(a) (b) (c)

Figure 4.2: *Performance of synchronous algorithms on a random graph of* 100 *nodes*



(a) (b) (c)

Figure 4.3: *Performance of asynchronous algorithms on the* 10×10 *grid*



(a) (b) (c)

Figure 4.4: *Performance of asynchronous algorithms on a random graph of* 100 *nodes*

4.4 Discussion

The results from synchronous algorithms indicate that the PDMM always outperforms the decentralized ADMM in the experiments, especially when the package loss rate is high. It is clear from the figures that the Jacobi ADMM does not have guarantee for convergence. Its convergence behavior can be strongly affected by the topology of the network and the package loss rate.

According to the results from asynchronous algorithms, the PDMM outperforms the Jacobi ADMM and the ADMM on the grid. While the Jacobi ADMM shows best performance among those three algorithms on random graphs. Although in the experiments we did not observe divergence behavior of the Jacobi ADMM, we still do not have the guarantee that it will always converge.

Although the results illustrated in the thesis only use $\rho = 1$, informal experiments indicate that different values of ρ do not lead to different conclusion.

In summary, these results verify previous conclusions that the Jacobi ADMM is not assured to converge, whilethe ADMM has a linear convergence rate for distributed consensus problems. In previous work, the PDMM has been proved to converge with a rate of $O(1/k)$ when transmission failure is absent. Nevertheless, in these experiments, the PDMM does not only show a linear convergence rate, but also outperforms the ADMM. Particularly, the PDMM is more robust than the ADMM against package loss.

Chapter 5

Case Study: Distributed Dictionary Learning Problem

Dictionary Learning is a widely used method in the machine learning field. A dictionary is a collection of basis functions by which a datum can be represented. With given dictionary elements known as atoms, the learning process involves representing the input feature vectors as sparse linear combinations of the atoms. It can be applied in many areas such as image denoising, face recognition and classification.

5.1 Introduction to Distributed Dictionary Learning

A general dictionary learning solving procedure consists of two steps:

1. The inference step
   - in which the sparse coding technique is used to find the linear combination of the dictionary elements over a connected network of agents. The name "sparse" indicates that the coefficients of the linear combination should be sparse. In other words, only a few of the coefficients are non-zero, so that the representation of the datum only uses a few dictionary elements.

2. The dictionary updating step
   - for the updating of the dictionary elements to improve the future reconstruction.

When big data applications are involved, dictionary learning tasks need to be operated in a distributed manner. One method is to let each agent in a connected network maintain a part of the dictionary. It is assumed that the dictionary is decomposable so that each agent owns an independent part of the dictionary. Furthermore, the regularization functions are also assumed to be decomposable.

5.2 Problem Formulation

In this study, we only focus on the inference step where the optimization techniques discussed in previous chapters are applied. The centralized problem can be constructed as an optimization problem as follows:

$$\mathbf{y}^{\circ} \triangleq \arg\min_{\mathbf{y}} [d(\mathbf{t}-W\mathbf{y}) + h_y(\mathbf{y})], \tag{5.1}$$

where $\mathbf{t} \in \mathbb{R}^{M\times 1}$ is an input data realization, $W \in \mathbb{R}^{M\times N}$ is a dictionary matrix. The distance function $d(\cdot)$ is a measure to guarantee the accuracy of the representation, i.e. $\mathbf{t} \approx W\mathbf{y}$. Furthermore, the penalty function $h_y(\cdot)$

In this study, we use the formulation proposed in [5]. It is assumed that the dictionary is distributed separately to L agents in the network. In this case, the dictionary matrix W is divided into block columns and the vector x is partitioned into row entries as follows:

$$W = \begin{bmatrix} W_1 & \cdots & W_L \end{bmatrix}, \ \mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_L \end{bmatrix}. \tag{5.2}$$

In order to decompose the problem, the following terms are assumed to be decomposable : Let $h_{y_l}(\cdot)$ and $h_{W_l}(\cdot)$ be the regularization functions for agent l, then

$$h_y(\mathbf{y}) = \sum_{l=1}^{L} h_{y_l}(\mathbf{y}_l), \ h_W(W) = \sum_{l=1}^{L} h_{W_l}(W_l). \tag{5.3}$$

With the assumptions in (5.3), the objective function in (5.1) can be written as:

$$f(W,\mathbf{y};\mathbf{t}) \triangleq d(\mathbf{t} - \sum_{l=1}^{L} W_l\mathbf{y}_l) + \sum_{l=1}^{L} h_{y_l}(\mathbf{y}_l). \tag{5.4}$$

By introducing a splitting variable $\mathbf{z}$, (5.1) becomes:

$$\begin{aligned} &\min_{\{\mathbf{y}_l\},\mathbf{z}} d(\mathbf{t}-\mathbf{z}) + \sum_{l=1}^{L} h_{y_l}(\mathbf{y}_l) \\ &\text{s.t. } \mathbf{z} = \sum_{l=1}^{L} W_l\mathbf{y}_l. \end{aligned} \tag{5.5}$$

The corresponding Lagrangian is then given by

$$L(\{\mathbf{y}_l\},\mathbf{z},\mathbf{x};\mathbf{t}) = d(\mathbf{t}-\mathbf{z}) + \mathbf{x}^T\mathbf{z} + \sum_{l=1}^{L} [h_{y_l}(\mathbf{y}_l) - \mathbf{x}^T W_l\mathbf{y}_l], \tag{5.6}$$

where $\mathbf{x} \in \mathbb{R}^{M\times 1}$ is the Lagrangian multiplier. Its dual function $g(\mathbf{x};\mathbf{t})$ can be then derived as:

$$\begin{aligned} g(\mathbf{x};\mathbf{t}) &\triangleq \inf_{\{\mathbf{y}_l\},\mathbf{z}} L(\{\mathbf{y}_l\},\mathbf{z},\mathbf{x};\mathbf{t}) \\ &= \inf_{\mathbf{z}} [d(\mathbf{t}-\mathbf{z}) + \mathbf{x}^T\mathbf{z}] + \sum_{l=1}^{K} \inf_{\mathbf{y}_l} [h_{y_l}(\mathbf{y}_l) - \mathbf{x}^T W_l\mathbf{y}_l] \end{aligned} \tag{5.7}$$

Let $\mathbf{u} \triangleq \mathbf{t} - \mathbf{z}$, $g(\mathbf{x};\mathbf{t})$ can be further expressed as:

$$\begin{aligned} g(\mathbf{x};\mathbf{t}) &= \inf_{\mathbf{u}}[d(\mathbf{u}) - \mathbf{x}^T\mathbf{u} + \mathbf{x}^T\mathbf{t}] + \sum_{l=1}^{L} \inf_{\mathbf{y}_l}[h_{y_l}(\mathbf{y}_l) - \mathbf{x}^T W_l \mathbf{y}_l] \\ &= -\sup_{\mathbf{u}}[\mathbf{x}^T\mathbf{u} - d(\mathbf{u})] + \mathbf{x}^T\mathbf{t} - \sum_{l=1}^{L} \sup_{\mathbf{y}_l}[\mathbf{x}^T W_l \mathbf{y}_l - h_{y_l}(\mathbf{y}_l)] \\ &= -d^*(\mathbf{x}) + \mathbf{x}^T\mathbf{t} - \sum_{l=1}^{L} h^*_{y_l}(W_l^T\mathbf{x}),\ \mathbf{x} \in \mathbb{X}_d, \end{aligned} \tag{5.8}$$

where $\mathbb{X}_d$ denotes the domain of $d^*(\cdot)$.

Therefore the original problem in (5.1) is equivalent to

$$\begin{aligned} &\min_{\mathbf{x}}\ -g(\mathbf{x};\mathbf{t}) = d^*(\mathbf{x}) - \mathbf{x}^T\mathbf{t} + \sum_{l=1}^{L} h^*_{y_l}(W_l^T\mathbf{x}) \\ &\text{s.t. } \mathbf{x} \in \mathbb{X}_d. \end{aligned} \tag{5.9}$$

This is further reformulated for a distributed problem in [5] as:

$$\begin{aligned} &\min_{\mathbf{x}} \sum_{l=1}^{L} f_l(\mathbf{x};\mathbf{t}), \\ &\text{s.t. } \mathbf{x} \in \mathbb{X}_d, \end{aligned} \tag{5.10}$$

where the cost function for each agent is given by

$$f_l(\mathbf{x};\mathbf{t}) \triangleq \begin{cases} -\frac{\mathbf{x}^T\mathbf{t}}{|\mathcal{N}_l|} + \frac{1}{L}d^*(\mathbf{x}) + h^*_{y_l}(W_l^T\mathbf{x}), & l \in \mathcal{N}_l \\ \frac{1}{L}f^*(\mathbf{x}) + h^*_{y_l}(W_l^T\mathbf{x}), & l \notin \mathcal{N}_l, \end{cases} \tag{5.11}$$

where $\mathcal{N}_l$ is the set of those agents who has access to $\mathbf{t}$.

