DISTRIBUTED SOLUTIONS FOR A CLASS OF MULTI-AGENT OPTIMIZATION PROBLEMS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Xiaodong Hou

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2019

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Jianghai Hu, Chair

School of Electrical and Computer Engineering

Dr. Shreyas Sundaram

School of Electrical and Computer Engineering

Dr. Inseok Hwang

School of Aeronautics and Astronautics

Dr. Dengfeng Sun

School of Aeronautics and Astronautics

Approved by:

Dr. Pedro Irazoqui

Head of the School Graduate Program

Dedicated to

My wife, sharer of my life,

My parents, givers of my life.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor, Professor Jianghai Hu, for his exceptional mentoring, guidance, and generous support. His broad and deep knowledge, as well as keen insights towards research has provided constant inspiration for me. He always maintains the highest availability for his students, and also allows them the flexibility to work at their own pace. His expertise together with his caring, gentle personality, made it a warm and pleasant experience working with him.

I am grateful to Professor Shreyas Sundaram, Dengfeng Sun and Inseok Hwang for serving on my Ph.D. committee and giving insightful comments and advice at various stages of this dissertation. Their excellent courses in structure and dynamics of large scale networks, convex optimization, and applied optimal control, not only inspired and brought me to the research area of optimization and control over large-scale networks, but also laid the foundation and contributed to many key parts of this dissertation. I am also thankful to Professor James Braun, Panagiota Karava from Ray W. Herrick Laboratories at Purdue University, and Professor Jie Cai from the University of Oklahoma for introducing me to and collaborating with me on modeling, control and optimization of energy efficient buildings, which helped diversify my research.

I would also like to thank all my colleagues, Yingying Xiao, Dr. Donghwan Lee, Dr. Jaewan Joe and Dr. Donghun Kim for their helpful discussions and feedbacks throughout the years. The presence of many other friends at Purdue, whose names are too numerous to list, made this journey much more enjoyable and memorable, I am grateful to all of them.

Last but certainly not least, no words of thanks are enough for my family who gives me constant support and unconditional love. To my parents, thank you for encouraging me in all my pursuits and inspiring me to follow my dreams. To my wife, my best friend and most precious gift, thank you for not only being there to share happiness along the journey, but also helping me through agonizing times in the most positive way.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

SYMBOLS

$\mathbf{R}$	set of real numbers
$\mathbf{R}_+$	set of positive real numbers
$\mathbf{R}^n$	n-dimensional Euclidean space
$\mathbf{R}^n_+$	positive orthant of n-dimensional Euclidean space
$\mathbf{R}^{n\times m}$	set of all $n \times m$ real matrix
$[L]$	set of positive integers from 1 to L, i.e., $\{1, \ldots, L\}$
x^T	transpose of vector x
$\langle x, y\rangle$	inner product of vectors x and y, i.e., $x^T y$
$\langle x, y\rangle_P$	inner product of vectors x and y induced by symmetric positive definite matrix P, i.e., $x^T P y$
(x, y)	concatenation of vectors x and y, i.e., $(x, y) = [x^T\ y^T]^T$
$(x_i)_{i\in\mathcal{N}}$	concatenation of vectors based on index set $\mathcal{N}$
A^T	transpose of matrix A
$A \succ 0$ $(A \succeq 0)$	matrix A is symmetric and positive (semi) definite
$A \succ B$ $(A \succeq B)$	matrix $A - B$ is symmetric and positive (semi) definite
$\otimes$	Kronecker product
I_n	$n \times n$ identity matrix
$0_{n\times m}$	$n \times m$ zero matrix
1_n	$n \times 1$ vector of all 1s
$\mathrm{diag}(A_1, \ldots, A_L)$	block diagonal matrix with matrices $A_1, \ldots, A_L$ on the diagonal blocks and zeros elsewhere
$\|\cdot\|$ or $\|\cdot\|_2$	Euclidean norm (L_2 norm) of a vector, or spectral radius of a matrix
$\|\cdot\|_1$	L_1 norm of a vector
$\|\cdot\|_\infty$	L_∞ norm of a vector

$\|\cdot\|_P$	P-norm of a vector $x \in \mathbf{R}^n$ defined as $\|x\|_P = \sqrt{x^T P x}$ for matrix $P \in \mathbf{R}^{n\times n}$ and $P \succ 0$
$\mathbf{1}_X(\cdot)$	convex indicator function on convex set X
$P_X(\cdot)$	projection onto set X, i.e., $P_X(x) = \arg\min_{y\in C} \|x-y\|_2$
$\prod_{i=1}^L X_i$	Cartesian product of sets X_i, $i \in [L]$
$\partial_x f$	subdifferential of function f with respect to variable x
gra(T)	graph of set-valued operator T, gra$(T) := \{(x,y) \,\|\, y \in T(x)\}$
Id	identity operator, i.e., Id$(x) = x$
A^{-1} (T^{-1})	inverse of matrix A (inverse of set-valued operator T, defined via its graph so that gra$(T^{-1}) = \{(y,x) \,\|\, y \in T(x)\}$)
dom	domain of an operator, i.e., $\mathrm{dom}(T) := \{x \,\|\, T(x) \neq \emptyset\}$
zer	zero set of an operator, i.e., $\mathrm{zer}(T) := \{x \in \mathrm{dom}(T) \,\|\, 0 \in T(x)\}$
Fix	fixed point set of an operator, i.e., $\mathrm{Fix}(T) := \{x \in \mathrm{dom}(T) \,\|\, \{x\} = T(x)\}$
$\|\cdot\|$	cardinality of a set and absolute value for real numbers

ABBREVIATIONS

ADMM	Alternating Directions Method of Multipliers
CCP	Closed, Convex and Proper
DR	Douglas-Rachford
DY	Davis-Yin
KKT	Karush-Kuhn-Tucker
NUM	Network Utility Maximization

ABSTRACT

Hou, Xiaodong Ph.D., Purdue University, May 2019. Distributed Solutions for a Class of Multi-agent Optimization Problems. Major Professor: Jianghai Hu.

Distributed optimization over multi-agent networks has become an increasingly popular research topic as it incorporates many applications from various areas such as consensus optimization, distributed control, network resource allocation, large scale machine learning, etc. Parallel distributed solution algorithms are highly desirable as they are more scalable, more robust against agent failure, align more naturally with either underlying agent network topology or big-data parallel computing framework. In this dissertation, we consider a multi-agent optimization formulation where the global objective function is the summation of individual local objective functions with respect to local agents' decision variables of different dimensions, and the constraints include both local private constraints and shared coupling constraints. Employing and extending tools from the monotone operator theory (including resolvent operator, operator splitting, etc.) and fixed point iteration of nonexpansive, averaged operators, a series of distributed solution approaches are proposed, which are all iterative algorithms that rely on parallel agent level local updates and inter-agent coordination. Some of the algorithms require synchronizations across all agents for information exchange during each iteration while others allow asynchrony and delays. The algorithms' convergence to an optimal solution if one exists are established by first characterizing them as fixed point iterations of certain averaged operators under certain carefully designed norms, then showing that the fixed point sets of these averaged operators are exactly the optimal solution set of the original multi-agent optimization problem. The effectiveness and performances of the proposed algorithms are demonstrated and compared through several numerical examples.

1. INTRODUCTION

1.1 Background and Review

There has been an increasing interest in the analysis and optimization over large-scale networks, which usually consist of multiple entities, or agents, with different local objectives. Some examples include sensor networks [52], cooperative multi-vehicle system [74], smart grid networks [4], social networks [79], etc. In such a large-scale network, each agent either has partial knowledge of the global system, or is responsible for making partial decision for the whole network. The goal of multi-agent optimization is to achieve optimal operating points, decision variables or network equilibrium in terms of certain global objective function, though the coordination of agents with limited information exchange.

Due to the topology of such networks, distributed algorithms naturally draw a tremendous amount of interest [16]. Centralized solution methods not only require the aggregate of all local variables to a central computation unit, which may lead to significant amount of communication and possible privacy issues since local variables of all agents are passed along the agent networks, but are also sensitive to subsystem failures. Distributed solution algorithms on the other hand: 1) are often more computationally efficient because it allocates a global optimization task to multiple local agents; 2) require less communication as agents often only need to communicate with immediate neighbors for the purpose of coordination and information propagation; 3) and are more scalable and resilient to subsystem failure. Many distributed optimization algorithms have been proposed and studied in the literature, we will give a brief review based on problems studied and methodologies used.

Consensus Optimization over Agent Networks This type of problems dates back to [88], in which all the agents share the same objective function. In a multi-agent network consisting of L agents over certain (possibly time-varying) communication topology, the

global objective is that the L agents work cooperatively to minimize $\sum_{i=1}^{L} f_i(x)$, where $x \in \mathbf{R}^n$, and $f_i : \mathbf{R}^n \to \mathbf{R} \cup \{+\infty\}$ is the part of the global objective function that is only known to agent i. These approaches often use consensus models/protocols with doubly stochastic matrix for the "weighted-averaging" of local variables with their immediate neighbors. [57, 63, 64] are the first works to study this problem. An extension to the case of quantized messages was investigated in [62]. Implementations over random networks were studied in [55]. In some other formulations [65], each agent has an additional private constraint set X_i that is only known to itself, and the global optimization problem is

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \sum_{i=1}^{L} f_i(x) \\ \text{subject to} \quad & x \in X = \bigcap_{i=1}^{L} X_i. \end{aligned}$$

This problem is called *constrained consensus optimization*. [50] proposed a randomized synchronous algorithm over time-varying graphs while [51] proposed a gossip-based (one egde activated at one time) asynchronous algorithm over a time-invariant graph. Cases with communication delays and nonidentical constraints were considered in [54, 92]. In addition, state-dependent communication was studied in [56]. In [101], additional global equality and inequality constraints were added to the above formulation, and two distributed primal-dual subgradient algorithms were developed for cases where only equality or inequality constraint is present, respectively; then, the algorithms were extended to nonconvex problems in [102]. Notice that for all the problems and studies in this category, each agent is trying to determine the same global decision variable, a common technique is to create a copy of the global decision variable for each agent, and at the end of the optimization process, all agents must reach consensus on the global decision variable. Apparently, all the local decision variables of different agents have the same dimension, and no explicit couplings between different agents are considered in this formulation.

Distributed Solutions to Convex Feasibility Problems In particular, if $f_i(x) = c$ for some constant c, the constrained consensus optimization problem becomes the *convex feasibility* problem [7, 20], where the objective is to find a common feasible point in the

intersection of a group of convex sets X_i. The convex feasibility problem is the abstraction of many classes of problems arising in e.g., image recovery [20], sensor network localization [13, 42], etc. Convex feasibility problems have been studied extensively in the literature. One important subclass of problems is distributed solutions to linear [61, 82, 3] or nonlinear [32, 31] equations. In [17, 2] convex feasibility problems are cast as the common fixed point problem of a family of nonexpansive operators, for which block iterative type algorithms were developed. Recently, synchronous as well as randomized distributed algorithms were proposed in [97] based on paracontraction operator for convex feasibility problems with sparsely coupled constraints.

Distributed Nonconvex Multi-agent Optimization For many applications in big data and machine learning, such as nonlinear least square, dictionary learning, matrix completion, and low rank approximation [47], global or local objective functions often turn out to be nonconvex. Random gossip protocol was combined with distributed stochastic projection algorithm in [12] to solve $\min_{x\in X}\sum_{i=1}^{L} f_i(x)$ where X is a convex set known to all agents and f_i's are nonconvex functions private to agent i's. A randomly perturbed push-sum gradient algorithm with diminishing step-size was proposed in [87] to unconstrained problem $\min\sum_{i=1}^{L} f_i(x)$. A more general formulation $\min_{x\in X}\sum_{i=1}^{L}(f_i(x)+g(x))$ where f_i's are smooth, nonconvex, nonspearable functions and g is a nonsmooth, convex, nonseparable function was investigated in [26, 81]. Successive convex approximation technique was combined with dynamic consensus mechanism and perturbed push-sum consensus mechanism in [26] and [81], respectively, to show asymptotic convergence to stationary solutions.

Other Problems Besides network problems, multi-agent optimization also has significant implications in image processing [70], compressive sensing [86], large-scale statistics and machine learning [14]. For example, for some large scale machine learning problems, one may also want to decompose a centralized problem into smaller scale subproblems because of either the large training data size or the large number of model parameters.

1.2 Motivations and Contributions

As reviewed in the previous section, most multi-agent optimization formulations in the literature focus on situations where all agents' local decision variables share the same dimension. In addition, explicit couplings in constraints across all agents were rarely considered, sparsity in the couplings was often assumed when dealing with coupling constraints. In this dissertation, we will consider a more general class of convex multi-agent optimization problems:

$$\begin{aligned} \underset{x_1,\ldots,x_L}{\text{minimize}} \quad & \sum_{i=1}^{L} f_i\left(x_i, (x_j)_{j\in\mathcal{N}_i^o}\right) \\ \text{subject to} \quad & \sum_{i=1}^{L} g_i\left(x_i, (x_j)_{j\in\mathcal{N}_i^c}\right) \leq 0, \\ & \left(x_i, (x_j)_{j\in\mathcal{N}_i^l}\right) \in X_i, \quad i \in [L]. \end{aligned}$$

In the above formulation, f_i's and g_i's are closed, convex, proper functions that are private to respective agents, X_i's are local private constraint sets. The details of this formulation will be discussed thoroughly in the next chapter. We only summarize and emphasize its distinctions and generality here. First of all, the local decision variable $x_i \in \mathbf{R}^{n_i}$ for each agent i do not necessarily have the same dimension. Although in theory, local copies of all other agents' decision variables can be created and kept by each agent to force dimension consistency across all agents' local decision variables, this inevitably results in excessive communication and computation and does not scale well as the size of the network grows. Secondly, a more general affine coupling involves local convex functions in agents' local decision variables can be incorporated. Thirdly, neighboring agents may have overlapping variables in each others' objective functions, coupling constraint functions and constraint sets. We assume such overlapping is sparse, i.e., $|\mathcal{N}_i^o|, |\mathcal{N}_i^c|, |\mathcal{N}_i^l| \ll L$. We will show in the next chapter that the most studied multi-agent optimization formulation in the literature, i.e. constrained consensus optimization problem, is a special cases of our formulation.

Recently, formulations with general global couplings similar to the one considered in this dissertation start to attract more attentions from researchers [29, 37, 67, 53, 84]. The push-

sum protocol was incorporated into dual subgradient methods in [37] to handle problems with global coupling constraints $\sum_{i=1}^{L} A_i x_i = b$. The push-sum protocol was also utilized in [53] for a primal-dual algorithm. Another primal-dual algorithm was developed in [67] based on relaxations of primal problem and several duality steps. A consensus-based dual decomposition algorithm was proposed in [84], but it only converges to a neighborhood of an optimal solution. A distributed algorithm combining dual decomposition and proximal minimization was proposed in [29].