Finally, this can be formulated as a consensus problem:

$$\begin{aligned} &\min_{\{\mathbf{x}_l\}} \sum_{l=1}^{L} f_l(\mathbf{x}_l;\mathbf{t}), \\ &\text{s.t. } \mathbf{x}_1 = \mathbf{x}_2 = \cdots \mathbf{x}_L. \end{aligned} \tag{5.12}$$

In this study, we used the same image denoising application as in [5] for illustration. Let

$$d(\mathbf{u}) = \frac{1}{2}\|\mathbf{u}\|^2,\ h_y(\mathbf{v}) = \gamma\|\mathbf{v}\|_1 + \frac{\delta}{2}\|\mathbf{v}\|^2. \tag{5.13}$$

Let $L = N$ which implies that each node possesses one single column $\mathbf{w}_i$ of the dictionary and $|\mathcal{N}_l| = N$ which means that all nodes have access to $\mathbf{t}$. The problem can be therefore rewritten as

$$\begin{aligned} &\min_{\{\mathbf{x}_i\}} \sum_{i=1}^{N} f_i(\mathbf{x}_i;\mathbf{t}), \\ &\text{s.t. } \mathbf{x}_1 = \mathbf{x}_2 = \cdots = \mathbf{x}_N, \end{aligned} \tag{5.14}$$

where

$$f_i(\mathbf{x_i};\mathbf{t}) = -\frac{\mathbf{x}_i^T\mathbf{t}}{N} + \frac{1}{N}d^*(\mathbf{x}_i) + h^*_{y_i}(\mathbf{w}_i^T\mathbf{x}_i). \tag{5.15}$$

In the next section, we apply different algorithms to this problem for illustration.

5.3 Algorithms

The updating schemes for each algorithm used in the experiments afterwards are derived in the following subsections.

Given $f_i(\mathbf{x}_i;\mathbf{t})$, its derivative is [5]:

$$\partial_{\mathbf{x}_i} f_i(\mathbf{x}_i;\mathbf{t}) = \begin{cases} \frac{1}{N}(\mathbf{x}_i-\mathbf{t}) + \frac{1}{\delta}\mathbf{w}_i\mathbf{w}_i^T\mathbf{x}_i - \frac{\gamma}{\delta}\mathbf{w}_i, & \mathbf{w}_i^T\mathbf{x}_i > \gamma \\ \frac{1}{N}(\mathbf{x}_i-\mathbf{t}), & |\mathbf{w}_i^T\mathbf{x}_i| \le \gamma \\ \frac{1}{N}(\mathbf{x}_i-\mathbf{t}) + \frac{1}{\delta}\mathbf{w}_i\mathbf{w}_i^T\mathbf{x}_i + \frac{\gamma}{\delta}\mathbf{w}_i, & \mathbf{w}_i^T\mathbf{x}_i < -\gamma. \end{cases} \tag{5.16}$$

5.3.1 Decentralized ADMM

Substituting (5.14) into (3.31) yields

$$\mathbf{x}_i^k = \begin{cases} \left(\left(\frac{1}{N}+\rho|\mathcal{N}(i)|\right)I + \frac{1}{\delta}\mathbf{w}_i\mathbf{w}_i^T\right)^{-1}\left(\sum_{j\in\mathcal{N}(i)}(\rho\mathbf{z}_{ij}^{k-1}-\lambda_{i|j}^{k-1}) + \frac{1}{N}\mathbf{t} + \frac{\gamma}{\delta}\mathbf{w}_i\right), & \mathbf{w}_i^T\mathbf{x}_i > \gamma \\ \left(\frac{1}{N}+\rho|\mathcal{N}(i)|\right)^{-1}\left(\sum_{j\in\mathcal{N}(i)}(\rho\mathbf{z}_{ij}^{k-1}-\lambda_{i|j}^{k-1}) + \frac{1}{N}\mathbf{t}\right), & |\mathbf{w}_i^T\mathbf{x}_i| \le \gamma \quad ,\forall i\in\mathcal{V} \\ \left(\left(\frac{1}{N}+\rho|\mathcal{N}(i)|\right)I + \frac{1}{\delta}\mathbf{w}_i\mathbf{w}_i^T\right)^{-1}\left(\sum_{j\in\mathcal{N}(i)}(\rho\mathbf{z}_{ij}^{k-1}-\lambda_{i|j}^{k-1}) + \frac{1}{N}\mathbf{t} - \frac{\gamma}{\delta}\mathbf{w}_i\right), & \mathbf{w}_i^T\mathbf{x}_i < -\gamma \end{cases} \tag{5.17}$$

$$\mathbf{z}_{ij}^k = \frac{1}{2}\left(\mathbf{x}_i^k + \mathbf{x}_j^k + \frac{1}{\rho}(\lambda_{ij}^{k-1}+\lambda_{ij}^{k-1})\right),\ \forall(i,j)\in\mathcal{E} \tag{5.18}$$

$$\lambda_{ij}^k = \lambda_{ij}^{k-1} + \rho(\mathbf{x}_i^k - \mathbf{z}_{ij}^k),\ \forall(i,j)\in\mathcal{E} \tag{5.19}$$

5.3.2 Synchronous Jacobi ADMM

From (3.34), we can obtain that

$$\mathbf{x}_i^k = \begin{cases} \left(\left(\frac{1}{N}+\rho|\mathcal{N}(i)|\right)I + \frac{1}{\delta}\mathbf{w}_i\mathbf{w}_i^T\right)^{-1}\left(\sum_{\substack{j\in\mathcal{N}(i)\\ j>i}}\lambda_{e_{ij}}^{k-1} - \sum_{\substack{j\in\mathcal{N}(i)\\ j<i}}\lambda_{e_{ij}}^{k-1} + \sum_{j\in\mathcal{N}(i)}\rho x_j^{k-1} + \frac{1}{N}\mathbf{t} + \frac{\gamma}{\delta}\mathbf{w}_i\right), & \mathbf{w}_i^T\mathbf{x}_i > \gamma \\ \left(\frac{1}{N}+\rho|\mathcal{N}(i)|\right)^{-1}\left(\sum_{\substack{j\in\mathcal{N}(i)\\ j>i}}\lambda_{e_{ij}}^{k-1} - \sum_{\substack{j\in\mathcal{N}(i)\\ j<i}}\lambda_{e_{ij}}^{k-1} + \sum_{j\in\mathcal{N}(i)}\rho x_j^{k-1} + \frac{1}{N}\mathbf{t}\right), & |\mathbf{w}_i^T\mathbf{x}_i| \le \gamma \quad ,\forall i\in\mathcal{V} \\ \left(\left(\frac{1}{N}+\rho|\mathcal{N}(i)|\right)I + \frac{1}{\delta}\mathbf{w}_i\mathbf{w}_i^T\right)^{-1}\left(\sum_{\substack{j\in\mathcal{N}(i)\\ j>i}}\lambda_{e_{ij}}^{k-1} - \sum_{\substack{j\in\mathcal{N}(i)\\ j<i}}\lambda_{e_{ij}}^{k-1} + \sum_{j\in\mathcal{N}(i)}\rho x_j^{k-1} + \frac{1}{N}\mathbf{t} - \frac{\gamma}{\delta}\mathbf{w}_i\right), & \mathbf{w}_i^T\mathbf{x}_i < -\gamma \end{cases} \tag{5.20}$$

$$\lambda_{e_{ij}}^k = \lambda_{e_{ij}}^{k-1} + \rho(x_i - x_j),\ \forall(i,j)\in\mathcal{E}, i<j. \tag{5.21}$$

5.3.3 Asynchronous Jacobi ADMM

By slightly modifying the synchronous Jacobi ADMM, we obtain the updating scheme for the asynchronous Jacobi ADMM:

$$\mathbf{x}_i^k = \begin{cases} \left((\frac{1}{N}+\rho|\mathcal{N}(i)|)I+\frac{1}{\delta}\mathbf{w}_i\mathbf{w}_i^T\right)^{-1}\left(\sum\limits_{\substack{j\in\mathcal{N}(i)\\ j>i}}\lambda_{e_{ij}}^{k-1}-\sum\limits_{\substack{j\in\mathcal{N}(i)\\ j<i}}\lambda_{e_{ij}}^{k-1}+\sum\limits_{j\in\mathcal{N}(i)}\rho x_j^{k-1}+\frac{1}{N}\mathbf{t}+\frac{\gamma}{\delta}\mathbf{w}_i\right), & \mathbf{w}_i^T\mathbf{x}_i>\gamma,\ i=p,q,\\ (\frac{1}{N}+\rho|\mathcal{N}(i)|)^{-1}\left(\sum\limits_{\substack{j\in\mathcal{N}(i)\\ j>i}}\lambda_{e_{ij}}^{k-1}-\sum\limits_{\substack{j\in\mathcal{N}(i)\\ j<i}}\lambda_{e_{ij}}^{k-1}+\sum\limits_{j\in\mathcal{N}(i)}\rho x_j^{k-1}+\frac{1}{N}\mathbf{t}\right), & |\mathbf{w}_i^T\mathbf{x}_i|\le\gamma,\ i=p,q\\ \left((\frac{1}{N}+\rho|\mathcal{N}(i)|)I+\frac{1}{\delta}\mathbf{w}_i\mathbf{w}_i^T\right)^{-1}\left(\sum\limits_{\substack{j\in\mathcal{N}(i)\\ j>i}}\lambda_{e_{ij}}^{k-1}-\sum\limits_{\substack{j\in\mathcal{N}(i)\\ j<i}}\lambda_{e_{ij}}^{k-1}+\sum\limits_{j\in\mathcal{N}(i)}\rho x_j^{k-1}+\frac{1}{N}\mathbf{t}-\frac{\gamma}{\delta}\mathbf{w}_i\right), & \mathbf{w}_i^T\mathbf{x}_i<-\gamma,\ i=p,q,\\ \mathbf{x}_i^{k-1}, & i\neq p,q, \end{cases} \tag{5.22}$$

$$\lambda_{ij}^k = \begin{cases} \lambda_{e_{ij}}^{k-1}+\rho(x_i-x_j), & (i,j)=(p,q),\quad i<j,\\ \lambda_{ei}^{k-1}, & (i,j)\neq(p,q). \end{cases} \tag{5.23}$$

5.3.4 Asynchronous ADMM

The updating scheme for the asynchronous ADMM is derived from (3.44) to (3.46) as follow:

$$\mathbf{x}_i^k = \begin{cases} \left((\frac{1}{N}+\rho|\mathcal{N}(i)|)I+\frac{1}{\delta}\mathbf{w}_i\mathbf{w}_i^T\right)^{-1}\left(\sum\limits_{j\in\mathcal{N}(i)}(\rho\mathbf{z}_{ij}^{k-1}-\lambda_{i|j}^{k-1})+\sum\limits_{j\in\mathcal{N}(i)}\rho x_j^{k-1}+\frac{1}{N}\mathbf{t}+\frac{\gamma}{\delta}\mathbf{w}_i\right), & \mathbf{w}_i^T\mathbf{x}_i>\gamma,\ i=p,q,\\ (\frac{1}{N}+\rho|\mathcal{N}(i)|)^{-1}\left(\sum\limits_{j\in\mathcal{N}(i)}(\rho\mathbf{z}_{ij}^{k-1}-\lambda_{i|j}^{k-1})+\sum\limits_{j\in\mathcal{N}(i)}\rho x_j^{k-1}+\frac{1}{N}\mathbf{t}\right), & |\mathbf{w}_i^T\mathbf{x}_i|\le\gamma,\ i=p,q,\\ \left((\frac{1}{N}+\rho|\mathcal{N}(i)|)I+\frac{1}{\delta}\mathbf{w}_i\mathbf{w}_i^T\right)^{-1}\left(\sum\limits_{j\in\mathcal{N}(i)}(\rho\mathbf{z}_{ij}^{k-1}-\lambda_{i|j}^{k-1})+\sum\limits_{j\in\mathcal{N}(i)}\rho x_j^{k-1}+\frac{1}{N}\mathbf{t}-\frac{\gamma}{\delta}\mathbf{w}_i\right), & \mathbf{w}_i^T\mathbf{x}_i<-\gamma,\ i=p,q,\\ \mathbf{x}_i^{k-1}, & i\neq p,q, \end{cases} \tag{5.24}$$

$$z_{ij}^k = \begin{cases} \frac{1}{2}(x_i^k+x_j^k+\frac{1}{\rho}(\lambda_{ij}^{k-1}+\lambda_{ij}^{k-1})), & (i,j)=(p,q)\\ z_{ij}^{k-1}, & (i,j)\neq(p,q) \end{cases} \tag{5.25}$$

$$\lambda_{ij}^k = \begin{cases} \lambda_{ij}^{k-1}+\rho(x_i^k-z_{ij}^k), & (i,j)=(p,q)\\ \lambda_{ei}^{k-1}, & (i,j)\neq(p,q) \end{cases} \tag{5.26}$$

5.3.5 Synchronous PDMM

Let $P_{p,ij} = \rho I$ and $P_{d,ij} = P_{p,ij}^{-1}$, then the updating scheme of the synchronous PDMM is

$$\mathbf{x}_i^k = \begin{cases} \left(\left(\frac{1}{N}+\rho|\mathcal{N}(i)|\right)I + \frac{1}{\delta}\mathbf{w}_i\mathbf{w}_i^T\right)^{-1}\left(\sum_{j\in\mathcal{N}(i)}(\rho\mathbf{x}_j^{k-1}+A_{ij}^T\lambda_{j|i}^{k-1}) + \frac{1}{N}\mathbf{t} + \frac{\gamma}{\delta}\mathbf{w}_i\right), & \mathbf{w}_i^T\mathbf{x}_i > \gamma \\ \left(\frac{1}{N}+\rho|\mathcal{N}(i)|\right)^{-1}\left(\sum_{j\in\mathcal{N}(i)}(\rho\mathbf{x}_j^{k-1}+A_{ij}^T\lambda_{j|i}^{k-1}) + \frac{1}{N}\mathbf{t}\right), & |\mathbf{w}_i^T\mathbf{x}_i| \le \gamma \quad , \forall i \in \mathcal{V} \\ \left(\left(\frac{1}{N}+\rho|\mathcal{N}(i)|\right)I + \frac{1}{\delta}\mathbf{w}_i\mathbf{w}_i^T\right)^{-1}\left(\sum_{j\in\mathcal{N}(i)}(\rho\mathbf{x}_j^{k-1}+A_{ij}^T\lambda_{j|i}^{k-1}) + \frac{1}{N}\mathbf{t} - \frac{\gamma}{\delta}\mathbf{w}_i\right), & \mathbf{w}_i^T\mathbf{x}_i < -\gamma \end{cases} \tag{5.27}$$

$$\lambda_{i|j}^k = \lambda_{j|i}^{k-1} - \rho(A_{ij}x_i^k + A_{ji}x_j^{k-1}),\ \forall (i,j) \in \mathcal{E}, \tag{5.28}$$

with

$$A_{ij} = \begin{cases} I, & i < j \\ -I, & j < i \end{cases} . \tag{5.29}$$

5.3.6 Asynchronous PDMM

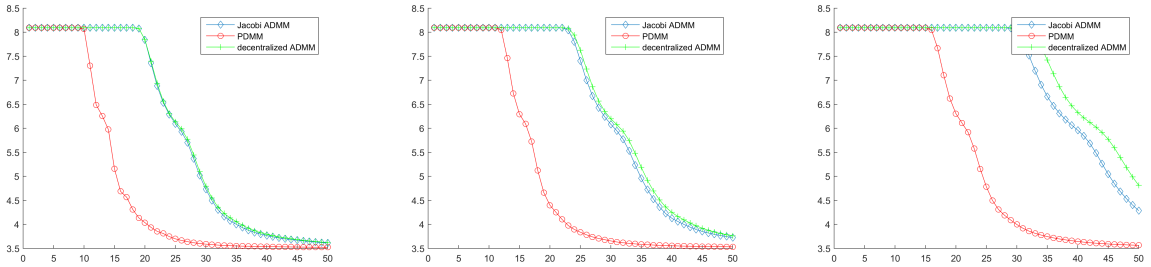
Applying the same setting as for the synchronous PDMM yields