Almost all of these existing works use analysis based on classic optimization and duality theory, and have several major drawbacks: 1) they all rely on the assumption that local constraint sets X_i's are compact (hence bounded), which may not hold in some applications. This assumption is often needed in their analysis to guarantee the boundness of dual subgradients; 2) most of them utilize diminishing step sizes, which might be difficult to tune in practice to achieve faster convergence; 3) they are all synchronous algorithms which require one or multiple rounds of synchronization among all agents during each optimization iteration, and it is not apparent how they can be modified to deal with asynchrony and delays; 4) in the case where local objective functions are summations of multiple convex functions with different structures and smoothness properties, the local updates which often involve solving a constrained optimization problem in local decision variable might be difficult to solve.

In this dissertation, we aim to design a series of distributed, parallel algorithms based on monotone operator theory and fixed point iterations of nonexpansive, averaged maps for our general multi-agent optimization formulation, which overcome some, or all of the drawbacks of current literature for similar problems. None of the proposed algorithms needs the assumption on the boundness of local constraint set X_i's, and all of them utilize constant stepsizes. All of the proposed algorithms share the same high level algorithm development methodology as demonstrated in Fig. 1.1: firstly, the optimal solution set of the original multi-agent optimization problem is characterized as the zero sets of certain carefully designed monotone operators; next, the equivalence between these zero sets and the fixed point sets of other operators (resolvent, or generalized resolvent operator) are established;

finally, fixed point iterations which yield distributed, parallel, or even asynchronous updates across agents can be carried out to find such a fixed point, which corresponds to an optimal solution of the multi-agent optimization problem. Table 1.1 summarizes the comparisons between the algorithms proposed in this dissertation and other existing algorithms designed towards similar multi-agent optimization formulations in literature.

Table 1.1.: Comparisons against existing algorithms

	Convergence to optimal solution?	Require bounded local constraint?	Coupling constraints	Central coordinator?	Step sizes	Synchronous or asynchronous?
[37]	Yes	Yes	$=$	No	Diminishing	Synchronous
[67]	Yes	Yes	$\leq$	No	Diminishing	Synchronous
[53]	No	Yes	$\leq$	No	Diminishing	Synchronous
[84]	No	Yes	$\leq$	Yes	Constant	Synchronous
[29]	Yes	Yes	$\leq$ or $=$	No	Diminishing	Synchronous
Algorithm 1-3	Yes	**No**	$\leq$ or $=$	Yes	Constant	Synchronous
Algorithm 4-7	Yes	**No**	$\leq$ or $=$	No	Constant	Synchronous
Algorithm 8	Yes	**No**	$\leq$ or $=$	No	Constant	**Asynchronous**

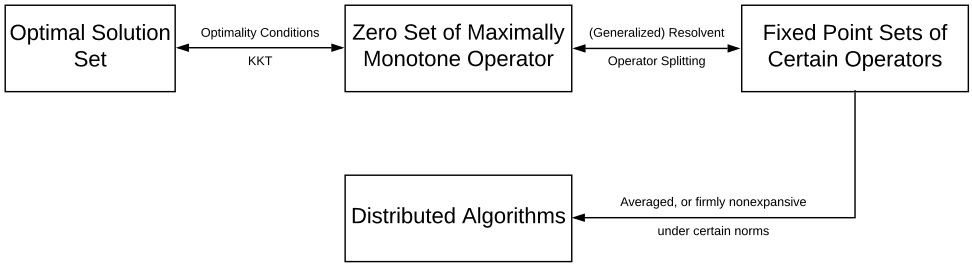

Fig. 1.1.: Algorithm development methodology

Next, we preview the organization of this dissertation and the contributions of each chapter. In Chapter 2, we introduce the multi-agent optimization formulation in detail, many other formulations in the literature will be shown as special cases of our general formulation. Several application examples are given to further motivate our problem formulation.

In Chapter 3, we review some of the theoretical background regarding monotone operators, (firmly) nonexpansive operators, averaged operators, resolvent operators and their fixed point iterations. Operator splitting techniques are also introduced, which play an important role to reduce the complexity of local update for certain cases where the local objective function has a summation structure.

In Chapter 4, by pre-conditioning monotone operators with a positive definite matrix P, we extend the existing results on resolvent operator and its fixed point iteration to a more general setting. We also present new operator splitting algorithms using the generalized resolvent operator and show their convergence. The preconditioning matrix P and the generalized resolvent operator provide more flexibility, and are fundamental tools for designing distributed algorithms.

Two distributed synchronous algorithms with coordinator are proposed in Chapter 5. The first algorithm is a varaint of the alternating direction method of multipliers (ADMM). This algorithm extends the Gauss-Seidel (sequential) type update between two-blocks of variables in the standard ADMM to Jacabi (parallel) update of multi-blocks of variables. The convergence of the algorithm is established by showing each algorithm iteration is indeed one step of the generalized resolvent iteration of a maximally monotone operator preconditioned by a particular positive definite matrix P, and resorting to the theoretical development in Chapter 4. One drawback of the first algorithm is that information exchange includes primal variables of agents, thus privacy issues need to be discussed. The second algorithm overcome this drawback by utilizing the Douglas-Rachford splitting method, and only copies of dual variables are exchanged.

The two algorithms proposed in Chapter 5 enjoy a synchronous parallel updating structure across agents, but both require a central coordinator, which collects information from all agents, updates certain variables, and then broadcasts them to all agents again. This type of algorithms is vulnerable to failure of the central coordinator. In addition, the requirement of central coordinator may not be practical for certain applications. Therefore, in Chapter 6, we consider a truely distributed algorithm framework with which agents only communicate and exchange information with its neighbors defined by an underlying graph.

The monotone operator used in Chapter 5 is augmented to guarantee the consensus across all agents on the dual variable at steady states (zeros of the augmented operator). Then, by utilizing the generalized resolvent iteration as well as generalized DR splitting method, parallel, distributed algorithms without central coordinator are obtained.

Chapter 7 focuses on extending the synchronous distributed algorithms in Chapter 6 to an asynchronous setting which also allows delays. The proposed distributed asynchronous algorithm is characterized as a special instance of random block coordinate fixed point iteration of an averaged operator. Under key assumptions of independent random agent activation and uniform upperbound for delay times, almost sure convergence to a random variable supported by some optimal solution can be guaranteed.

In Chapter 8, several numerical examples are presented. Convergence of different algorithms are demonstrated and compared under different settings. Finally, some concluding remarks and possible future directions are discussed in Chapter 9.

2. MULTI-AGENT OPTIMIZATION FORMULATIONS AND APPLICATION EXAMPLES

In this chapter, we will discuss the multi-agent optimization formulations in detail. Starting from a base standard problem, we will gradually increase its generality while establishing equivalency between different formulations. In addition, several important classes of application examples are introduced to show the significance of studying our multi-agent optimization formulation. In particular, we will demonstrate that the most studied formulation in literature, i.e., constrained consensus optimization over agent networks, is a special case.

2.1 Standard Formulation

In this dissertation, we consider multi-agent optimization problems that have the following base standard formulation:

$$\underset{x}{\text{minimize}} \quad \sum_{i=1}^{L} f_i(x_i) \tag{2.1}$$

$$\text{subject to} \quad Ax = b, \tag{2.2}$$

$$x_i \in X_i, \quad i \in [L], \tag{2.3}$$

where L is the number of agents, $A \in \mathbf{R}^{m\times n}$, $b \in \mathbf{R}^m$, $x = (x_1, \ldots, x_L) \in \mathbf{R}^n$ is the centralized global decision variable and $x_i \in \mathbf{R}^{n_i}$ is the local decision variable for agent i with $\sum_{i=1}^{L} n_i = n$.

Assumption 2.1.1 *Each $f_i : \mathbf{R}^{n_i} \to \mathbf{R} \cup \{+\infty\}$ is an extended-real-valued closed convex proper (CCP) function, i.e., f_i is lower semi-continuous, convex, and $f_i \not\equiv +\infty$.*

Assumption 2.1.2 *Local constraint sets X_i's are nonempty, closed, and convex.*

Assumption 2.1.3 *There exists at least one $z^* = (x_1^*, \ldots, x_L^*, \lambda^*)$ that satisfies the KKT conditions of problem* (2.1)*, which implies that saddle points of the corresponding Lagrange function exist.*

The objective function (2.1) is the summation of local objective functions $f_i(x_i)$, each of which represents a cost that depends on x_i. The first constraint (2.2) couples local decision variables' of all agents together as $Ax = \sum_{i=1}^{L} A_i x_i$, where $A_i \in \mathbf{R}^{m \times n_i}$ contains the columns in A that correspond to x_i. This constraint can be thought of as a shared resource constraint among all agents. Without loss of generality, we assume b has the partition $b = \sum_{i=1}^{L} b_i$, e.g., $b_i = b/L$, $\forall i \in [L]$. On the other hand, (2.3) represents the local, private constraint on each individual agent. In addition, we assume that the local objective function f_i, local feasible set X_i, and coupling constraint parameters that pertain to local decision variables, A_i, b_i, are private and only known to agent i.

Remark 2.1.1 *We do not assume that each local constraint set is compact, thus bounded, as in many existing literatures.*

Remark 2.1.2 *Assumption 2.1.3 guarantees that the optimal solution set of problem* (2.1) *is not empty.*

2.2 Other Formulations

Next, we discuss several important extensions or variations of the base standard formulation (2.1). These formulations are all equivalent to the base standard formulation through simple transformation or introduction of extra variables. Therefore, although all algorithms' development in later chapters will focus on (2.1), they can be easily modified to accommodate other formulations.

Linear Inequality Coupling Constraints Instead of coupling linear equality constraints, some applications might involve coupling linear inequality constraints:

$$\underset{x}{\text{minimize}} \quad \sum_{i=1}^{L} f_i(x_i) \tag{2.4}$$

$$\begin{aligned} \text{subject to} \quad & Ax \leq b, \\ & x_i \in X_i, \quad i \in [L]. \end{aligned}$$

In this case, we can introduce a slack variable x_{L+1}, with private feasible set $X_{L+1} = \mathbf{R}_+^m$, such that $Ax + x_{L+1} = b$, and the new optimization problem becomes

$$\begin{aligned} \underset{(x,\, x_{L+1})}{\text{minimize}} \quad & \sum_{i=1}^{L+1} f_i(x_i) \\ \text{subject to} \quad & \begin{bmatrix} A & I_m \end{bmatrix} \begin{bmatrix} x \\ x_{L+1} \end{bmatrix} = b, \\ & x_i \in X_i, \quad i \in [L+1], \end{aligned}$$

where $f_{L+1}(x_{L+1}) = 0$. Obviously, f_{L+1} and X_{L+1} satisfy the predefined assumptions on convexity, thus the above problem is a special case of the standard formulation in (2.1).

Coupling Objective Functions For some problems, in addition to shared resources coupling constraints between different agents, there might be couplings in the objective function. That is, local objective functions depend on not only local decision variables, but also decision variables from other neighboring agents. For example,

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \sum_{i=1}^{L} f_i\left(x_i, (x_j)_{j \in \mathcal{N}_i}\right) \\ \text{subject to} \quad & Ax = b, \\ & x_i \in X_i, \quad i \in [L]. \end{aligned} \tag{2.5}$$

In this case, the local objective function for agent i not only depends on x_i, but also on x_j where $j \in \mathcal{N}_i$. Here, $\mathcal{N}_i$ is the set of "neighboring" agents outside agent i that affect objective function f_i. For this problem, we can introduce additional variables $(x_{j,i})_{j \in \mathcal{N}_i}$ for agent i such that $\widehat{x}_i := (x_i, (x_{j,i})_{j \in \mathcal{N}_i})$ becomes the new local decision variable for agent i. At the same time, additional consensus constraints $x_{j,i} = x_j$, $\forall j \in \mathcal{N}_i$ are introduced to ensure the consistency between auxiliary variables and neighboring agents' local decision variables.

Notice that the added consensus constraints can be combined with the coupling equality constraints to yield the following problem:

$$\begin{array}{ll} \underset{\widehat{x}}{\text{minimize}} & \displaystyle\sum_{i=1}^{L} f_i\left(\widehat{x}_i\right) \\ \text{subject to} & \widehat{A}\widehat{x} = \widehat{b}, \\ & \widehat{x}_i \in \widehat{X}_i, \quad i \in [L], \end{array}$$

where $\widehat{x} = (\widehat{x}_i)_{i\in[L]}$, $\widehat{X}_i = X_i \times \mathbf{R}^{d_i}$, and $d_i = \sum_{j\in\mathcal{N}_i} n_j$. This becomes the standard formulation in (2.1).

Convex Inequality Coupling Constraints The base standard formulation (2.1) can also be generalized to the following formulation:

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \displaystyle\sum_{i=1}^{L} f_i(x_i) \\ \text{subject to} & \displaystyle\sum_{i}^{L} g_i(x_i) \leq b, \\ & x_i \in X_i, \quad i \in [L], \end{array} \tag{2.6}$$

where g_i's are CCP functions. The base standard formulation (2.4) is a special case of the above formulation where g_i's are affine functions. By adding an extra variable $c_i \in \mathbf{R}^m$ for each agent, optimization problem (2.6) can be equivalently written as

$$\begin{array}{ll} \underset{x,\ (c_i)_{i\in[L]}}{\text{minimize}} & \displaystyle\sum_{i=1}^{L} f_i(x_i) \\ \text{subject to} & \displaystyle\sum_{i}^{L} c_i = b, \\ & g_i(x_i) \leq c_i, \quad x_i \in X_i, \quad i \in [L]. \end{array}$$

Define the new local decision variable for each agent as $\widehat{x}_i := (x_i, c_i)$, and define the new local constraint set as $\widehat{X}_i := \{(x_i, c_i) \in \mathbf{R}^{n_i+m} \,|\, x_i \in X_i,\ g_i(x_i) \leq c_i\}$, which is nonempty,

closed and convex as long as X_i is nonempty, closed and convex. Local objective functions are still CCP functions in $\widehat{x}_i$'s. Then the problem can be cast as

$$\begin{aligned} \underset{\widehat{x}}{\text{minimize}} \quad & \sum_{i=1}^{L} f_i(\widehat{x}_i) \\ \text{subject to} \quad & A\widehat{x} = b, \\ & \widehat{x}_i \in \widehat{X}_i, \quad i \in [L], \end{aligned}$$

which is our base standard formulation with $A_i = [0_{m\times n_i}\ I_m]$.