$$\mathbf{x}_i^k = \begin{cases} \left(\left(\frac{1}{N}+\rho|\mathcal{N}(i)|\right)I + \frac{1}{\delta}\mathbf{w}_i\mathbf{w}_i^T\right)^{-1}\left(\sum_{j\in\mathcal{N}(i)}(\rho\mathbf{x}_j^{k-1}+A_{ij}^T\lambda_{j|i}^{k-1}) + \frac{1}{N}\mathbf{t} + \frac{\gamma}{\delta}\mathbf{w}_i\right), & \mathbf{w}_i^T\mathbf{x}_i > \gamma,\ i \text{ is activated} \\ \left(\frac{1}{N}+\rho|\mathcal{N}(i)|\right)^{-1}\left(\sum_{j\in\mathcal{N}(i)}(\rho\mathbf{x}_j^{k-1}+A_{ij}^T\lambda_{j|i}^{k-1}) + \frac{1}{N}\mathbf{t}\right), & |\mathbf{w}_i^T\mathbf{x}_i| \le \gamma,\ i \text{ is activated} \\ \left(\left(\frac{1}{N}+\rho|\mathcal{N}(i)|\right)I + \frac{1}{\delta}\mathbf{w}_i\mathbf{w}_i^T\right)^{-1}\left(\sum_{j\in\mathcal{N}(i)}(\rho\mathbf{x}_j^{k-1}+A_{ij}^T\lambda_{j|i}^{k-1}) + \frac{1}{N}\mathbf{t} - \frac{\gamma}{\delta}\mathbf{w}_i\right), & \mathbf{w}_i^T\mathbf{x}_i < -\gamma,\ i \text{ is activated} \\ \mathbf{x}_i^{k-1}, & i \text{ is not activated} \end{cases} , \tag{5.30}$$

$$\lambda_{i|j}^k = \begin{cases} \lambda_{j|i}^{k-1} - \rho(A_{ij}x_i^k + A_{ji}x_j^{k-1}), & i \text{ is activated}, \\ \lambda_{i|j}^{k-1}, & i \text{ is not activated}. \end{cases} \tag{5.31}$$

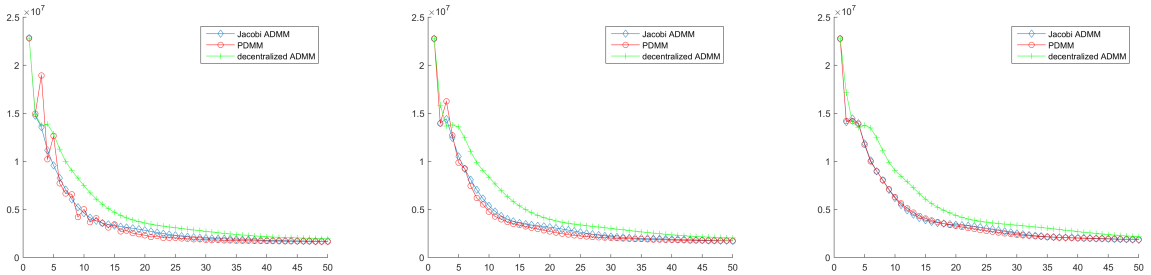
5.4 Experimental Results

The experiments have been done on two datasets, which we refer as the Lena dataset and the IML dataset [15]. For each dataset, we have trained a separate dictionary for experiments. One part of the datasets were used to train the dictionaries, while another part of the datasets were used for validation. In these experiments, we used randomly generated graphs with 100 nodes. The possibility that each two nodes are connected is 0.5. We tested the algorithms with three package loss rate, namely 0, 0.2 and 0.4. To test the performance of image denoising, we distorted the original images with Gaussian noise.



(a) Without package loss (b) With package loss rate of 0.2 (c) With package loss rate of 0.4

Figure 5.1: *Performance of synchronous algorithms on the Lena dataset*



(a) Without package loss (b) With package loss rate of 0.2 (c) With package loss rate of 0.4

Figure 5.2: *Performance of synchronous algorithms on the IML dataset*

5.4.1 Synchronous algorithms

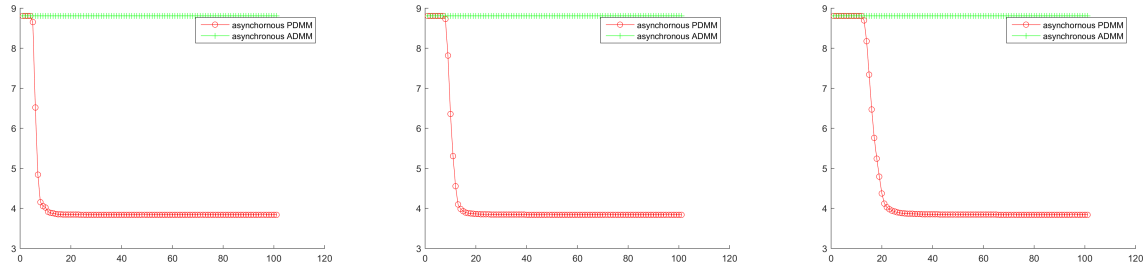
The results for the Lena dataset are shown in Figure 5.1, where the curves illustrate the relation between objective function value with number of iterations in average. Similarly, the convergence curves for the IML dateset are shown in Figure 5.2.

5.4.2 Asynchronous algorithms

The results of asynchronous algorithms for the Lena dataset are shown in Figure 5.3, whereas the results for the IML dataset are shown in Figure 5.4. In these experiments, the asynchronous Jacobi ADMM algorithm diverged too fast. Therefore its performances are not shown in the figures.

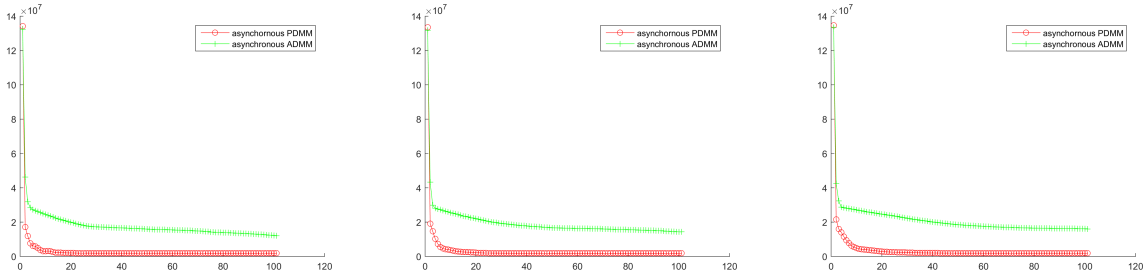
5.4.3 Discussion

The results for these two datasets are not exactly the same. From these results we can conclude that the synchronous PDMM always outperforms the decentralized ADMM in the experiments. The synchronous Jacobi ADMM converges much slower than the synchronous PDMM in the experiments using the Lena dataset, whereas they have similar convergence rates for the other dataset. When there is package losses, the synchronous PDMM presents a robust performance on both datasets. These results conform the conclusion we took in the previous case study.



(a) Without package loss (b) With package loss rate of 0.2 (c) With package loss rate of 0.4

Figure 5.3: *Performance of asynchronous algorithms on the Lena dataset*



(a) Without package loss (b) With package loss rate of 0.2 (c) With package loss rate of 0.4

Figure 5.4: *Performance of asynchronous algorithms on the IML dataset*

On the other hand, the comparison between the asynchronous algorithms indicates again that the asynchronous PDMM not only converges fast, but also is robust against package losses. Unlike its fast convergence rate in the distributed averaging problem, the performance of the asynchronous Jacobi ADMM in this case is poor since it does not converge.

Informal experiments imply that the results do not differ much when different values of ρ are used.

As a whole, it can be seen that the convergence behavior of the Jacobi ADMM is not clear. In contrast, the PDMM has a linear and robust convergence rate. Furthermore, the PDMM outperforms the decentralized ADMM and the asynchronous ADMM in all experiments done in this study.

Chapter 6

Convergence Rate Analysis

From the experiment results illustrated in Chapters 4 and 5, we observe a linear convergence rate for the PDMM. In this chapter, we discuss several methods for proving the convergence rate. Afterwards we show that under certain assumptions, the PDMM algorithm indeed a has linear convergence rate.

6.1 The Method Using Variational Inequality

In [11], a convergence rate of $O(1/K)$ has been proven using variational inequality (VI), which was also applied to proving $O(1/K)$ convergence rate in [16]. Both of them used the fact that for closed, proper and convex functions f_1 and f_2, we have

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} f_1(\mathbf{w}) + f_2(\mathbf{w}) = \arg\min_{\mathbf{w}} f_1(\mathbf{w}) + \mathbf{w}^T g(\mathbf{w}^*), \tag{6.1}$$

where $g(\mathbf{w}^*) = \partial_{\mathbf{w}} f_2(\mathbf{w}^*)$. As a consequence, we have

$$f_1(\mathbf{w}) - f_1(\mathbf{w}^*) + (\mathbf{w} - \mathbf{w}^*)^T g(\mathbf{w}^*) \geq 0. \tag{6.2}$$

The method used in [11] can be summarized as follow:

1. Define a cost function $p(\mathbf{w})$
2. Show the equivalence between finding an optimal solution and letting the following equation hold for all feasible $\mathbf{w}$
$$p(\mathbf{w}) - p(\mathbf{w}^*) + (\mathbf{w} - \mathbf{w}^*)^T g(\mathbf{w}^*) = 0 \tag{6.3}$$
3. Prove the following inequality
$$0 \leq p(\mathbf{w}^k) - p(\mathbf{w}^*) + (\mathbf{w}^k - \mathbf{w}^*)^T g(\mathbf{w}^*) \leq u^k \tag{6.4}$$
4. Define $\bar{\mathbf{w}}^K = \frac{1}{K}\sum_{k=1}^{K} \mathbf{w}^k$ and prove for a constant U
$$0 \leq p(\bar{\mathbf{w}}^k) - p(\mathbf{w}^*) + (\bar{\mathbf{w}}^k - \mathbf{w}^*)^T g(\mathbf{w}^*) \leq \frac{U}{K} = O(\frac{1}{K}) \tag{6.5}$$

This method can be applied to an algorithm to prove a $O(1/K)$ convergence rate in average. To prove a linear convergence rate, however, we need an exponential term on the right-hand side of (6.5), which is hard to find.

6.2 The Method for the gradient descent algorithm

In [12], the assumption of strongly convex objective function was made to prove the linear convergence rate of the gradient descent algorithm. This strategy was applied to two kinds of line search methods for the gradient descent algorithm. It first proved that for a constant α, the following inequality holds

$$f(\mathbf{x}^k) - f(\mathbf{x}^*) \leq f(\mathbf{x}^{k-1}) - f(\mathbf{x}^*) - \alpha\|\nabla f(\mathbf{x}^{k-1})\|^2. \tag{6.6}$$

Afterwards, substituting (3.70) into (6.6) yields

$$f(\mathbf{x}^k) - f(\mathbf{x}^*) \leq f(\mathbf{x}^{k-1}) - f(\mathbf{x}^*) - 2m\alpha(f(\mathbf{x}^{k-1}) - f(\mathbf{x}^*)) \tag{6.7}$$

$$= c(f(\mathbf{x}^{k-1}) - f(\mathbf{x}^*)), \tag{6.8}$$

where $c = 1 - 2m\alpha < 1$.

This method is also hard to apply on the PDMM and the attempt was failed.

6.3 The Method for the ADMM

The assumption of strongly convex objective function was also made in [7], in which the linear convergence rate of the ADMM for centralized problems was proved. This proof is further extended to distributed consensus problems in [8]. In the latter work, (3.73) was used as the expression of strong convexity. Furthermore, in [8]) the KKT conditions are combined with the updating scheme to obtain the term $\alpha^k - \alpha^*$, in which $\alpha = \mathbf{x}, \mathbf{z}, \lambda$.

Inspired by this method, we derived a restricted proof for the PDMM algorithm. Under certain assumptions, we can prove the synchronous PDMM algorithm has a linear convergence rate when there is no transmission failure.

6.3.1 The assumption on strong convexity

We assume that $f_i(\cdot)$ is strongly convex for all $i \in \mathcal{V}$ and $f_i^*(\cdot)$ is strongly convex for all $(i,j) \in \mathcal{E}$, which means there exist some $m_i, n_i > 0$ so that

$$[\partial_{\mathbf{x}_i} f_i(\mathbf{x}_i^A) - \partial_{\mathbf{x}_i} f_i(\mathbf{x}_i^B)]^T(\mathbf{x}_i^A - \mathbf{x}_i^B) \geq m_i\|\mathbf{x}_i^A - \mathbf{x}_i^B\|^2, \tag{6.9}$$

$$[\partial_{\mathbf{y}_i} f_i^*(\mathbf{y}_i^A) - \partial_{\mathbf{y}_i} f_i^*(\mathbf{y}_i^B)]^T(\mathbf{y}_i^A - \mathbf{y}_i^B) \geq n_i\|\mathbf{y}_i^A - \mathbf{y}_i^B\|^2, \tag{6.10}$$

where $\mathbf{x}_i^A, \mathbf{x}_i^B$ and $\mathbf{y}_i^A, \mathbf{y}_i^B$ are variables in the domain of $f_i(\cdot)$ and $f_i^*(\cdot)$, respectively.

Let A_i^T be a matrix obtained by horizontally concatenating $A_{ij}^T, j \in \mathcal{N}(i)$ and λ_i be a vector obtained by vertically concatenating $\lambda_{i|j}, j \in \mathcal{N}(i)$. It is obvious that

$$A_i^T\lambda_i = \sum_{j \in \mathcal{N}i)} A_{ij}^T\lambda_{i|j} \tag{6.11}$$

Substituting $\mathbf{y}_i = A_i^T\lambda_i$ in (6.10) leads to

$$[\partial_{A_i^T\lambda_i} f_i^*(A_i^T\lambda_i^A) - \partial_{A_i^T\lambda_i} f_i^*(A_i^T\lambda_i^B)]^T(A_i^T\lambda_i^A - A_i^T\lambda_i^B) \geq n_i\|A_i^T\lambda_i^A - A_i^T\lambda_i^B\|^2. \tag{6.12}$$

The left-hand side of (6.12) can be further derived as

$$\begin{aligned}
&[\partial_{A_i^T\lambda_i} f_i^*(A_i^T\lambda_i^A) - \partial_{A_i^T\lambda_i} f_i^*(A_i^T\lambda_i^B)]^T (A_i^T\lambda_i^A - A_i^T\lambda_i^B) \\
=&[\partial_{A_i^T\lambda_i} f_i^*(A_i^T\lambda_i^A) - \partial_{A_i^T\lambda_i} f_i^*(A_i^T\lambda_i^B)]^T A_i^T(\lambda_i^A - \lambda_i^B) \\
=&[A_i\partial_{A_i^T\lambda_i} f_i^*(A_i^T\lambda_i^A) - A_i\partial_{A_i^T\lambda_i} f_i^*(A_i^T\lambda_i^B)]^T (\lambda_i^A - \lambda_i^B) \\
=&[\partial_{\lambda_i} f_i^*(A_i^T\lambda_i^A) - \partial_{\lambda_i} f_i^*(A_i^T\lambda_i^B)]^T (\lambda_i^A - \lambda_i^B) \\
=&\sum_{j\in\mathcal{N}(i)} [\partial_{\lambda_{i|j}} f_i^*(\sum_{j\in\mathcal{N}(i)} A_{ij}^T\lambda_{i|j}^A) - \partial_{\lambda_{i|j}} f_i^*(\sum_{j\in\mathcal{N}i)} A_{ij}^T\lambda_{i|j}^B)]^T (\lambda_{i|j}^A - \lambda_{i|j}^B).
\end{aligned} \tag{6.13}$$

The right-hand side of (6.12) can be worked out as

$$\begin{aligned}
&n_i\|A_i^T\lambda_i^A - A_i^T\lambda_i^B\|^2 \\
=&n_i\|A_i^T(\lambda_i^A - \lambda_i^B)\|^2 \\
\geq&n_i\sigma_{min}^2(A_i)\|\lambda_i^A - \lambda_i^B\|^2.
\end{aligned} \tag{6.14}$$

We hereby assume that λ_i lies in the column space of A_i. One of the cases in which this assumption holds is that A_i is a wide matrix and has full rank. Define

$$n_i' = n_i\sigma_{min}^2(A_i) > 0, \tag{6.15}$$

then we have

$$\begin{aligned}
&n_i\|A_i^T\lambda_i^A - A_i^T\lambda_i^B\|^2 \\
\geq&n_i'\|\lambda_i^A - \lambda_i^B\|^2 \\
=&\sum_{j\in\mathcal{N}(i)} n_i'\|\lambda_{i|j}^A - \lambda_{i|j}^B\|^2
\end{aligned} \tag{6.16}$$

Consequently, we obtain

$$\sum_{j\in\mathcal{N}(i)} [\partial_{\lambda_{i|j}} f_i^*(\sum_{j\in\mathcal{N}(i)} A_{ij}^T\lambda_{i|j}^A) - \partial_{\lambda_{i|j}} f_i^*(\sum_{j\in\mathcal{N}(i)} A_{ij}^T\lambda_{i|j}^B)]^T (\lambda_{i|j}^A - \lambda_{i|j}^B) \geq \sum_{j\in\mathcal{N}(i)} n_i'\|\lambda_{i|j}^A - \lambda_{i|j}^B\|^2. \tag{6.17}$$

6.3.2 The assumption on regularization matrices

As mentioned in Section 3.2.4, by properly choosing the matrices $P_{p,ij}$ and $P_{d,ij}$ such that $P_{p,ij} = P_{d,ij}^{-1}$, the optimization problem with the conjugate functions can be avoid. Under this assumption, (3.63) holds true.