General Coupling Formulation Previously, we considered three formulations with different coupling structures, all of which can be recast as the standard formulation. Combining and extending those coupling structures, we obtain a multi-agent optimization problem with general coupling structure:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \sum_{i=1}^{L} f_i\left(x_i, (x_j)_{j\in\mathcal{N}_i^o}\right) \\ \text{subject to} \quad & \sum_{i=1}^{L} g_i\left(x_i, (x_j)_{j\in\mathcal{N}_i^c}\right) \le b, \\ & \left(x_i, (x_j)_{j\in\mathcal{N}_i^l}\right) \in X_i, \quad i \in [L]. \end{aligned} \tag{2.7}$$

Sets $\mathcal{N}_i^o$, $\mathcal{N}_i^c$, and $\mathcal{N}_i^l$ denotes the sets of agents that have coupling with agent i in its local objective function, local coupling contraint function, and local constraints, respectively.

Obviously, the standard formulation (2.1) as well as the formulations (2.4), (2.5), (2.6) are all special cases of the general formulation (2.7). It is straightforward to transform formulation (2.7) into the standard formulation using the technique presented previously: 1) introducing additional variables $(x_{j,i})_{j\in\mathcal{N}_i}$ where $\mathcal{N}_i = \mathcal{N}_i^o \cup \mathcal{N}_i^c \cup \mathcal{N}_i^l$ and correspondingly consensus constraints $x_{j,i} = x_j$, $\forall j \in \mathcal{N}_i$, $\forall i \in [L]$; 2) introducing slack variables c_i for each agent and convert coupling inequality constraint $\sum_{i=1}^{L} g_i\left(x_i, (x_j)_{j\in\mathcal{N}_i^c}\right) \le 0$ into $\sum_{i=1}^{L} c_i = b$ and $g_i\left(x_i, (x_j)_{j\in\mathcal{N}_i^c}\right) \le c_i$; 3) combining $g_i\left(x_i, (x_j)_{j\in\mathcal{N}_i^c}\right) \le c_i$ into local constraint.

Therefore, all the formulations in this chapter are equivalent through certain transformations. One only needs to slightly modify Assumption 2.1.3 to guarantee the existence of

solutions under different formulations. For examples, for the general coupling formulation (2.7), Slater's conditions can be used to guarantee the existence of saddle points of the corresponding Lagrange function. For the purpose of algorithm development, we will stick to the base standard formulation (2.1) as it allows for simpler notations.

2.3 Application Examples

Constrained Consensus Optimization in Multi-agent Networks In Chapter 1, it is mentioned that the constrained consensus optimization problem

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \sum_{i=1}^{L} f_i(x) \\ \text{subject to} \quad & x \in X = \bigcap_{i=1}^{L} X_i, \end{aligned}$$

has been extensively studied in the literature. It is straightforward to see that this problem is equivalent to the following formulation

$$\begin{aligned} \underset{\widehat{x}=(x_1,\ldots,x_L)}{\text{minimize}} \quad & \sum_{i=1}^{L} f_i(x_i) \\ \text{subject to} \quad & x_i = x_j, \quad \forall i, j \in [L] \text{ and } i \neq j \\ & x_i \in X_i, \quad i \in [L], \end{aligned}$$

where each agent i keeps a copy of the whole original global decision variable x, and the consensus constraint between different agents can be written in the form of $A\widehat{x} = 0$. Therefore, the constrained consensus optimization problem is a special case of our base standard formulation.

Distributed Model Predictive Control of Linear Systems In model predictive control (MPC) or receding horizon control, at each time step, a new optimal control problem is formulated based on the current measured or estimated state variables, as well as the predicted exogenous input in the predicted horizon. Once the optimal control sequence is determined, only the first control action will be applied to the system and at the next time step, this process is repeated.

For many MPC problems, as the scale of the system or the prediction horizon increases, the centralized problem becomes very difficult to solve. For some other cases consisting of multiple subsystems with possibly time-varying interaction topology, even though computation is not an issue, it will be desirable to have distributed solution methods as they are often more robust to subsystem failure and more suitable for adding or removing subsystems without major modification of the algorithm. Much research effort has been devoted to the so called distributed model predictive control (DMPC) algorithms [80, 91].

A general formulation of a centralized MPC problem for a discrete-time linear system is

$$\underset{x(t+1),\,u(t)}{\text{minimize}} \quad \sum_{t=0}^{N-1}\sum_{i=1}^{L} g_i\left(x_i(t), u_i(t)\right) \tag{2.8}$$

$$\text{subject to} \quad x(t+1) = A_d x(t) + B_d u(t), \quad x(0) = x_0, \tag{2.9}$$

$$x_i(t+1) \in X_i(t+1), \quad u_i(t) \in U_i(t), \tag{2.10}$$

$$i = 1, \ldots, L; \quad t = 0, 1, \ldots, N-1, \tag{2.11}$$

where x_i and u_i are the local state and control variables, respectively; $x = (x_1, \ldots x_L)$, $u = (u_1, \ldots, u_L)$; N denotes the prediction horizon. Constraint (2.9) is the dynamics of the centralized system that needs to be satisfied at all time. Constraint (2.10) could be either the local constraint imposed on the operation of the system or terminal constraint used to guarantee stability [91].

Remark 2.3.1 *The centralized linear dynamics* (2.9) *could consist of multiple local dynamics in terms of the local variables x_i and u_i, and those dynamics might be coupled. If individual local dynamics are uncoupled, then* (2.9) *can be decomposed and incorporated into the local constraint* (2.10)*.*

By stacking the state and control variables together, and further concatenating the dynamics of the system during the whole prediction horizon, the optimization problem (2.8) can be equivalently represented as

$$\underset{z}{\text{minimize}} \quad \sum_{i=1}^{L} f_i(z_i)$$

$$\text{subject to} \quad Az = b, \quad z_i \in Z_i, \quad i \in [L],$$

where $z = (z_1, \ldots, z_L)$, and $z_i = (x_i(1), \ldots, x_i(N), u_i(0), \ldots, u_i(N-1))$; A and b are proper matrix and vector, constructed from A_d, B_d, and x_0; $f_i(z_i) = \sum_{t=0}^{N-1} g_i(x_i(t+1), u_i(t))$. Obviously, this centralized MPC formulation is special case of our base standard formulation.

Commodity Exchange The exchange problem [89, 90] has the following form

$$\begin{aligned} \underset{x_i}{\text{minimize}} \quad & \sum_{i=1}^{L} f_i(x_i) \\ \text{subject to} \quad & \sum_{i=1}^{L} x_i = 0. \end{aligned}$$

This is a classical problem in the economics literature. The interpretation is that each entry of the local decision variable x_i represents how much agent i contributes to the exchange of the corresponding goods or commodity. If $(x_i)_j$ is positive, it means that agent i receives that amount of commodity j through the exchange; if $(x_i)_j$ is negative, it means that agent i sends $|(x_i)_j|$ amount of commodity j to other agents. The constraint $\sum_{i=1}^{L} x_i = 0$ balances the total sent and received amounts of each commodity among all agents. The local objective function is the total operational cost for each agent to sell and buy certain amount of each commodity.

A similar formulation also arises in dynamic network energy management problem [48], where each agent denotes a node in the electricity grid, and commodities denote electricity energy transmitted through a node in a certain time period. The constraint $\sum_{i=1}^{L} x_i = 0$ enforces that during that time the energy flow into a node balances the energy flows out of that node. The exchange problem is apparently one instance of our multi-agent optimization formulation (2.1), where the local decision variable of each agent has the same dimension.

Resource Allocation The resource allocation problem is very similar to the exchange problem, and has the following form

$$\underset{x_i}{\text{minimize}} \quad \sum_{i=1}^{L} f_i(x_i)$$

$$\begin{aligned} \text{subject to} \quad & \sum_{i=1}^{L} x_i = b, \\ & x_i \geq 0, \quad i \in [L]. \end{aligned}$$

Here, each entry $b_j \geq 0$ represents certain type of resources that is to be allocated into L activities or agents. Since the resource received by agent i cannot be negative, we have the extra constraint set $X_i = \mathbf{R}_+^{n_i}$.

One special case that belongs to this class is the network utility maximization (NUM) problem studied in [10, 93, 94]. In an NUM problem, a network of m links are considered, each of which has a finite positive capacity of $b_i > 0$, $i \in [m]$. The network is shared by L sources that are transmitting information along some predetermined routes. The capacity constraints on all links can be compactly represented as $Ax \leq b$, where $A \in \mathbf{R}^{m \times L}$ is the routing matrix defined as

$$A_{ij} = \begin{cases} 1, \text{ if link } i \text{ is on the route of source } j; \\ 0, \text{ otherwise.} \end{cases}$$

A concave utility function f_i is associated with each source i, $\forall i \in [L]$, i.e., $f_i(x_i)$ denotes the utility of source i as a function of the source rate $x_i \geq 0$. Assuming additivity of individual utility functions, the global utility of the whole network is given by $\sum_{i=1}^{L} f_i(x_i)$, and the NUM is formulated as

$$\begin{aligned} \underset{x_i}{\text{maximize}} \quad & \sum_{i=1}^{L} f_i(x_i) \\ \text{subject to} \quad & Ax \leq b, \\ & x_i \geq 0, \quad i \in [L], \end{aligned}$$

which is exactly the same as formulation (2.4).

Distributed Model Fitting and Statistical Machine Learning A fairly general convex formulation for many model fitting problems in statistics and machine learning can be represented as

$$\underset{x}{\text{minimize}} \quad g(Cx - d) + r(x), \tag{2.12}$$

where $x \in \mathbf{R}^n$ is the model parameters to be estimated/learned; $C \in \mathbf{R}^{m\times n}$ is the feature matrix that contains all training examples as row vectors; $d \in \mathbf{R}^m$ is the output vector; $g : \mathbf{R}^m \to \mathbf{R}$ is a convex loss function; and $r : \mathbf{R}^n \to \mathbf{R}$ is a convex regularization function that imposes certain desirable properties on the optimal parameter x^*. Some common regularization functions include Tikhonov regularization or ridge penalty $r(x) = \beta\|x\|_2^2$, where $\beta > 0$; and lasso penalty (least absolute shrinkage and selection operator) $r(x) = \beta\|x\|_1$.

In some cases, there is a modest number of model parameters but a very large number of training examples, i.e., $m \gg n$. Many classical estimation problems belong to this class (large number of low dimensional data). It makes sense to split the training data into smaller groups such that each group of data can be handled by a different processor. Sometimes, the data are naturally stored in a distributed fashion and it might be difficult or impractical to collect all training data into one place, such as social network data, webserver log data, and other cloud computing applications. The following partitions can be made

$$C = \begin{bmatrix} C_1^T \\ \vdots \\ C_L^T \end{bmatrix}, \quad d = \begin{bmatrix} d_1 \\ \vdots \\ d_L \end{bmatrix},$$

where $C_i^T \in \mathbf{R}^{m_i\times n}$ and $d_i \in \mathbf{R}^{m_i}$ with $\sum_{i=1}^L m_i = m$. In this case, C_i^T and d_i corresponds to the ith block of data that is only available to agent i. Then problem (2.12) is transformed into

$$\begin{aligned} \underset{x_i,\, y}{\text{minimize}} \quad & \sum_{i=1}^{L} g_i\left(C_i^T x_i - d_i\right) + r(y), \\ \text{subject to} \quad & x_i - y = 0, \quad i \in [L], \end{aligned}$$

where $x_i, y \in \mathbf{R}^n$. This formulation is a special instance of (2.1) with the centralized decision variable $\widehat{x} = (x_1, \ldots, x_L, y)$, $b = 0$, and

$$A = \begin{bmatrix} I & 0 & \cdots & -I \\ 0 & I & \cdots & -I \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & I & -I \end{bmatrix} \in \mathbf{R}^{nL\times n(L+1)}.$$

In some other cases, the number of model parameters or features far exceeds the number of training examples, i.e., $m \ll n$. For example, in some natural language processing problems, the training examples might be a corpus of documents and features including all possible words and pairs of adjacent words. For problems like this, we can partition the data as follows: $C = [C_1 \cdots C_L]$, with $C_i \in \mathbf{R}^{m \times n_i}$; $x = (x_1, \ldots, x_L)$, with $x_i \in \mathbf{R}^{n_i}$ and $\sum_{i=1}^{L} n_i = n$. The regularization function can be conformably decomposed as $r(x) = \sum_{i=1}^{L} r_i(x_i)$. Hence, problem (2.12) equivalently becomes

$$\underset{x_i}{\text{minimize}} \quad g\left(\sum_{i=1}^{L} C_i x_i - d\right) + \sum_{i=1}^{L} r_i(x_i).$$

After introducing the auxiliary variable y, the above problem can be rewritten as

$$\begin{aligned} \underset{x_i,\, y}{\text{minimize}} \quad & g(y - d) + \sum_{i=1}^{L} r_i(x_i), \\ \text{subject to} \quad & \sum_{i=1}^{L} C_i x_i - y = 0, \end{aligned}$$

which is a special case of (2.1) with the aggregated decision variable $\widehat{x} = (x_1, \ldots, x_L, y)$, $A = [C \ -I]$ and $b = 0$.

3. MONOTONE OPERATOR AND FIXED POINT ITERATION

In this chapter, we will briefly review some basics and facts from monotone operator theory and fixed point iteration of nonexpansive, averaged operators. These are not only the theoretical foundations of many iterative algorithms for convex optimization problems, but also very powerful tools for designing distributed, parallel algorithms in later chapters of the dissertation.

3.1 Operators

An operator T, also called relation, is a point-to-set map, or set-valued map, defined by its graph $\text{gra}(T) := \{(x, y) \,|\, y \in T(x)\}$. If $T(x)$ is a singleton or empty for any x, then T is a function or single-valued.

Some useful sets associated with operators and common operations performed on operators are summarized as follows:

1. (**Domain**) The domain of an operator T is defined as $\text{dom}(T) := \{x \,|\, T(x) \neq \emptyset\}$.

2. (**Zero set**) The zero set an operator T is defined as $\text{zer}(T) := \{x \in \text{dom}(T) \,|\, 0 \in T(x)\}$.

3. (**Fixed point set**) The fixed point set an operator T is defined as $\text{Fix}(T) := \{x \in \text{dom}(T) \,|\, \{x\} = T(x)\}$.

4. (**Summation**) If T_1 and T_2 are two operators, their summation is defined by its graph $\text{gra}(T_1 + T_2) = \{(x, y + z) \,|\, y \in T_1(x),\, z \in T_2(x)\}$.

5. (**Composition**) If T_1 and T_2 are two operators, their composition is defined by its graph $\text{gra}(T_1 \circ T_2) = \{(x, z) \,|\, \exists y \in T_2(x),\, z \in T_1(y)\}$.

6. (**Inversion**) The inverse of an operator T is defined by its graph $\text{gra}(T^{-1}) = \{(y, x) \,|\, y \in T(x)\}$. In general, $T^{-1} \circ T(x) \neq x$, but the equality does hold when T^{-1} is a function.

7. (**Identity**) The identity operator Id, is an operator that maps any point to itself, i.e., $\text{Id}(x) = x$.

3.2 Monotone Operator

Definition 3.2.1 *A set-valued operator $T : \mathbf{R}^n \hookrightarrow \mathbf{R}^n$ is called*

1. ***monotone** if*

$$\langle u - v, x - y \rangle \geq 0, \quad \forall (x, u),\ (y, v) \in \textit{gra}(T).$$

2. ***maximally monotone** if T is monotone, and its graph is not properly contained in the graph of any other monotone operator.*

Some operations that preserve monotonicity are summarized as follows [78].

1. (**Summation**) If T_1 and T_2 are monotone, then $T_1 + T_2$ is also monotone. If T_1 and T_2 are maximally monotone and $\text{dom}(T_1) \cap \text{int}\,\text{dom}(T_2) \neq \emptyset$ where $\text{int}\,\text{dom}(T_2)$ denotes the interior of set $\text{dom}(T_2)$, then $T_1 + T_2$ is also maximally monotone [75].

2. (**Non-negative scaling**) If T is (maximally) monotone, then αT is also (maximally) monotone for $\alpha \geq 0$.

3. (**Inversion**) If T is (maximally) monotone, T^{-1} is also (maximally) monotone.

4. (**Concatenation**) If T_1 and T_2 are (maximally) monotone operators on $\mathbf{R}^m$ and $\mathbf{R}^n$, respectively, then the operator $T_3(x, y) := \{(u, v) \,|\, u \in T_1(x), v \in T_2(y)\}$, is also (maximally) monotone on $\mathbf{R}^{m+n}$.

Some classes of (maximally) monotone operators that play important roles in optimization are summarized below.

1. (**Subdifferential of CCP function**) Suppose $f : \mathbf{R}^n \to \mathbf{R} \cup \{\infty\}$ is a closed, convex, proper (CCP) function. Then its subdifferential operator $\partial_x f(x)$ is maximally monotone.

2. (**Affine function**) An affine function $T(x) = Ax + b$ is maximally monotone if and only if $A + A^T \succeq 0$. One special case is when A is a skew-symmetric matrix, i.e., $A^T = -A$.

3. (**Normal cone operator**) The normal cone operator of a closed convex set C,

$$N_C(x) = \begin{cases} \emptyset, & \text{if } x \notin C \\ \{y \,|\, y^T(z - x) \leq 0,\ \forall z \in C\} & \text{if } x \in C, \end{cases}$$

   is maximally monotone, since $N_C(x)$ is the subdifferential of the convex indicator function defined as

$$\mathbf{1}_C(x) = \begin{cases} 0 & \text{if } x \in C, \\ +\infty & \text{if } x \notin C, \end{cases}$$

   which is CCP if C is closed and nonempty.

4. (**KKT operator**) Consider the following convex optimization problem

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f(x) \\ \text{subject to} & g(x) \leq 0, \quad h(x) = 0, \end{array}$$

   where $f(x)$ and $g(x)$ are closed, convex, proper functions and $h(x)$ is affine. The associated Lagrangian function is $L(x, \lambda, \mu) = f(x) + \lambda^T g(x) + \mu^T h(x)$, $\lambda \geq 0$. The KKT operator is defined as

$$T(x, \lambda, \mu) = \begin{bmatrix} \partial_x L(x, \lambda, \mu) \\ -g(x) + N_{\{\lambda \geq 0\}}(\lambda) \\ -h(x) \end{bmatrix}.$$

   By the operations that preserve monotonicity we summarized before, it is easy to verify the T is maximally monotone. It should also be noted that $0 \in T(x, \lambda, \mu)$ if (x, λ, μ) is a primal-dual solution to the above convex optimization problem.

3.3 Nonexpansive and Averaged Operators

Definition 3.3.1 *Let D be a nonempty subset of $\mathbf{R}^n$ and let $T : D \to \mathbf{R}^n$. Then T is*

1. ***nonexpansive** if*
$$\|T(x) - T(y)\| \leq \|x - y\|, \quad \forall x, y \in D;$$

2. ***firmly nonexpansive** if*
$$\|T(x) - T(y)\|^2 \leq \|x - y\|^2 - \|(x - T(x)) - (y - T(y))\|^2, \quad \forall x, y \in D;$$

3. ***contractive** if*
$$\|T(x) - T(y)\| \leq \beta\|x - y\|, \quad \forall x, y \in D;$$
for some $\beta \in (0, 1)$.

Notice that firm nonexpansiveness implies nonexapnsiveness, thus the former is a stronger notion than the latter. Also, any of the above three properties implies that the underlying operator T is a function as $x = y$ if and only if $T(x) = T(y)$.

Remark 3.3.1 *$\|\cdot\|$ denotes the Euclidean norm in this chapter. Some of the results will be extended to other norms in the next chapter.*

It is straightforward to verify that the composition of nonexpansive operators is nonexpansive; the composition of contractive operators is contractive; the composition of nonexpansive operators with contractive operators is contractive. In addition, suppose $T_3 = \theta T_1 + (1 - \theta)T_2$, with $\theta \in [0, 1]$. If T_1 and T_2 are both nonexpansive, T_3 is also nonexpansive. Next, we summarize some well-known results regarding nonexpansive and firmly nonexpansive operators.

Proposition 3.3.1 ([9]) *Let D be a nonempty subset of $\mathbf{R}^n$ and let $T : D \to \mathbf{R}^n$. Then the following statements are equivalent:*

1. *T is firmly nonexpansive.*

2. *$Id - T$ is firmly nonexpansive.*

3. $2T - Id$ *is nonexpansive.*

4. $\langle x - y, T(x) - T(y)\rangle \geq \|T(x) - T(y)\|^2$, $\forall x, y \in D$.

Definition 3.3.2 *Let D be a nonempty subset of $\mathbf{R}^n$, let $T : D \to \mathbf{R}^n$ and let $\beta > 0$. Then T is called β-cocoercive if $\langle x - y, T(x) - T(y)\rangle \geq \beta\|T(x) - T(y)\|^2$, $\forall x, y \in \mathbf{R}^n$.*

Remark 3.3.2 *Note that T is β-cocoercive if βT is firmly nonexpansive. We can also show that T is nonexpansive if and only if Id $- T$ is β-cocoercive.*

Definition 3.3.3 *An operator T is called **averaged** with constant θ, or θ-**averaged** if $T = \theta Id + (1 - \theta)T_0$ for some $\theta \in (0, 1)$ and some nonexpansive operator T_0.*

An averaged operator is always nonexpansive; composition or convex combinations of multiple averaged operators are still averaged [21, 23].

Proposition 3.3.2 ([9]) *Let D be a nonempty subset of $\mathbf{R}^n$, let $T : D \to \mathbf{R}^n$, and let $\beta \in (0, 1)$. Then the following statements are equivalent:*

1. *T is θ-averaged.*

2. *$(1 - 1/\theta)Id + (1/\theta)T$ is nonexpansive.*

3. *$\|T(x) - T(y)\|^2 \leq \|x - y\|^2 - \frac{1-\theta}{\theta}\|(x - T(x)) - (y - T(y))\|^2$, $\forall x, y \in D$.*

Remark 3.3.3 *Note that an operator is firmly nonexpansive if and only if it is $1/2$-averaged. Therefore, averagedness is a more general notion compared to firmly nonexpansiveness.*

3.4 Fixed Point Iteration

If T is nonexapnsive and $\operatorname{dom} T = \mathbf{R}^n$, then $\operatorname{Fix}(T)$ is closed and convex (could be empty). In addition, if T is contractive and $\operatorname{dom} T = \mathbf{R}^n$, then $\operatorname{Fix}(T)$ contains exactly one point. For an averaged operator $T = \theta\mathrm{Id} + (1 - \theta)T_0$ with $\theta \in (0, 1)$, it is straightforward

to verify that $\mathrm{Fix}(T) = \mathrm{Fix}(T_0)$. One of the most fundamental and significant algorithms in computational mathematics is the fixed point iteration algorithm

$$x^{k+1} = T(x^k), \tag{3.1}$$

for $T : \mathbf{R}^n \to \mathbf{R}^n$ with some initial point $x_0 \in \mathbf{R}^n$. This algorithm dates back over 100 years [6, 73].

Proposition 3.4.1 ([9, 78]) *Suppose operator $T : \mathbf{R}^n \to \mathbf{R}^n$ is contractive. Then the fixed point iteration algorithm* (3.1) *converges linearly to the unique fixed point of T.*

Proposition 3.4.2 ([9, 78]) *Suppose operator $T : \mathbf{R}^n \to \mathbf{R}^n$ is averaged and $\mathrm{Fix}(T) \neq \emptyset$. Then the fixed point iteration algorithm* (3.1) *converges to some $x^* \in \mathrm{Fix}(T)$.*

Corollary 3.4.1 *Suppose operator $T : \mathbf{R}^n \to \mathbf{R}^n$ is firmly nonexpansive and $\mathrm{Fix}(T) \neq \emptyset$. Then the fixed point iteration algorithm* (3.1) *converges to some $x^* \in \mathrm{Fix}(T)$.*

When (3.1) is applied to an averaged operator, it is also called the Mann-Krasnosel'skii iteration [49, 58]. Note that if we apply the fixed point iteration (3.1) to a nonexpansive operator, it does not necessarily converge. One example is given in Fig. 3.1, where operator $T_0 :=$ 45 degree counter-clockwise rotation, is nonexpansive, but the corresponding fixed point iteration does not converge. On the other hand, iterations (3.1) converges when applied to $T = 1/2 \cdot \mathrm{Id} + 1/2 \cdot T_0$.

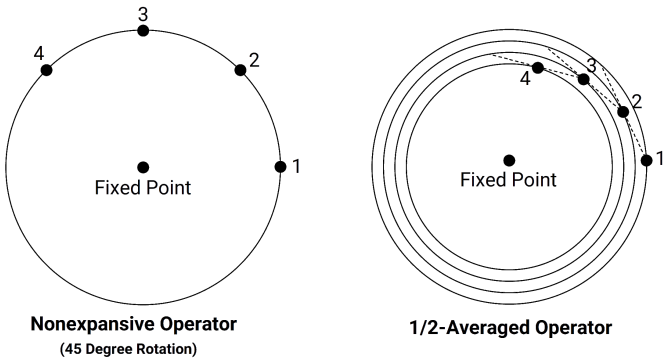

Fig. 3.1.: Fixed point iterations applied to nonexpansive and averaged operators

3.5 Resolvent and Cayley Operator

Definition 3.5.1 *Let T be a set-valued operator on $\mathbf{R}^n$ and $\beta > 0$. The resolvent of T is defined as*

$$R_T = (Id + \beta T)^{-1}. \tag{3.2}$$

The corresponding Cayley operator is defined as $C_T = 2R_T - Id$. The Cayley operator is also called the reflection or reflected resolvent.

Two important resolvent operators that will be used later are:

1. (**Proximal operator**) The resolvent of the subddiferential of a CCP function f is called the *proximal operator* [69], denoted by

$$\underset{\beta f}{\text{prox}}(x) = \arg\min_u \left(f(u) + (1/2\beta)\|u - x\|^2 \right).$$

   To see this, note that if $z = R_{\partial f}(x)$, we have

$$\begin{aligned} z = (\text{Id} + \beta \partial f)^{-1}(x) \quad &\Longleftrightarrow \quad x \in z + \beta \partial f(z) \\ &\Longleftrightarrow \quad 0 \in (z - x)/\beta + \partial f(z) \\ &\Longleftrightarrow \quad z = \arg\min_u \left(f(u) + (1/2\beta)\|u - x\|^2 \right). \end{aligned}$$

2. (**Projection operator**) The resolvent of a normal cone operator corresponding to a nonempty, closed convex set C, is the projection operator onto C, i.e., $R_{N_C}(x) = P_C(x)$. To see this, take $f(x) = \mathbf{1}_C(x)$, we know $\partial f(x) = \partial \mathbf{1}_C(x) = N_C(x)$, thus the resolvent of $N_C(x)$ is the proximal operator of $f(x)$, i.e., $R_{N_C}(x) = \text{prox}_{\beta f}(x) = \arg\min_u \left(\mathbf{1}_C(u) + (1/2\beta)\|u - x\|^2 \right) = P_C(x)$.

Proposition 3.5.1 ([9, 78]) *Let T be a set-valued operator with $\text{dom}(T) \neq \emptyset$ and $\text{dom}(T) \subseteq \mathbf{R}^n$:*

1. *If T is monotone, then C_T is nonexpansive, and R_T is $1/2$-averaged, or firmly nonexpansive.*

2. *If T is further maximally monotone, then* $\operatorname{dom} C_T = \operatorname{dom} R_T = \mathbf{R}^n$.

Proposition 3.5.2 ([78]) *Suppose a set-valued operator T is monotone. Then, $x \in \operatorname{zer}(T)$ if and only if $x \in \operatorname{Fix}(R_T) = \operatorname{Fix}(C_T)$.*

Proof For any $x \in \operatorname{zer}(T)$, we have $0 \in T(x)$, if and only if $x \in (\mathrm{Id} + \beta T)x$ for any $\beta > 0$, if and only if $x = (\mathrm{Id} + \beta T)^{-1}(x) = R_T(x)$, i.e., $x \in \operatorname{Fix}(R_T)$. The second part of the claim follows from the definition of Cayley operator. ■

Proposition 3.5.2 implies that the zero set of a monotone operator T is the same as the fixed point set of its resolvent or Cayley operators. Therefore, finding an optimal solution to the convex optimization problem associated with the operator T is equivalent to finding a fixed point of the firmly nonexpansive operator R_T. If R_T is efficient to evaluate, one can apply the resolvent iteration $x^{k+1} = R_T(x^k)$, for $k = 0, 1 \ldots$ to find a fixed point. This algorithm is also called the proximal point algorithm [15, 59, 77], and always converges to a solution if one exists since R_T is averaged when T is maximally monotone.

3.6 Operator Splitting

When the resolvent or Cayley operators are difficult to evaluate or compute directly, one alternative methodology is operator splitting. Certain monotone operator T admits a natural splitting into the summation of two or three maximally monotone operators, and the task of finding a zero point of T is equivalent to solving $0 \in (T_1 + T_2)x$ or $0 \in (T_1 + T_2 + T_3)x$, where $T_1,\ T_2,\ T_3$ are maximally monotone operators. Operator splitting methods aim to transform such problems into fixed point iterations with operators associated with $T_1,\ T_2,\ T_3$ and their corresponding resolvent, Cayley operators, which are often much more efficient to evaluate or compute compared to those of operator T.