Subtracting (6.21) and (6.22) from (3.63) leads to the equation as follow:

$$(\lambda_{i|j}^k - \lambda_{i|j}^*) + P_{p,ij}[A_{ij}(\mathbf{x}_i^k - \mathbf{x}_i^*)] = (\lambda_{j|i}^{k-1} - \lambda_{j|i}^*) - P_{p,ij}[A_{ji}(\mathbf{x}_j^{k-1} - \mathbf{x}_j^*)]. \tag{6.18}$$

6.3.3 Karush-Kuhn-Tucker conditions

Before proving the convergence rate, we first derive the KKT conditions for the PDMM algorithm. Let $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ be the optimal values that solve the primal and dual problems, then the following optimality (KKT) conditions hold:

$$\partial_{\mathbf{x}_i} f_i(\mathbf{x}_i^*) = \sum_{j\in\mathcal{N}(i)} A_{ij}^T \boldsymbol{\lambda}_{j|i}^* \tag{6.19}$$

$$\partial_{\lambda_{i|j}} f_i^*\Big(\sum_{j\in\mathcal{N}(i)} A_{ij}^T \boldsymbol{\lambda}_{i|j}^*\Big) = \mathbf{c}_{ij} - A_{ji}\mathbf{x}_j^* \tag{6.20}$$

$$\mathbf{c}_{ij} = A_{ij}\mathbf{x}_i^* + A_{ji}\mathbf{x}_j^* \tag{6.21}$$

$$\boldsymbol{\lambda}_{i|j}^* = \boldsymbol{\lambda}_{j|i}^* \tag{6.22}$$

If there exist a point $(\tilde{\mathbf{x}}, \tilde{\boldsymbol{\lambda}})$ which satisfy for $i \in \mathcal{V}$ that

$$A_{ij}\tilde{\mathbf{x}}_i = A_{ij}\mathbf{x}_i^*, \tag{6.23}$$

$$\tilde{\boldsymbol{\lambda}}_i = \boldsymbol{\lambda}_{i|j}^*, \tag{6.24}$$

then we can conclude that $(\tilde{\mathbf{x}}, \tilde{\boldsymbol{\lambda}}_i)$ is an optimal solution. This can be easily verified using the KKT conditions from (6.19) to (6.22).

6.3.4 Convergence analysis on the synchronous updating scheme

From (3.59), we obtain that

$$\mathbf{x}_i^k = \arg\min_{\mathbf{x}_i} f_i(\mathbf{x}_i) - \mathbf{x}_i^T\Big(\sum_{j\in\mathcal{N}(i)} A_{ij}^T \boldsymbol{\lambda}_{j|i}^{k-1}\Big) + \sum_{j\in\mathcal{N}(i)} \frac{1}{2}\|A_{ij}\mathbf{x}_i + A_{ji}\mathbf{x}_j^{k-1} - \mathbf{c}_{ij}\|_{P_{p,ij}}^2 \tag{6.25}$$

Since the objective function of (6.25) is convex, by setting its derivative to 0, we have

$$\partial_{\mathbf{x}_i} f_i(\mathbf{x}_i^k) = \sum_{j\in\mathcal{N}(i)} A_{ij}^T \boldsymbol{\lambda}_{j|i}^{k-1} - \sum_{j\in\mathcal{N}(i)} A_{ij}^T [P_{p,ij}(A_{ij}\mathbf{x}_i^k + A_{ji}\mathbf{x}_j^{k-1} - \mathbf{c}_{ij})] \tag{6.26}$$

By subtracting (6.19) and (6.21) from (6.26), we obtain

$$\partial_{\mathbf{x}_i} f_i(\mathbf{x}_i^k) - \partial_{\mathbf{x}_i} f_i(\mathbf{x}_i^*) = \sum_{j\in\mathcal{N}(i)} A_{ij}^T(\boldsymbol{\lambda}_{j|i}^{k-1} - \boldsymbol{\lambda}_{j|i}^*) - \sum_{j\in\mathcal{N}(i)} A_{ij}^T P_{p,ij}[A_{ij}(\mathbf{x}_i^k - \mathbf{x}_i^*) + A_{ji}(\mathbf{x}_j^{k-1} - \mathbf{x}_j^*)] \tag{6.27}$$

Similarly, it can be derived from (3.60) that

$$\partial_{\lambda_{i|j}} f_i^*\Big(\sum_{j\in\mathcal{N}(i)} A_{ij}^T \boldsymbol{\lambda}_{i|j}^k\Big) = P_{d,ij}(\boldsymbol{\lambda}_{j|i}^{k-1} - \boldsymbol{\lambda}_{i|j}^k) + \mathbf{c}_{ij} - A_{ji}\mathbf{x}_j^{k-1} \tag{6.28}$$

After subtracting (6.20) and (6.22) from (6.28), it becomes

$$\partial_{\lambda_{i|j}} f_i^*\Big(\sum_{j\in\mathcal{N}(i)} A_{ij}^T \boldsymbol{\lambda}_{i|j}^k\Big) - \partial_{\lambda_{i|j}} f_i^*\Big(\sum_{j\in\mathcal{N}(i)} A_{ij}^T \boldsymbol{\lambda}_{i|j}^*\Big) = P_{d,ij}(\boldsymbol{\lambda}_{j|i}^{k-1} - \boldsymbol{\lambda}_{j|i}^*) - R_{d,ij}(\boldsymbol{\lambda}_{i|j}^k - \boldsymbol{\lambda}_{i|j}^*) - A_{ji}(\mathbf{x}_j^{k-1} - \mathbf{x}_j^*) \tag{6.29}$$

Substituting (6.27) and (6.29) into (6.9) and (6.17) and then summing them up over $i \in \mathcal{V}$ and $j \in \mathcal{N}(i)$ yields

$$\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)}[(\lambda_{j|i}^{k-1}-\lambda_{j|i}^{*})-P_{p,ij}A_{ji}(\mathbf{x}_j^{k-1}-\mathbf{x}_j^{*})]^T A_{ij}(\mathbf{x}_i^{k}-\mathbf{x}_i^{*}) \geq \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)}(\|A_{ij}(\mathbf{x}_i^{k}-\mathbf{x}_i^{*})\|_{P_{p,ij}}^2+\frac{m_i}{|\mathcal{N}(i)|}\|\mathbf{x}_i^{k}-\mathbf{x}_i^{*}\|^2), \tag{6.30}$$

$$\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)}[(\lambda_{j|i}^{k-1}-\lambda_{j|i}^{*})-P_{p,ij}A_{ji}(\mathbf{x}_j^{k-1}-\mathbf{x}_j^{*})]^T P_{d,ij}(\lambda_{i|j}^{k}-\lambda_{i|j}^{*}) \geq \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)}(\|\lambda_{i|j}^{k}-\lambda_{i|j}^{*}\|_{P_{d,ij}}^2+n_i'\|\lambda_{i|j}^{k}-\lambda_{i|j}^{*}\|^2). \tag{6.31}$$