There are many operator splitting methods available, but we will focus on Douglas-Rachford splitting (two operators) and Davis-Yin splitting (three operators) as they require milder assumptions on operators $T_1,\ T_2,\ T_3$ to guarantee convergence and pertain more to the generalization and algorithms development that will be presented later in this dissertation.

We refer interested readers to [9] and [78] for other operator splitting methods and their applications.

Proposition 3.6.1 (Douglas-Rachford Splitting [9]) *Suppose T_1, T_2 are maximally monotone on $\mathbf{R}^n$ and $\operatorname{zer}(T_1+T_2) \neq \emptyset$. Define operator $T_{DR} := Id - 2\alpha R_{T_2} + 2\alpha R_{T_1} \circ (2R_{T_2} - Id)$ where $\alpha \in (0,1)$. Then T_{DR} is α-averaged. Starting from any z^0, for $k = 0, 1, \ldots$, let*

$$w^{k+1} = R_{T_2}(z^k); \tag{3.3a}$$

$$z^{k+1} = z^k + 2\alpha \left(R_{T_1}(2w^{k+1} - z^k) - w^{k+1} \right). \tag{3.3b}$$

The sequence $(z^k)_{k\geq 0}$ converges to some $z^* \in \operatorname{Fix}(T_{DR})$, and the sequence $(w^k)_{k\geq 0}$ converges to $R_{T_2}(z^*) \in \operatorname{zer}(T_1 + T_2)$.

Notice that for Douglas-Rachford splitting, each iteration requires evalutaing R_{T_1} and R_{T_2} each once, and their roles can be switched in (3.3).

Proposition 3.6.2 (Davis-Yin Splitting [25]) *Suppose T_1, T_2 are maximally monotone on $\mathbf{R}^n$; T_3 is β-cocoercive; $\operatorname{zer}(T_1 + T_2 + T_3) \neq \emptyset$. Define operator $T_{DY} := Id - R_{T_2} + R_{T_1} \circ (2R_{T_2} - Id - \gamma T_3 \circ R_{T_2})$ where $\gamma \in (0, 2\beta)$. Then T_{DY} is θ-averaged with coefficient $\theta = \frac{2\beta}{4\beta - \gamma} < 1$. Starting from any z^0, for $k = 0, 1, \ldots$, let*

$$w^{k+1} = R_{T_2}(z^k); \tag{3.4a}$$

$$z^{k+1} = z^k - w^{k+1} + R_{T_1}\left(2w^{k+1} - z^k - \gamma T_3(w^{k+1})\right). \tag{3.4b}$$

The sequence $(z^k)_{k\geq 0}$ converges to some $z^* \in \operatorname{Fix}(T_{DY})$, and the sequence $(w^k)_{k\geq 0}$ converges to $R_{T_2}(z^*) \in \operatorname{zer}(T_1 + T_2 + T_3)$.

Notice that for Davis-Yin splitting, each iteration requires evalutaing R_{T_1}, R_{T_2} and T_3 each once, and the roles of R_{T_1} and R_{T_2} can be switched in (3.4).

4. GENERALIZED RESOLVENT ITERATION AND GENERALIZED OPERATOR SPLITTING

In the previous chapter, we reviewed some well-known facts and results regarding monotone operators, fixed point iteration of averaged maps and resolvent operators. We demonstrated that for many convex optimization problems, the optimal solution set can be characterized by the zero set of certain monotone operators, and an optimal solution can be found by carrying out the corresponding resolvent iteration. In cases where the resolvent of the underlying monotone operator is difficult to compute, but yields certain decomposition of multiple monotone operators, whose resolvent can be computed efficiently, operator splitting methods can be utilized.

In this chapter, we will extend the theoretical results in the previous chapter. In particular, we will define a generalized resolvent operator, which takes the standard resolvent operator as a special case. The averagedness, or firm nonexpansiveness of the generalized resolvent operator will be shown under a new norm. This generalization provides greater flexibility to make the computation of resolvent more efficient, or distributed in nature, which will play a fundamental role in the development of a series of different distributed algorithms later. In addition, the operator splitting methods will be extended with the introduction of generalized resolvent operator.

4.1 Generalized Resolvent and Cayley Operator

Definition 4.1.1 (Generalized Resolvent and Cayley Operator) *Let T be a set-valued operator on $\mathbf{R}^n$ and $P \in \mathbf{R}^{n\times n}$ be a symmetric positive definite matrix. The generalized resolvent of T is defined as*

$$\widehat{R}_T = (P + T)^{-1}P. \tag{4.1}$$

The corresponding generalized Cayley operator is defined as $\widehat{C}_T = 2\widehat{R}_T - Id$.

Remark 4.1.1 *Notice that the standard resolvent and Cayley operator definitions in Definition 3.5.1 are special cases of the above generalized definitions with* $P = \frac{1}{\beta} I_n$.

Definition 4.1.2 *For a symmetric positive definite matrix* $P \in \mathbf{R}^{n\times n}$*, we define an associated vector* P*-norm as* $\|x\|_P = \sqrt{x^T P x}$*,* $\forall x \in \mathbf{R}^n$.

Proposition 4.1.1 *Suppose a set-valued operator* T *is monotone, then* $\widehat{C}_T$ *is nonexpansive with respect to the norm* $\|\cdot\|_P$*, and* $\widehat{R}_T$ *is an* $1/2$*-averaged, or firmly nonexpansive operator with respect to the norm* $\|\cdot\|_P$.

Proof Let $u_1 \in \widehat{R}_T(x_1)$ and $u_2 \in \widehat{R}_T(x_2)$. Then $(P+T)u_1 \ni Px_1$ and $(P+T)u_2 \ni Px_2$. Subtract these to obtain

$$P(u_1 - u_2) + (T(u_1) - T(u_2)) \ni P(x_1 - x_2).$$

Multiplying both sides by $(u_1 - u_2)^T$ and utilizing the monotonicity of operator T yields

$$\begin{aligned} & \|u_1 - u_2\|_P^2 \le (u_1 - u_2)^T P(x_1 - x_2) \\ \iff & \|u_1 - u_2\|_P^2 \le \|x_1 - x_2\|_P^2 - (\|x_1 - x_2\|_P^2 - 2(u_1 - u_2)^T P(x_1 - x_2) + \\ & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \|u_1 - u_2\|_P^2) \\ \iff & \|u_1 - u_2\|_P^2 \le \|x_1 - x_2\|_P^2 - \|(x_1 - u_1) - (x_2 - u_2)\|_P^2. \end{aligned}$$

Since the choice of (x_1, u_1) and (x_2, u_2) is arbitrary, we conclude that $\widehat{R}_T$ is firmly nonexpansive with respect to the norm $\|\cdot\|_P$.

Next, we have $\|\widehat{C}_T(x_1) - \widehat{C}_T(x_2)\|_P^2 = \|2(u_1 - u_2) - (x_1 - x_2)\|_P^2 = 4\|u_1 - u_2\|_P^2 - 4(u_1 - u_2)^T P(x_1 - x_2) + \|x_1 - x_2\|_P^2 \le \|x_1 - x_2\|_P^2$. Thus, $\widehat{C}_T$ is nonexpansive under the norm $\|\cdot\|_P$. Since $\widehat{R}_T$ and $\widehat{C}_T$ are nonexpansive, they are also single-valued. Since $\widehat{R}_T = 1/2 \cdot \mathrm{Id} + 1/2 \cdot \widehat{C}_T$, $\widehat{R}_T$ is also $1/2$-averaged with respect to the norm $\|\cdot\|_P$. ■

Remark 4.1.2 *Suppose* $u = \widehat{R}_T(x)$*. Then* $(P+T)u \ni Px \iff (Id + P^{-1}T)u \ni x$*. Therefore,* $\widehat{R}_T$ *can be equivalently represented as* $\widehat{R}_T = (Id + P^{-1}T)^{-1}$*. Although* P^{-1} *is*

positive definite by assumption, in general $P^{-1}T$ is not necessarily maximally monotone. Therefore, we cannot directly use the result from standard resolvent operator in the previous chapter to conclude the firm nonexpansiveness or averagedness of $\widehat{R}_T$.

Proposition 4.1.2 *Suppose a set-valued operator operator T is monotone. Then, $x \in \mathrm{zer}(T)$ if and only if $x \in \mathrm{Fix}(\widehat{R}_T) = \mathrm{Fix}(\widehat{C}_T)$.*

Proof The first part of the claim follows from

$$\begin{aligned} 0 \in T(x) \quad &\Longleftrightarrow \quad Px \in (P+T)x, \\ &\Longleftrightarrow \quad x = (P+T)^{-1}Px, \\ &\Longleftrightarrow \quad x = \widehat{R}_T(x), \qquad (4.2) \\ &\Longleftrightarrow \quad x \in \mathrm{Fix}(\widehat{R}_T). \qquad (4.3) \end{aligned}$$

The second part of the claim follows from the definition of the generalized Cayley operator. ■

Proposition 4.1.2 implies that the zero set of an operator T is the same as the fixed point set of its generalized resolvent or Cayley operators. Therefore, if the optimal solution set of certain optimization problem can be characterized as the zero set of a proper maximally monotone operator T, we can utilize the following generalized resolvent iteration to find an optimal solution.

Proposition 4.1.3 (Generalized Resolvent Iteration) *Let T be a set-valued operator on $\mathbf{R}^n$ and $P \in \mathbf{R}^{n\times n}$ be a symmetric positive definite matrix. Then, the sequence $(z^k)_{k\geq 0}$ generated by the iteration*

$$z^{k+1} = (P+T)^{-1}Pz^k, \qquad (4.4)$$

converges to some $z^* \in \mathrm{zer}(T)$ if $\mathrm{zer}(T) \neq \emptyset$.

4.2 Generalized Douglas-Rachford Splitting

Proposition 4.2.1 *Suppose $T_1,\ T_2$ are maximally monotone on $\mathbf{R}^n$ and $\operatorname{zer}(T_1+T_2)\neq\emptyset$. Define operator $\widehat{T}_{DR} := Id - 2\alpha\widehat{R}_{T_2} + 2\alpha\widehat{R}_{T_1}\circ(2\widehat{R}_{T_2} - Id)$ where $\alpha\in(0,1)$. Then $\widehat{T}_{DR}$ is α-averaged with respect to the norm $\|\cdot\|_P$. Starting from any z^0, for $k=0,1,\ldots$, let*

$$w^{k+1} = \widehat{R}_{T_2}(z^k); \tag{4.5a}$$

$$z^{k+1} = z^k + 2\alpha\left(\widehat{R}_{T_1}(2w^{k+1}-z^k) - w^{k+1}\right). \tag{4.5b}$$

The sequence $(z^k)_{k\geq 0}$ converges to some $z^*\in\operatorname{Fix}(\widehat{T}_{DR})$, and the sequence $(w^k)_{k\geq 0}$ converges to $w^* = \widehat{R}_{T_2}(z^*)\in\operatorname{zer}(T_1+T_2)$.

Proof Let w^* be a point in zer(T_1+T_2). Then

$$\begin{aligned}
& 0\in(T_1+T_2)w^* \\
\iff\quad & 0\in(P^{-1}T_1+P^{-1}T_2)w^* \\
\iff\quad & 0\in(\mathrm{Id}+P^{-1}T_1)w^* - (\mathrm{Id}-P^{-1}T_2)w^* \\
\iff\quad & 0\in(\mathrm{Id}+P^{-1}T_1)w^* - \widehat{C}_{T_2}\circ(\mathrm{Id}+P^{-1}T_2)w^* \\
\iff\quad & 0\in(\mathrm{Id}+P^{-1}T_1)w^* - \widehat{C}_{T_2}(z^*), \quad \text{for some } z^*\in(\mathrm{Id}+P^{-1}T_2)w^* \\
\iff\quad & \widehat{C}_{T_2}(z^*)\in(\mathrm{Id}+P^{-1}T_1)\circ\widehat{R}_{T_2}(z^*), \quad w^* = \widehat{R}_{T_2}(z^*) \\
\iff\quad & 2\widehat{R}_{T_1}\circ\widehat{C}_{T_2}(z^*) = 2\widehat{R}_{T_2}(z^*), \quad w^* = \widehat{R}_{T_2}(z^*) \\
\iff\quad & (\widehat{C}_{T_1}+\mathrm{Id})\circ\widehat{C}_{T_2}(z^*) = (\widehat{C}_{T_2}+\mathrm{Id})z^*, \quad w^* = \widehat{R}_{T_2}(z^*) \\
\iff\quad & \widehat{C}_{T_1}\circ\widehat{C}_{T_2}(z^*) = z^*, \quad w^* = \widehat{R}_{T_2}(z^*).
\end{aligned}$$

The thrid line to the fourth line is due to the fact that $\widehat{C}_{T_2}\circ(\mathrm{Id}+P^{-1}T_2) = (2(\mathrm{Id}+P^{-1}T_2)^{-1}-\mathrm{Id})\circ(\mathrm{Id}+P^{-1}T_2) = (2\mathrm{Id}-\mathrm{Id}-P^{-1}T_2) = \mathrm{Id}-P^{-1}T_2$.

Therefore, a point $w^*\in\operatorname{zer}(T_1+T_2)$ can be obtained as $w^* = \widehat{R}_{T_2}(z^*)$ where z^* is a fixed point of map $\widehat{C}_{T_1}\circ\widehat{C}_{T_2}$, which is nonexpansive under the norm $\|\cdot\|_P$ since it is the composition of two nonexpansived maps under the norm $\|\cdot\|_P$. Therefore $\widehat{T}_{DR} := (1-\alpha)\mathrm{Id} + \alpha\widehat{C}_{T_1}\circ\widehat{C}_{T_2} = \mathrm{Id} - 2\alpha\widehat{R}_{T_2} + 2\alpha\widehat{R}_{T_1}\circ(2\widehat{R}_{T_2}-\mathrm{Id})$ where $\alpha\in(0,1)$ is α-averaged with respect to the norm $\|\cdot\|_P$. The sequence $(z^k)_{k\geq 0}$ generated by iteration (4.5)

converges to a fixed point z^* of $\widehat{T}_{DR}$ and $\widehat{C}_{T_1} \circ \widehat{C}_{T_2}$, and the sequence $(w^k)_{k\geq 0}$ converges to $w^* = \widehat{R}_{T_2}(z^*) \in \operatorname{zer}(T_1 + T_2)$. This completes the proof. ■

Notice that similarly to the standard Douglas-Rachford splitting, each iteration requires evalutaing $\widehat{R}_{T_1}$ and $\widehat{R}_{T_2}$ each once, and their roles can be switched in (4.5).