The sum of (6.30) and (6.31) is

$$\begin{aligned}&\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)}[(\lambda_{j|i}^{k-1}-\lambda_{j|i}^{*})-P_{p,ij}A_{ji}(\mathbf{x}_j^{k-1}-\mathbf{x}_j^{*})]^T[A_{ij}(\mathbf{x}_i^{k}-\mathbf{x}_i^{*})+P_{d,ij}(\lambda_{i|j}^{k}-\lambda_{i|j}^{*})]\\ \geq&\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)}(\|A_{ij}(\mathbf{x}_i^{k}-\mathbf{x}_i^{*})\|_{P_{p,ij}}^2+\frac{m_i}{|\mathcal{N}(i)|}\|\mathbf{x}_i^{k}-\mathbf{x}_i^{*}\|^2+\|\lambda_{i|j}^{k}-\lambda_{i|j}^{*}\|_{P_{d,ij}}^2+n_i'\|\lambda_{i|j}^{k}-\lambda_{i|j}^{*}\|^2).\end{aligned} \tag{6.32}$$

Under the assumption that $P_{p,ij}=P_{d,ij}^{-1}$, the left-hand side of (6.32) can be derived as

$$\begin{aligned}&\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)}[(\lambda_{j|i}^{k-1}-\lambda_{j|i}^{*})-P_{p,ij}A_{ji}(\mathbf{x}_j^{k-1}-\mathbf{x}_j^{*})]^T[A_{ij}(\mathbf{x}_i^{k}-\mathbf{x}_i^{*})+P_{d,ij}(\lambda_{i|j}^{k}-\lambda_{i|j}^{*})]\\ =&\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)}[(\lambda_{j|i}^{k-1}-\lambda_{j|i}^{*})-P_{p,ij}A_{ji}(\mathbf{x}_j^{k-1}-\mathbf{x}_j^{*})]^T P_{d,ij}[(\lambda_{i|j}^{k}-\lambda_{i|j}^{*})+P_{p,ij}A_{ij}(\mathbf{x}_i^{k}-\mathbf{x}_i^{*})]\end{aligned} \tag{6.33}$$

If we substitute (6.18) and (6.33) into (6.32), we can obtain

$$\begin{aligned}&\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)}[(\lambda_{j|i}^{k-1}-\lambda_{j|i}^{*})-P_{p,ij}A_{ji}(\mathbf{x}_j^{k-1}-\mathbf{x}_j^{*})]^T P_{d,ij}[(\lambda_{i|j}^{k}-\lambda_{i|j}^{*})+P_{p,ij}A_{ij}(\mathbf{x}_i^{k}-\mathbf{x}_i^{*})]\\ =&\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)}[(\lambda_{i|j}^{k}-\lambda_{i|j}^{*})+P_{p,ij}A_{ij}(\mathbf{x}_i^{k}-\mathbf{x}_i^{*})]^T P_{d,ij}[(\lambda_{i|j}^{k}-\lambda_{i|j}^{*})+P_{p,ij}A_{ij}(\mathbf{x}_i^{k}-\mathbf{x}_i^{*})]\\ =&\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)}[\|\lambda_{i|j}^{k}-\lambda_{i|j}^{*}\|_{P_{d,ij}}^2+\|A_{ij}(\mathbf{x}_i^{k}-\mathbf{x}_i^{*})\|_{P_{p,ij}}^2+2(\lambda_{i|j}^{k}-\lambda_{i|j}^{*})^T A_{ij}(\mathbf{x}_i^{k}-\mathbf{x}_i^{*})]\\ \geq&\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)}(\|A_{ij}(\mathbf{x}_i^{k}-\mathbf{x}_i^{*})\|_{P_{p,ij}}^2+\frac{m_i}{|\mathcal{N}(i)|}\|\mathbf{x}_i^{k}-\mathbf{x}_i^{*}\|^2+\|\lambda_{i|j}^{k}-\lambda_{i|j}^{*}\|_{P_{d,ij}}^2+n_i'\|\lambda_{i|j}^{k}-\lambda_{i|j}^{*}\|^2),\end{aligned} \tag{6.34}$$

which yields

$$\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)}2(\lambda_{i|j}^{k}-\lambda_{i|j}^{*})^T A_{ij}(\mathbf{x}_i^{k}-\mathbf{x}_i^{*}) \geq \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)}(\frac{m_i}{|\mathcal{N}(i)|}\|\mathbf{x}_i^{k}-\mathbf{x}_i^{*}\|^2+n_i'\|\lambda_{i|j}^{k}-\lambda_{i|j}^{*}\|^2). \tag{6.35}$$

Again by substituting (6.18) and (6.35) into (6.33), we can also derive

$$\begin{aligned}
&\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)}[(\lambda_{j|i}^{k-1}-\lambda_{j|i}^{*})-P_{p,ij}A_{ji}(\mathbf{x}_j^{k-1}-\mathbf{x}_j^{*})]^T P_{d,ij}[(\lambda_{i|j}^{k}-\lambda_{i|j}^{*})+P_{p,ij}A_{ij}(\mathbf{x}_i^{k}-\mathbf{x}_i^{*})]\\
=&\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)}[(\lambda_{j|i}^{k-1}-\lambda_{j|i}^{*})-P_{p,ij}A_{ji}(\mathbf{x}_j^{k-1}-\mathbf{x}_j^{*})]^T P_{d,ij}[(\lambda_{j|i}^{k-1}-\lambda_{j|i}^{*})-P_{p,ij}A_{ji}(\mathbf{x}_j^{k-1}-\mathbf{x}_j^{*})]\\
=&\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)}\|(\lambda_{j|i}^{k-1}-\lambda_{j|i}^{*})-P_{p,ij}A_{ji}(\mathbf{x}_j^{k-1}-\mathbf{x}_j^{*})\|_{P_{d,ij}}^2\\
=&\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)}\|(\lambda_{i|j}^{k-1}-\lambda_{i|j}^{*})-P_{p,ij}A_{ij}(\mathbf{x}_i^{k-1}-\mathbf{x}_i^{*})\|_{P_{d,ij}}^2\\
=&\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)}[\|\lambda_{i|j}^{k-1}-\lambda_{i|j}^{*}\|_{P_{d,ij}}^2+\|A_{ij}(\mathbf{x}_i^{k-1}-\mathbf{x}_i^{*})\|_{P_{p,ij}}^2-2(\lambda_{i|j}^{k-1}-\lambda_{i|j}^{*})^T A_{ij}(\mathbf{x}_i^{k-1}-\mathbf{x}_i^{*})]\\
\leq&\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)}(\|\lambda_{i|j}^{k-1}-\lambda_{i|j}^{*}\|_{P_{d,ij}}^2+\|A_{ij}(\mathbf{x}_i^{k-1}-\mathbf{x}_i^{*})\|_{P_{p,ij}}^2-\frac{m_i}{|\mathcal{N}(i)|}\|\mathbf{x}_i^{k-1}-\mathbf{x}_i^{*}\|^2+n_i'\|\lambda_{i|j}^{k-1}-\lambda_{i|j}^{*}\|^2)\\
\leq&\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)}(\|\lambda_{i|j}^{k-1}-\lambda_{i|j}^{*}\|_{P_{d,ij}}^2+\|A_{ij}(\mathbf{x}_i^{k-1}-\mathbf{x}_i^{*})\|_{P_{p,ij}}^2)
\end{aligned}\tag{6.36}$$

Therefore (6.32) becomes

$$\begin{aligned}
&\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)}(\|\lambda_{i|j}^{k-1}-\lambda_{i|j}^{*}\|_{P_{d,ij}}^2+\|A_{ij}(\mathbf{x}_i^{k-1}-\mathbf{x}_i^{*})\|_{P_{p,ij}}^2)\\
\geq&\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)}(\|A_{ij}(\mathbf{x}_i^{k}-\mathbf{x}_i^{*})\|_{P_{p,ij}}^2+\frac{m_i}{|\mathcal{N}(i)|}\|\mathbf{x}_i^{k}-\mathbf{x}_i^{*}\|^2+\|\lambda_{i|j}^{k}-\lambda_{i|j}^{*}\|_{P_{d,ij}}^2+n_i'\|\lambda_{i|j}^{k}-\lambda_{i|j}^{*}\|^2).
\end{aligned}\tag{6.37}$$