4.3 Generalized Davis-Yin Splitting

Proposition 4.3.1 *Suppose $T_1,\ T_2$ are maximally monotone operators on $\mathbf{R}^n$; T_3 is β-cocoercive; $\operatorname{zer}(T_1 + T_2 + T_3) \neq \emptyset$. Define operator $\widehat{T}_{DY} := Id - \widehat{R}_{T_2} + \widehat{R}_{T_1} \circ (2\widehat{R}_{T_2} - Id - P^{-1}T_3 \circ \widehat{R}_{T_2})$ and suppose $\lambda_{min}(P) > \frac{1}{2\beta}$. Then T_{DY} is θ-averaged with respect to the norm $\|\cdot\|_P$ with coefficient $\theta = \frac{2\beta\lambda_{min}(P)}{4\beta\lambda_{min}(P)-1} < 1$. Starting from any z^0, for $k = 0, 1, \ldots$, let*

$$w^{k+1} = \widehat{R}_{T_2}(z^k); \tag{4.6a}$$

$$z^{k+1} = z^k - w^{k+1} + \widehat{R}_{T_1}\left(2w^{k+1} - z^k - P^{-1}T_3(w^{k+1})\right). \tag{4.6b}$$

The sequence $(z^k)_{k\geq 0}$ converges to some $z^* \in \operatorname{Fix}(\widehat{T}_{DY})$, and the sequence $(w^k)_{k\geq 0}$ converges to $\widehat{R}_{T_2}(z^*) \in \operatorname{zer}(T_1 + T_2 + T_3)$.

Proof We first show that

$$\widehat{R}_{T_2}\big(\operatorname{Fix}(\widehat{T}_{DY})\big) = \operatorname{zer}(T_1 + T_2 + T_3).$$

Let w^* be a arbitrary point in zer$(T_1 + T_2 + T_3)$. Then

$$\begin{aligned}
& 0 \in (T_1 + T_2 + T_3)w^* \\
\iff\quad & 0 \in (P^{-1}T_1 + P^{-1}T_2 + P^{-1}T_3)w^* \\
\iff\quad & 0 \in (\mathrm{Id} + P^{-1}T_1)w^* - (\mathrm{Id} - P^{-1}T_2)w^* + P^{-1}T_3(w^*) \\
\iff\quad & 0 \in (\mathrm{Id} + P^{-1}T_1)w^* - \widehat{C}_{T_2} \circ (\mathrm{Id} + P^{-1}T_2)w^* + P^{-1}T_3(w^*) \\
\iff\quad & 0 \in (\mathrm{Id} + P^{-1}T_1)w^* - \widehat{C}_{T_2}(z^*) + P^{-1}T_3(w^*), \quad \text{for some } z^* \in (\mathrm{Id} + P^{-1}T_2)w^* \\
\iff\quad & (\widehat{C}_{T_2} - P^{-1}T_3 \circ \widehat{R}_{T_2})z^* \in (\mathrm{Id} + P^{-1}T_1) \circ \widehat{R}_{T_2}(z^*), \quad w^* = \widehat{R}_{T_2}(z^*) \\
\iff\quad & 2\widehat{R}_{T_1} \circ (\widehat{C}_{T_2} - P^{-1}T_3 \circ \widehat{R}_{T_2})z^* = 2\widehat{R}_{T_2}(z^*), \quad w^* = \widehat{R}_{T_2}(z^*)
\end{aligned}$$

$$\iff \quad (\widehat{C}_{T_1} + \mathrm{Id}) \circ (\widehat{C}_{T_2} - P^{-1}T_3 \circ \widehat{R}_{T_2})z^* = (\widehat{C}_{T_2} + \mathrm{Id})z^*, \quad w^* = \widehat{R}_{T_2}(z^*)$$
$$\iff \quad \Big(\widehat{C}_{T_1} \circ (\widehat{C}_{T_2} - P^{-1}T_3 \circ \widehat{R}_{T_2}) - P^{-1}T_3 \circ \widehat{R}_{T_2}\Big) z^* = z^*, \quad w^* = \widehat{R}_{T_2}(z^*).$$

We have

$$\begin{aligned}
&\frac{1}{2}\mathrm{Id} + \frac{1}{2}\Big(\widehat{C}_{T_1} \circ (\widehat{C}_{T_2} - P^{-1}T_3 \circ \widehat{R}_{T_2}) - P^{-1}T_3 \circ \widehat{R}_{T_2}\Big) \\
=&\frac{1}{2}\mathrm{Id} + \frac{1}{2}\Big((2\widehat{R}_{T_1} - \mathrm{Id}) \circ \big((2\widehat{R}_{T_2} - \mathrm{Id}) - P^{-1}T_3 \circ \widehat{R}_{T_2}\big) - P^{-1}T_3 \circ \widehat{R}_{T_2}\Big) \\
=&\frac{1}{2}\mathrm{Id} + \frac{1}{2}\Big(4\widehat{R}_{T_1} \circ \widehat{R}_{T_2} - 2\widehat{R}_{T_1} - 2\widehat{R}_{T_2} + \mathrm{Id} - 2\widehat{R}_{T_1}P^{-1}T_3 \circ \widehat{R}_{T_2}\Big) \\
=&2\widehat{R}_{T_1} \circ \widehat{R}_{T_2} - \widehat{R}_{T_1} - \widehat{R}_{T_2} + \mathrm{Id} - \widehat{R}_{T_1}P^{-1}T_3 \circ \widehat{R}_{T_2} \\
=&\mathrm{Id} - \widehat{R}_{T_2} + \widehat{R}_{T_1} \circ (2\widehat{R}_{T_2} - \mathrm{Id} - P^{-1}T_3 \circ \widehat{R}_{T_2}) \\
=&\widehat{T}_{DY}.
\end{aligned}$$

Therefore, $\widehat{T}_{DY}$ has the same fixed point set as $\widehat{C}_{T_1} \circ (\widehat{C}_{T_2} - P^{-1}T_3 \circ \widehat{R}_{T_2}) - P^{-1}T_3 \circ \widehat{R}_{T_2}$, and we have $\widehat{R}_{T_2}\big(\mathrm{Fix}(\widehat{T}_{DY})\big) = \mathrm{zer}(T_1 + T_2 + T_3)$.

Next, we show that $\widehat{T}_{DY}$ is θ-averaged with respect to the norm $\|\cdot\|_P$ with coefficient $\theta = \frac{2\beta\lambda_{\min}(P)}{4\beta\lambda_{\min}(P)-1}$. Let $T_4 = \mathrm{Id} - \widehat{R}_{T_2}$, $T_5 = 2\widehat{R}_{T_2} - \mathrm{Id} - P^{-1}T_3 \circ \widehat{R}_{T_2}$. Since $\widehat{R}_{T_2}$ is firmly nonexpansive with respect to the norm $\|\cdot\|_P$, T_4 is also firmly nonexpansive. For any x, y, we have

$$\begin{aligned}
\|\widehat{T}_{DY}(x) - \widehat{T}_{DY}(y)\|_P^2 &= \|T_4(x) - T_4(y)\|_P^2 + \|\widehat{R}_{T_1} \circ T_5(x) - \widehat{R}_{T_1} \circ T_5(y)\|_P^2 \\
&\qquad + 2\langle \widehat{R}_{T_1} \circ T_5(x) - \widehat{R}_{T_1} \circ T_5(y), T_4(x) - T_4(y)\rangle_P \\
&\le \langle T_4(x) - T_4(y), x - y\rangle_P + \langle \widehat{R}_{T_1} \circ T_5(x) - \widehat{R}_{T_1} \circ T_5(y), T_5(x) - T_5(y)\rangle_P \\
&\qquad + 2\langle \widehat{R}_{T_1} \circ T_5(x) - \widehat{R}_{T_1} \circ T_5(y), T_4(x) - T_4(y)\rangle_P \\
&= \langle T_4(x) - T_4(y), x - y\rangle_P \\
&\qquad + \langle \widehat{R}_{T_1} \circ T_5(x) - \widehat{R}_{T_1} \circ T_5(y), (2T_4 + T_5)x - (2T_4 + T_5)y\rangle_P \\
&= \langle \widehat{T}_{DY}(x) - \widehat{T}_{DY}(y), x - y\rangle_P \\
&\qquad + \langle \widehat{R}_{T_1} \circ T_5(x) - \widehat{R}_{T_1} \circ T_5(y), (2T_4 + T_5 - \mathrm{Id})x - (2T_4 + T_5 - \mathrm{Id})y\rangle_P \\
&= \langle \widehat{T}_{DY}(x) - \widehat{T}_{DY}(y), x - y\rangle_P \\
&\qquad - \langle \widehat{R}_{T_1} \circ T_5(x) - \widehat{R}_{T_1} \circ T_5(y), P^{-1}T_3 \circ \widehat{R}_{T_2}(x) - P^{-1}T_3 \circ \widehat{R}_{T_2}(y)\rangle_P,
\end{aligned}$$

where the inequality is because of the firm nonexpansiveness of operators T_4 and $\widehat{R}_{T_1}$. Since the following equality always holds

$$\langle \widehat{T}_{DY}(x) - \widehat{T}_{DY}(y), x - y\rangle_P = \frac{1}{2}\|x-y\|_P^2 + \frac{1}{2}\|\widehat{T}_{DY}(x) - \widehat{T}_{DY}(y)\|_P^2 - \frac{1}{2}\|(\mathrm{Id} - \widehat{T}_{DY})x - (\mathrm{Id} - \widehat{T}_{DY})y\|_P^2,$$

plugging it in the previous equality and rearranging items, we have

$$\begin{aligned}
\|\widehat{T}_{DY}(x) - \widehat{T}_{DY}(y)\|_P^2 &\leq \|x-y\|_P^2 - \|(\mathrm{Id} - \widehat{T}_{DY})x - (\mathrm{Id} - \widehat{T}_{DY})y\|_P^2 \\
&\quad - 2\langle \widehat{R}_{T_1}\circ T_5(x) - \widehat{R}_{T_1}\circ T_5(y), P^{-1}T_3\circ \widehat{R}_{T_2}(x) - P^{-1}T_3\circ \widehat{R}_{T_2}(y)\rangle_P \\
&= \|x-y\|_P^2 - \|(\mathrm{Id} - \widehat{T}_{DY})x - (\mathrm{Id} - \widehat{T}_{DY})y\|_P^2 \\
&\quad + 2\langle (\mathrm{Id} - \widehat{T}_{DY})x - (\mathrm{Id} - \widehat{T}_{DY})y, T_3\circ \widehat{R}_{T_2}(x) - T_3\circ \widehat{R}_{T_2}(y)\rangle \\
&\quad - 2\langle \widehat{R}_{T_2}(x) - \widehat{R}_{T_2}(y), T_3\circ \widehat{R}_{T_2}(x) - T_3\circ \widehat{R}_{T_2}(y)\rangle \\
&\leq \|x-y\|_P^2 - \|(\mathrm{Id} - \widehat{T}_{DY})x - (\mathrm{Id} - \widehat{T}_{DY})y\|_P^2 \\
&\quad + \frac{1}{2\beta}\|(\mathrm{Id} - \widehat{T}_{DY})x - (\mathrm{Id} - \widehat{T}_{DY})y\|^2 + 2\beta\|T_3\circ \widehat{R}_{T_2}(x) - T_3\circ \widehat{R}_{T_2}(y)\|^2 \\
&\quad - 2\langle \widehat{R}_{T_2}(x) - \widehat{R}_{T_2}(y), T_3\circ \widehat{R}_{T_2}(x) - T_3\circ \widehat{R}_{T_2}(y)\rangle \\
&\leq \|x-y\|_P^2 - \|(\mathrm{Id} - \widehat{T}_{DY})x - (\mathrm{Id} - \widehat{T}_{DY})y\|_P^2 \\
&\quad + \frac{1}{2\beta}\|(\mathrm{Id} - \widehat{T}_{DY})x - (\mathrm{Id} - \widehat{T}_{DY})y\|^2 + 2\beta\|T_3\circ \widehat{R}_{T_2}(x) - T_3\circ \widehat{R}_{T_2}(y)\|^2 \\
&\quad - 2\beta\|T_3\circ \widehat{R}_{T_2}(x) - T_3\circ \widehat{R}_{T_2}(y)\|^2 \\
&\leq \|x-y\|_P^2 - \left(1 - \frac{1}{2\beta\lambda_{\min}(P)}\right)\|(\mathrm{Id} - \widehat{T}_{DY})x - (\mathrm{Id} - \widehat{T}_{DY})y\|_P^2 \\
&= \|x-y\|_P^2 - \frac{1-\theta}{\theta}\|(\mathrm{Id} - \widehat{T}_{DY})x - (\mathrm{Id} - \widehat{T}_{DY})y\|_P^2,
\end{aligned}$$

where $0 < \theta = \frac{2\beta\lambda_{\min}(P)}{4\beta\lambda_{\min}(P)-1} < 1$ since $\lambda_{\min}(P) > \frac{1}{2\beta}$. The second inequality is due to the β-cocoercity of operator T_3. Therefore, operator $\widehat{T}_{DY}$ is θ-averaged with respect to the norm $\|\cdot\|_P$. The sequence $(z^k)_{k\geq 0}$ generated by iteration (4.6) converges to a fixed point z^* of $\widehat{T}_{DY}$, and the sequence $(w^k)_{k\geq 0}$ converges to $w^* = \widehat{R}_{T_2}(z^*) \in \mathrm{zer}(T_1 + T_2 + T_3)$. This completes the proof. ■

Remark 4.3.1 *Notice that the standard Davis-Yin splitting in Proposition* (3.6.2) *is a special case of the above generalized Davis-Yin splitting with* $P = \frac{1}{\gamma}I_n$.

5. DISTRIBUTED SYNCHRONOUS ALGORITHMS WITH COORDINATOR

In this chapter, we will propose two distributed synchronous algorithms for the base standard formulation of our multi-agent optimization problem. Both algorithms are designed to find zero points of saddle subdifferential operators (to be defined later), which are also saddle points that represent the optimal primal-dual pairs of certain Lagrange functions associated with the multi-agent optimization problem.

The first algorithm applies the generalized resolvent iteration to the regular Lagrange function and its corresponding saddle subdifferential operator, and can be thought of as an extension of the famous alternating direction method of multipliers (ADMM) to the multi-block case with a parallel updating structure. The second algorithm is an application of the standard Douglas-Rachford splitting method to a different, but equivalent Lagrange function and its corresponding saddle subdifferential operator.

Although both algorithms require a central coordinator to collect certain variables from all agents and perform updates on some dual variable, the second algorithm has the advantage that no primal local decision variables need to be exchanged between the coordinator and agents, thus privacy of agents are better protected.