According to the fact in (3.80), (6.37) can be reformulated as

$$\begin{aligned}
&\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)}(\|A_{ij}(\mathbf{x}_i^{k-1}-\mathbf{x}_i^{*})\|_{P_{p,ij}}^2+\|\lambda_{i|j}^{k-1}-\lambda_{i|j}^{*}\|_{P_{d,ij}}^2)\\
\geq&\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)}(\|A_{ij}(\mathbf{x}_i^{k}-\mathbf{x}_i^{*})\|_{P_{p,ij}}^2+\|A_{ij}(\mathbf{x}_i^{k}-\mathbf{x}_i^{*})\|^2_{\frac{m_i}{|\mathcal{N}(i)|\sigma_{max}^2(A_{ij})}I}+\|\lambda_{i|j}^{k}-\lambda_{i|j}^{*}\|_{P_{d,ij}}^2+\|\lambda_{i|j}^{k}-\lambda_{i|j}^{*}\|_{n_i'I}^2).
\end{aligned}\tag{6.38}$$

Define a new variable β_{ij} for all $(i,j)\in\mathcal{E}$ as

$$\beta_{ij}^{k}=\begin{bmatrix}A_{ij}\mathbf{x}_i^{k}\\ \lambda_{i|j}^{k}\end{bmatrix},\tag{6.39}$$

then (6.38) can be rewritten as

$$\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)}\|\beta_{ij}^{k-1}-\beta_{ij}^{*}\|_{P_{pd,ij}}^2\geq\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}(i)}(\|\beta_{ij}^{k}-\beta_{ij}^{*}\|_{P_{pd,ij}}^2+\|\beta_{ij}^{k}-\beta_{ij}^{*}\|_{P_{mn,ij}}^2),\tag{6.40}$$

where

$$P_{pd,ij}=\begin{bmatrix}P_{p,ij}&0\\0&P_{d,ij}\end{bmatrix},\ P_{mn,ij}=\begin{bmatrix}\frac{m_i}{|\mathcal{N}i|\sigma_{max}^2(A_{ij})}I&0\\0&n_i'I\end{bmatrix}\tag{6.41}$$

It is obvious that $P_{pd,ij}$ and $P_{mn,ij}$ are positive-definite for all $(i,j)\in\mathcal{E}$.

Let $\lambda_{max}(D_{ij})$ and $\lambda_{min}(D_{ij})$ denote the largest and the smallest eigenvalue of the matrix $D_{ij}\succ 0$, respectively. Accordingly, the following inequalities hold:

$$\|\beta_{ij}^{k}-\beta_{ij}^{*}\|_{P_{pd,ij}}^2\leq\lambda_{max}(P_{pd,ij})\|\beta_{ij}^{k}-\beta_{ij}^{*}\|^2\tag{6.42}$$

$$\|\beta_{ij}^{k}-\beta_{ij}^{*}\|_{P_{pd,ij}}^2\geq\lambda_{min}(P_{pd,ij})\|\beta_{ij}^{k}-\beta_{ij}^{*}\|^2\tag{6.43}$$

As a result, we obtain

$$\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}i)}\lambda_{max}(P_{pd,ij})\|\beta_{ij}^{k-1}-\beta_{ij}^{*}\|^2\geq\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}i)}[\lambda_{min}(P_{pd,ij})\|\beta_{ij}^{k}-\beta_{ij}^{*}\|^2+\lambda_{min}(P_{mn,ij})\|\beta_{ij}^{k}-\beta_{ij}^{*}\|^2]. \tag{6.44}$$

Furthermore, we denote $\lambda_{max}(D)$ and $\lambda_{min}(D)$ as the maximum of $\lambda_{max}(D_{ij})$ and the minimum of $\lambda_{min}(D_{ij})$, respectively for all $(i,j)\in\mathcal{E}$. Consequently, we have

$$\lambda_{max}(P_{pd})\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}i)}\|\beta_{ij}^{k-1}-\beta_{ij}^{*}\|^2\geq[\lambda_{min}(P_{pd,})+\lambda_{min}(P_{mn})]\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}i)}\|\beta_{ij}^{k}-\beta_{ij}^{*}\|^2, \tag{6.45}$$

which yields

$$\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}i)}\|\beta_{ij}^{k}-\beta_{ij}^{*}\|^2\leq\frac{\lambda_{max}(P_{pd})}{\lambda_{min}(P_{pd})+\lambda_{min}(P_{mn})}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}i)}\|\beta_{ij}^{k-1}-\beta_{ij}^{*}\|^2 \tag{6.46}$$

Therefore, by properly choosing the parameters, the synchronous PDMM algorithm has a linear convergence rate.

In this convergence analysis, we considered three methods for proving the convergence rate. As discussed above, attempts with two of these methods failed. Nevertheless, we succeeded by using the third method, which was also used for proving the linear convergence rate of the ADMM. As a conclusion, we have shown that the synchronous PDMM has a linear convergence rate under certain assumptions.

Chapter 7

Conclusion

The PDMM is a new algorithm for distributed convex optimization problems. Only a few researches were done on this algorithm, which already imply its fast convergence rate and robustness against package losses. However, we still lack of knowledge to fully understand it.

From this study, we have gained deeper understanding of the new algorithm. We compared it with other state-of-the-art algorithms in two cases for both the synchronous and the asynchronous updating schemes. In the synchronous cases, the PDMM converges faster than the decentralized ADMM, especially when there are package losses. Furthermore, in the experiments, the PDMM also presents better performances than the Jacobi ADMM, which does not even always converge. In the asynchronous cases, the results again indicate that the PDMM is robust against package losses. Compared to the asynchronous ADMM, the PDMM shows much faster convergence rate, while the convergence behavior of the Jacobi ADMM is not clear. Under both updating schemes, we observed a linear convergence rate for the PDMM. As a whole, we conclude that the PDMM outperforms other algorithms in the comparison. It has fast convergence rate and robustness against package losses.

On the other hand, we explored three different methods for proving the linear convergence rate theoretically. Although we failed with two of these methods, we showed that under certain assumptions, the synchronous PDMM does have a linear convergence rate.

As a conclusion, we have studied the PDMM algorithm both empirically and theoretically. Based on its good performance in the applications, we derived its linear convergence rate under certain assumptions. A part of the gap in this algorithm has therefore been filled. Future works can be done in investigating more applications and further proving its convergence rate with less assumptions.

Bibliography

[1] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[2] Brendt Wohlberg. Efficient convolutional sparse coding. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 7173–7177. IEEE, 2014.

[3] Tyler H Summers and John Lygeros. Distributed model predictive consensus via the alternating direction method of multipliers. In *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*, pages 79–84. IEEE, 2012.

[4] Ermin Wei and Asuman Ozdaglar. On the o (1/k) convergence of asynchronous distributed alternating direction method of multipliers. *arXiv preprint arXiv:1307.8254*, 2013.

[5] Zaid J Towfic, Jianshu Chen, and Ali H Sayed. Dictionary learning over large distributed models via dual-admm strategies. In *Machine Learning for Signal Processing (MLSP), 2014 IEEE International Workshop on*, pages 1–6. IEEE, 2014.

[6] Mingyi Hong and Zhi-Quan Luo. On the linear convergence of the alternating direction method of multipliers. *arXiv preprint arXiv:1208.3922*, 2012.

[7] Wei Deng and Wotao Yin. On the global and linear convergence of the generalized alternating direction method of multipliers. Technical report, DTIC Document, 2012.

[8] Wei Shi, Qing Ling, Kun Yuan, Gang Wu, and Wotao Yin. On the linear convergence of the admm in decentralized consensus optimization. 2014.

[9] Guoqiang Zhang and Richard Heusdens. Bi-alternating direction method of multipliers. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 3317–3321. IEEE, 2013.

[10] Guoqiang Zhang, Richard Heusdens, and WB Kleijn. On the convergence rate of the bi-alternating direction method of multipliers. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 3869–3873. IEEE, 2014.

[11] Guoqiang Zhang and Richard Heusdens. Bi-alternating direction method of multipliers over graphs. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015.

[12] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[13] Mingyi Hong and Zhi-Quan Luo. On the linear convergence of the alternating direction method of multipliers. *arXiv preprint arXiv:1208.3922*, 2012.

[14] Huahua Wang, Arindam Banerjee, and Zhi-Quan Luo. Parallel direction method of multipliers. In *Advances in Neural Information Processing Systems*, pages 181–189, 2014.

[15] J Hans van Hateren and Arjen van der Schaaf. Independent component filters of natural images compared with simple cells in primary visual cortex. *Proceedings of the Royal Society of London B: Biological Sciences*, 265(1394):359–366, 1998.

[16] Huahua Wang and Arindam Banerjee. Online alternating direction method. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 1119–1126, 2012.