5.1 Primal-dual Precursor Algorithms

Recall our base standard multi-agent optimization problem has the following form,

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \sum_{i=1}^{L} f_i(x_i) \\ \text{subject to} \quad & Ax = b, \quad x_i \in X_i, \quad \forall i \in [L], \end{aligned}$$

where we have the shared coupling equality constraint and the private local constraint $x_i \in X_i$. In fact, since we assume that the local constraint sets X_i's are convex and

nonempty, they can be incorporated into the local objective function f_i by adding the convex indicator functions $\mathbf{1}_{X_i}(x_i)$'s.

Therefore, the base standard formulation is equivalent to

$$\underset{x}{\text{minimize}} \quad \sum_{i=1}^{L} \left(f_i(x_i) + \mathbf{1}_{X_i}(x_i)\right) \tag{5.1}$$

$$\text{subject to} \quad Ax = \sum_{i=1}^{L} A_i x_i = b. \tag{5.2}$$

One classic primal dual algorithm that has been used to solve problem (5.1) is dual decomposition [28]. The Lagrange function of (5.1) is

$$\begin{aligned} \mathcal{L}(x,\, \lambda) &= \sum_{i=1}^{L} \left(f_i(x_i) + \mathbf{1}_{X_i}(x_i)\right) + \lambda^T(Ax - b) \\ &= \sum_{i=1}^{L} \left(f_i(x_i) + \mathbf{1}_{X_i}(x_i) + \lambda^T A_i x_i\right) - \lambda^T b \\ &= \sum_{i=1}^{L} \mathcal{L}_i(x_i,\, \lambda) - \lambda^T b, \end{aligned} \tag{5.3}$$

where λ is the Lagrange multiplier, or dual variable. The dual function is $d(\lambda) = \min_x \mathcal{L}(x, \lambda) = \sum_{i=1}^{L} \min_{x_i} \mathcal{L}_i(x_i,\, \lambda) - \lambda^T b$. If we assume strong duality holds, then the original problem is equivalent to the dual problem

$$\underset{\lambda}{\text{maximize}} \quad \sum_{i=1}^{L} \left(f_i(x_i^*) + \mathbf{1}_{X_i}(x_i^*) + \lambda^T A_i x_i^*\right) - \lambda^T b, \tag{5.4}$$

where $x^* = (x_1^*, \ldots, x_L^*) = \arg\min_x \mathcal{L}(x, \lambda)$. One subgradient of λ for problem (5.4) is $Ax^* - b$. The dual decomposition method iterates between updating primal variable x and updating dual variable λ,

$$x^{k+1} = \arg\min_x \mathcal{L}(x, \lambda^k), \tag{5.5}$$

$$\lambda^{k+1} = \lambda^k + \alpha_k(Ax^{k+1} - b), \tag{5.6}$$

where the primal update step (5.5) is equivalent to solving L individual smaller problems:

$$x_i^{k+1} = \arg\min_{x_i} \mathcal{L}_i(x_i, \lambda), \quad \forall i \in [L]. \tag{5.7}$$

Obviously, (5.7) can be carried out in parallel.

Notice that dual decomposition is essentially the subgradient method [83] applied to the dual problem, and the evaluation of the subgradient of dual variable λ during each iteration involves solving an optimization problem (5.5). However, the convergence of the subgradient method requires a careful choice of the step size α_k and the satisfaction of some additional assumptions, such as the boundness of local constraint set X_i's so that the subgradient $\|Ax^{k+1} - b\|$ is bounded for all k. Another drawback of the dual decomposition/subgradient method is its relatively slow convergence speed at $O(1/\sqrt{k})$.

Another distributed algorithm that dates back to the 70s but has become popular again in the last decade is the ADMM method [36, 34]. The standard ADMM method is designed for a special case of problem (5.1) where $L = 2$, i.e., there are only two blocks of variables. It involves the augmented Lagrange function $\mathcal{L}_\rho(x,\, \lambda)$:

$$\begin{aligned}\mathcal{L}_\rho(x,\, \lambda) &= \mathcal{L}(x,\, \lambda) + \frac{\rho}{2}\|Ax - b\|^2 \\ &= \mathcal{L}_1(x_1,\, \lambda) + \mathcal{L}_2(x_2,\, \lambda) - \lambda^T b + \frac{\rho}{2}\|A_1x_1 + A_2x_2 - b\|^2.\end{aligned}$$

where $\rho > 0$ is the penalty parameter. In each iteration, we minimize the augmented Lagrange function over x_1 and x_2 separately, followed by a dual update step:

$$\begin{aligned}x_1^{k+1} &= \arg\min_{x_1} \mathcal{L}_\rho(x_1, x_2^k, \lambda^k), \\ x_2^{k+1} &= \arg\min_{x_2} \mathcal{L}_\rho(x_1^{k+1}, x_2, \lambda^k), \\ \lambda^{k+1} &= \lambda^k + \rho(Ax^{k+1} - b),\end{aligned}$$

Notice that the update of x_1 and x_2 must be performed in a sequential fashion instead of in parallel. The standard ADMM has good convergence behavior under relatively mild assumptions. Interestingly, it can be shown that the standard ADMM is a special case of the Douglas-Rachford splitting technique (3.3) applied to the dual problem (5.4), we refer readers to [33, 78] for more details.

If the standard ADMM is extended to cases where $L \geq 3$, the algorithm is called Gauss-Seidel ADMM, where one sequentially update x_i for $i = 1, \ldots, L$ as follows

$$x_i^{k+1} = \arg\min_{x_i} \mathcal{L}_\rho \left(x_1^{k+1}, \ldots, x_{i-1}^{k+1}, x_i, x_{i+1}^k, \ldots, x_L^k \right).$$

The Gauss-Seidel ADMM was proposed recently [40], but has been shown to be not necessarily convergent [19]. Some additional assumptions are needed for guaranteed convergence. For example, if we assume that local objective functions f_i are strongly convex, then the Gauss-Seidel ADMM method converges [38]. One big disadvantage of the Gauss-Seidel ADMM is that agents have to take turns to update their local decision variables in a serial fashion since they need the updated information from those indexed before them.

A parallel update scheme across agents such as in the case of dual decomposition is desirable since it would take better advantage of different parallel computing infrastructure. That is, all the agents perform local optimization simultaneously without having to wait for other's updated information as in the serial scheme. One straightforward modification of the serial ADMM towards this direction is the *parallel* ADMM, where at every iteration each agent will minimize the augmented Lagrange function with respect to its local decision variable x_i, assuming other parts of x fixed:

$$x_i^{k+1} = \arg\min_{x_i} \mathcal{L}_\rho \left(x_1^k, \ldots, x_{i-1}^k, x_i, x_{i+1}^k, \ldots, x_L^k \right). \tag{5.8}$$

However, even in the simplest two block setting ($L = 2$), this scheme does not converge in general [39].

It is desirable to have distributed algorithms that require minimal set of assumptions for convergence, such as the standard two block ADMM, and also have a parallel updating structure as in dual decomposition. We will propose two such algorithms in this chapter. Both algorithms are designed to find points that satisfy the KKT conditions of optimization problem (5.1). For constrained convex optimization problems, any points satisfying the KKT conditions are optimal solutions. To find such points, we start from a more general concept of saddle function. The Lagrange function (5.3) is a special class of saddle functions, and KKT points of the original optimization problem are saddle points of the

Lagrange function, which are also zero points of the saddle subdifferential operator that will defined next.

5.2 Saddle Function and Saddle Subdifferential Operator

Definition 5.2.1 ([76]) *$K : X \times Y \to \mathbf{R} \cup \{+\infty, -\infty\}$ is called a saddle function on $X \times Y$ if $K(x, y)$ is a convex function of x for each fixed y and a concave function of y for each fixed x. The saddle function K is closed if $K(x, y)$ is lower semicontinuous in x for each fixed y and upper semicontinuous in y for each fixed x. The effective domain of K is defined as*

$$\operatorname{dom} K := \{(x, y) \in X \times Y \mid K(x, y') < +\infty,\ \forall y' \in Y \text{ and } K(x', y) > -\infty,\ \forall x' \in X\}.$$

K is called proper if $\operatorname{dom} K \neq \emptyset$.

A point $(x^*, y^*) \in X \times Y$ is called a *saddle point* of the saddle function K if

$$K(x^*, y) \leq K(x^*, y^*) \leq K(x, y^*), \quad \forall x \in X,\ y \in Y. \tag{5.9}$$

In other words, x^* is a minimizer of $K(\cdot, y^*)$ and y^* is a maximizer of $K(x^*, \cdot)$. In this case, it holds that $\sup_y \inf_x K(x, y) = \inf_x \sup_y K(x, y) = K(x^*, y^*)$.

General saddle functions may have no saddle points. A sufficient condition for the existence of saddle points based on the Sion's Minimax Theorem [85] is given below.

Proposition 5.2.1 ([85]) *Suppose K is a real-valued closed saddle function on $X \times Y$. Let $C \subset X$ and $D \subset Y$ be two nonempty compact convex subsets. Then*

$$\min_{x \in C} \max_{y \in D} K(x, y) = \max_{y \in D} \min_{x \in C} K(x, y).$$

For the saddle function K on $X \times Y$, a set-valued operator T_K can be defined by

$$T_K(x, y) = \begin{bmatrix} \partial_x K(x, y) \\ \partial_y (-K)(x, y) \end{bmatrix}, \quad \forall (x, y) \in X \times Y.$$

In [78], T_K is referred to as the saddle subdifferential operator of K. The condition (5.9) for a point (x^*, y^*) to be a saddle point of K is equivalent to $0 \in \partial_x K(x^*, y^*)$ and $0 \in \partial_y(-K)(x^*, y^*)$, i.e., $0 \in T_K(x^*, y^*)$. We thus have the following result.

Proposition 5.2.2 *The set of saddle points of K is* $\operatorname{zer}(T_K)$.

Theorem 5.2.1 ([76]) *Let K be a saddle function on $X \times Y$. If K is proper, then T_K is a monotone operator with the domain* $\operatorname{dom} T_K \subset \operatorname{dom} K$. *If K is proper and closed, then T_K is a maximally monotone operator.*

By the above result, for a closed proper saddle function K, its saddle subdifferential T_K is maximally monotone, thus the corresponding resolvent, denoted by R_K, is an averaged operator with the fixed point set $\operatorname{Fix}(R_K) = \operatorname{zer}(T_K)$ being exactly the set of saddle points of K. The iteration $(x^{k+1}, y^{k+1}) = R_K(x^k, y^k)$ will converge to a point in $\operatorname{Fix}(R_K)$ and hence a saddle point of K. We next characterize how R_K can be computed. For any $(x, y) \in X \times Y$, $(p, q) = R_K(x, y)$ if and only if

$$
\begin{aligned}
& (x, y) \in (\mathrm{Id} + \beta T_K)(p, q) \\
\iff \quad & (x, y) \in (p + \beta \partial_p K(p, q), q + \beta \partial_q(-K)(p, q)) \\
\iff \quad & \begin{cases} 0 \in \partial_p K(p, q) + (p - x)/\beta \\ 0 \in \partial_q(-K)(p, q) + (q - y)/\beta \end{cases} && \text{(5.10a)} \\
\iff \quad & \begin{cases} p = \arg\min_{p \in X} \ K(p, q) + \frac{1}{2\beta}\|p - x\|^2 \\ q = \arg\max_{q \in Y} \ K(p, q) - \frac{1}{2\beta}\|q - y\|^2 \end{cases} && \text{(5.10b)} \\
\iff \quad & (p, q) \text{ is a saddle point of } K(p, q) + \frac{1}{2\beta}\left(\|p - x\|^2 - \|q - y\|^2\right). && \text{(5.10c)}
\end{aligned}
$$

By letting $p = x$ and $q = y$ in (5.10c), we see that, as expected, fixed points of R_K are exactly the saddle points of K.

For many closed proper saddle functions, the computational cost involved with evaluating R_K directly as in (5.10b) might be high. In order to find saddle points of those saddle functions, there are three main strategies to reduce the computational cost or make the computation have a nicer structure, such as parallel structure. Firstly, if the saddle function has a cartesian structure and is separable, we can easily prove the following proposition.

Proposition 5.2.3 *Suppose $K(x, y) = K_1(x_1, y_1) + \cdots + K_L(x_L, y_L)$ is separable, where $K_i(x_i, y_i)$ is a closed proper saddle function on $X_i \times Y_i$; $x = (x_1, \ldots, x_L) \in X =$*

$X_1 \times \cdots \times X_L$; *and* $y = (y_1, \ldots, y_L) \in Y = Y_1 \times \cdots \times Y_L$. *Then,* $(p, q) = R_K(x, y)$ *is given by* $p = (p_1, \ldots, p_L)$ *and* $q = (q_1, \ldots, q_L)$ *where* $(p_i, q_i) = R_{K_i}(x_i, y_i)$ *for each* i.

Secondly, instead of using the standard resolvent R_K, generalized resolvent $\widehat{R}_K$ as defined in (4.1) with some carefully designed positive definite matrix P can be utilized. Thirdly, in some cases, a saddle function can be written as the summation of two or three closed proper saddle functions, for which the resolvent or generalized resolvent operators are much easier to compute. Then, by utilizing the tools from operator splitting or generalized operator splitting methods, we can derive iterative algorithms to efficiently compute saddle points of the original saddle function.

5.3 Proximal Parallel ADMM

Notice that the Lagrange function $\mathcal{L}(x,\ \lambda)$ in (5.3) is a saddle function on $X \times \mathbf{R}^m$ where $X = \prod_{i=1}^{L} X_i$ because $\mathcal{L}$ is a convex function of x for each fixed λ and concave function of λ for each fixed x. The saddle subdifferential operator of a Lagrange function for a constrained convex optimization problem is also called the KKT operator as we discussed in Chapter 3. The saddle subdifferential operator $T_{\mathcal{L}}(x, \lambda)$ with $\mathrm{dom}(T_{\mathcal{L}}) \neq \emptyset$ and $\mathrm{dom}(T_{\mathcal{L}}) \subseteq \mathbf{R}^{n+m}$ of the Lagrange function $\mathcal{L}(x,\ \lambda)$ in (5.3) can be written as:

$$T_{\mathcal{L}}(x, \lambda) = \begin{bmatrix} \partial_x \mathcal{L}(x, \lambda) \\ \partial_\lambda(-\mathcal{L})(x, \lambda) \end{bmatrix} = \begin{bmatrix} F(x) + N_X(x) + A^T\lambda \\ -Ax + b \end{bmatrix}, \tag{5.11}$$

where $F(x) := (\partial_{x_1} f_1(x_1), \ldots, \partial_{x_L} f_L(x_L)) \in \mathbf{R}^n$, and $N_X(x)$ is the normal cone operator of set X.

Proposition 5.3.1 *The set of optimal primal-dual pairs for optimization problem* (5.1) *is* $\mathrm{zer}(T_{\mathcal{L}})$.

Proposition 5.3.2 *Operator* $T_{\mathcal{L}}(x, \lambda)$ *as defined in* (5.11) *is maximally monotone.*

Proof Note that the indicator function is lower semicontiuous. Therefore, $\mathcal{L}(x,\ \lambda)$ is closed since it is lower semicontinuous in x for each fixed λ and upper semicontinuous in

λ for each fixed x; it is also proper because its effective domain is nonempty. Therefore, $T_{\mathcal{L}}(x, \lambda)$ is a maximally monotone operator according to Theorem 5.2.1. ■

Remark 5.3.1 *Another way to prove $T_{\mathcal{L}}(x, \lambda)$ is maximally monotone is by observing that*

$$T_{\mathcal{L}}(x, \lambda) = T_1(x, \lambda) + T_2(x, \lambda) = \begin{bmatrix} F(x) + N_X(x) \\ b \end{bmatrix} + \begin{bmatrix} 0 & A^T \\ -A^T & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix}.$$

$T_1(x, \lambda)$ is maximally monotone since individually, $F(x)$, $N_X(x)$, and b are all maximally monotone operators in their respective domains, and both summation and concatenation operations preserve maximal monotonicity. $T_2(x, \lambda)$ is maximally monotone since it is an affine mapping with a skew-symmetric matrix. Hence, $T_{\mathcal{L}}(x, \lambda)$ is maximally monotone.

We propose the following *proximal parallel* ADMM method, which builds on the parallel ADMM method, and has a parallel update structure across agents. The proximal parallel ADMM algorithm is summarized in Algorithm 1. At each iteration, local agents solve the local optimization problems in parallel according to (5.12), followed by an update on the dual variable (5.13). Notice that the dual update requires the most up to date information from all the agents, thus needs to be carried out by a central coordinator that can communicate with all agents.

The proximal parallel ADMM algorithm will next be characterized as a special case of the generalized resolvent iteration (4.4) applied to the operator $T_{\mathcal{L}}$ with a carefully designed symmetric positive definite matrix P. Therefore, its convergence to some optimal solution starting from any initial conditions is guaranteed. Notice that the local update of each agent (5.12) can also be written as

$$x_i^{k+1} = \arg\min_{x_i} \mathcal{L}_\rho \left(x_1^k, \ldots, x_{i-1}^k, x_i, x_{i+1}^k, \ldots, x_L^k\right) + \frac{\varphi_i}{2}\|x_i - x_i^k\|^2.$$

Compared with the parallel ADMM method (5.8), a proximal term $\frac{\varphi_i}{2}\|x_i - x_i^k\|^2$ is added to regularize each agent's subproblem for some $\varphi_i > 0$. The motivation and intuition behind adding this proximal term is that it adds certain levels of "inertia" to the local updates of all agents, hence agents become less aggressive when moving to a new point. These "controlled" updates help prevent the divergence of the algorithm.

Algorithm 1 Proximal Parallel ADMM

1: Initialize $(x^0,\ \lambda^0)$, set $k = 0$;

2: **while** stopping criterion is not satisfied **do**

3: Coordinator sends λ^k and $c_i^k = \sum_{j\neq i} A_j x_j^k - b$ to each agent i;

4: Each agent i updates x_i (in parallel) according to

$$x_i^{k+1} = \arg\min_{x_i \in X_i} \left(f_i(x_i) + (\lambda^k)^T A_i x_i + \frac{\rho}{2}\|A_i x_i + c_i^k\|^2 + \frac{\varphi_i}{2}\|x_i - x_i^k\|^2 \right); \quad (5.12)$$

5: Each agent i sends x_i^{k+1} to Coordinator;

6: Coordinator updates λ according to

$$\lambda^{k+1} = \lambda^k + \rho(Ax^{k+1} - b). \quad (5.13)$$

7: $k \leftarrow k + 1$;

8: **end while**

Define matrix P as follows,

$$P = \begin{bmatrix} P_z - \rho A^T A & 0 \\ 0 & \frac{1}{\rho} I_m \end{bmatrix}, \quad (5.14)$$

where $P_z = \text{diag}(P_1, P_2, \ldots, P_L)$, and $P_i = \varphi_i I_{n_i} + \rho A_i^T A_i$. P_z is symmetric since each of P_i is symmetric. Therefore, matrix P is symmetric since both of its diagonal blocks are symmetric.

Definition 5.3.1 ([68]) *A matrix $B \in \mathbf{R}^{n\times n}$ is an M-matrix if 1) any off diagonal entry of B is non-positive; 2) B is invertible and any entry of B^{-1} is non-negative.*

It is easy to see that any diagonal matrix with positive diagonal entries is an M-matrix.

Definition 5.3.2 *Suppose $B \in \mathbf{R}^{n\times n}$ is a block matrix partitioned as*

$$B = \begin{bmatrix} B_{11} & B_{12} & \cdots & B_{1m} \\ B_{21} & B_{22} & \cdots & B_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ B_{m1} & B_{m2} & \cdots & B_{mm} \end{bmatrix},$$

where $B_{ii} \in \mathbf{R}^{n_i \times n_i}$ *and* $\sum_{i=1}^{m} n_i = n$*. Then,* B *is called block diagonally dominant if*

$$\|B_{ii}^{-1}\|^{-1} \geq \sum_{j \neq i} \|B_{ij}\|, \quad \forall i. \tag{5.15}$$

If strict inequality holds in (5.15)*,* B *is called block strictly diagonally dominant.*

Note that the induced matrix norm is defined as $\|B\| := \sup_{x \neq 0} \frac{\|Bx\|}{\|x\|}$.

Lemma 5.3.1 ([30], Theorem 9) *Let* $B \in \mathbf{R}^{n \times n}$ *be block strictly diagonally dominant with all the diagonal blocks being* M*-matrices. Then, any eigenvalue of* B *has positive real part.*

Proposition 5.3.3 *Suppose the parameters* ρ *and* φ_i*,* $\forall i \in [L]$ *satisfy that* $\rho > 0$*,* $\varphi_i > \rho \sum_{j \in [L] \setminus \{i\}} \|A_i^T A_j\|$*,* $\forall i \in [L]$*. Then the matrix* P *defined in* (5.14) *is positive definite.*

Proof Since $\rho > 0$, $\frac{1}{\rho} I_m \succ 0$, it is sufficient to show that $\widehat{P} = P_z - \rho A^T A$ is positive definite. Writing out $\widehat{P}$ gives

$$\widehat{P} = \begin{bmatrix} \varphi_1 I_{n_1} & -\rho A_1^T A_2 & \cdots & -\rho A_1^T A_L \\ -\rho A_2^T A_1 & \varphi_2 I_{n_2} & \cdots & -\rho A_2^T A_L \\ \vdots & \vdots & \vdots & \vdots \\ -\rho A_L^T A_1 & \cdots & -\rho A_L^T A_{L-1} & \varphi_L I_{n_L} \end{bmatrix}. \tag{5.16}$$

Notice that $\widehat{P}$ is a block matrix with diagonal blocks being positive definite diagonal matrices, or, M-matrices. The second condition says that

$$\varphi_i = \left\| \frac{1}{\varphi_i} I \right\|^{-1} > \rho \sum_{j \in [L] \setminus \{i\}} \|A_i^T A_j\|,$$

which means matrix $\widehat{P}$ is block strictly diagonally dominant. Since $\widehat{P}$ is also symmetric, applying Lemma 5.3.1 yields that $\widehat{P} \succ 0$. Therefore, $P \succ 0$. ■

Proposition 5.3.4 *The proximal parallel ADMM algorithm, outlined in Algorithm 1, is the generalized resolvent iteration* (4.4) *applied to* $T_{\mathcal{L}}(x, \lambda)$*, with the matrix* P *defined in* (5.14)*.*

Proof Let $z^k = (x^k, \lambda^k) = (x_1^k, \ldots, x_L^k, \lambda^k)$. From the iteration (4.4), we have

$$\begin{aligned} & (\mathrm{Id} + P^{-1}T_{\mathcal{L}})z^{k+1} \ni z^k, \\ \iff \quad & Pz^{k+1} + T_{\mathcal{L}}(z^{k+1}) \ni Pz^k. \end{aligned} \tag{5.17}$$

Substituting (5.14) into (5.17) gives the following

$$\partial f_i(x_i^{k+1}) + N_{X_i}(x_i^{k+1}) + P_i(x_i^{k+1} - x_i^k) - \rho A_i^T A(x^{k+1} - x^k) + A_i^T \lambda^{k+1} \ni 0, \tag{5.18}$$

$$\lambda^{k+1} = \lambda^k + \rho(Ax^{k+1} - b). \tag{5.19}$$

Notice that (5.19) is exactly the dual update (5.13) in Algorithm 1. Substituting (5.19) into (5.18) yields

$$\begin{aligned} & \partial f_i(x_i^{k+1}) + N_{X_i}(x_i^{k+1}) + A_i^T \lambda^k + \rho A_i^T(Ax^k - b) + P_i(x_i^{k+1} - x_i^k) \ni 0, \\ \iff \quad & \partial f_i(x_i^{k+1}) + N_{X_i}(x_i^{k+1}) + A_i^T \lambda^k + \rho A_i^T(A_i x_i^{k+1} + \sum_{j \neq i} A_j x_j^k - b) + \\ & \qquad \varphi_i(x_i^{k+1} - x_i^k) \ni 0, \\ \iff \quad & \partial f_i(x_i^{k+1}) + N_{X_i}(x_i^{k+1}) + A_i^T \lambda^k + \rho A_i^T(A_i x_i^{k+1} + c_i^k) + \varphi_i(x_i^{k+1} - x_i^k) \ni 0. \end{aligned} \tag{5.20}$$

Noe that (5.20) corresponds to the first order optimality condition of the following minimization problem

$$\begin{aligned} x_i^{k+1} &= \arg\min_{x_i} \left(f_i(x_i) + \mathbf{1}_{X_i}(x_i) + (\lambda^k)^T A_i x_i + \frac{\rho}{2}\|A_i x_i + c_i^k\|^2 + \frac{\varphi_i}{2}\|x_i - x_i^k\|^2 \right), \\ &= \arg\min_{x_i \in X_i} \left(f_i(x_i) + (\lambda^k)^T A_i x_i + \frac{\rho}{2}\|A_i x_i + c_i^k\|^2 + \frac{\varphi_i}{2}\|x_i - x_i^k\|^2 \right), \end{aligned} \tag{5.21}$$

which is exactly the primal update (5.12) in Algorithm 1. Combining (5.19) and (5.21), we obtain the desired conclusion. ■

Theorem 5.3.1 (Convergence of Proximal Parallel ADMM) *Under Assumptions 2.1.1, 2.1.2 and 2.1.3, and suppose ρ and φ_i, $\forall i \in [L]$ are chosen according to Proposition 5.3.3 such that matrix P in* (5.14) *is positive definite. Then, the sequence $(x^k, \lambda^k)_{k \geq 0}$ generated by Algorithm 1 converges to $(x^*, \lambda^*) \in \mathrm{zer}(T_{\mathcal{L}})$ with $T_{\mathcal{L}}$ being defined in* (5.11)*, and x^* is an optimal solution to the original problem* (5.1)*.*

Proof Combining results from Proposition 4.4, Proposition 5.2.2, Proposition 5.3.1, Proposition 5.3.2, Proposition 5.3.3 and Proposition 5.3.4 yields the desired statement. ■

Remark 5.3.2 *One observation regarding proximal parameter φ_i is that suppose $A_i^T A_j = 0$, $\exists\, i \in [L]$ and $\forall j \neq i$, then φ can be chosen arbitrarity small according to the second condition of Proposition 5.3.3. This means that for a particular agent i, the stronger orthogonality its coupling matrix A_i has with those of other agents, the smaller weight needs be placed on the proximal term, thus the more aggresive agent i can be when updating local decision variables.*

The proposed proximal parallel ADMM has been applied to distributed model predictive control of multiple thermal zone buildings for minimizing the electricity bill of building's air conditioning systems while maintaining the thermal comfort of occupants' in different thermal zones [45, 44, 46].

One drawback of the proposed proximal parallel ADMM algorithm is that at each iteration, the central coordinator needs to collect the most recent optimal local decision variables x_i^k from all agents, update the dual variable according to (5.13) and calculate c_i^k, then send them back to the corresponding agents. The exchange of local decision variables x_i^k between agents and coordinator is not ideal as x_i^k usually contains important private information regarding agent i, which agent i may not want to share with others. Suppose an adversary/eavesdropper intercepts the communication between agents and the central coordinator, it could be detrimental to the securities of local agents. In addition, variable c_i^k actually contains information from other agents $j \neq i$, this may also raise private concerns.

To avoid this issue, we propose a new distributed synchronous algorithm in the next section, in which each agent will keep a local copy of the dual variable λ. The local updates of each agent include updates on both local decision variable x_i and local dual variable copy λ_i. At each iteration, agents will only send information regarding dual variables to the central coordinator and receive certain feedback. With only dual variable information, it is much more difficult for an adversary/eavesdropper to decipher private local decision

variable information. Also, local decision variables will no longer be shared between different agents.

5.4 Dual Averaging via Douglas-Rachford Splitting

First, we create a local copy of the dual variable λ for each agent, and deonote the augmented dual variable as $\lambda_a := (\lambda_1, \ldots, \lambda_L) \in \mathbf{R}^{mL}$. The new Lagrange function is

$$\begin{aligned}\mathcal{L}_a(x,\, \lambda_a) &= \sum_{i=1}^{L} \left(f_i(x_i) + \lambda_i^T A_i x_i - \lambda_i^T b_i\right) + \mu_{X\times\mathcal{A}_\lambda}(x, \lambda_a) \\ &= \sum_{i=1}^{L} \left(f_i(x_i)\right) + \lambda_a^T \Phi x - \lambda_a^T b_a + \mu_{X\times\mathcal{A}_\lambda}(x, \lambda_a),\end{aligned} \tag{5.22}$$

where $\mathcal{A}_\lambda = \{\lambda_a \,|\, \lambda_1 = \cdots = \lambda_L\}$, $\Phi = \text{diag}(A_1, \ldots, A_L) \in \mathbf{R}^{mL\times n}$, $b_a = (b_1, \ldots, b_L) \in \mathbf{R}^{mL}$, and

$$\mu_{X\times\mathcal{A}_\lambda}(x, \lambda_a) := \mathbf{1}_X(x) - \mathbf{1}_{(X\times\mathcal{A}_\lambda^c)^c}(x, \lambda_a) = \begin{cases} 0 & \text{if } x \in X \text{ and } \lambda_a \in \mathcal{A}_\lambda; \\ -\infty & \text{if } x \in X \text{ and } \lambda_a \notin \mathcal{A}_\lambda; \\ +\infty & \text{if } x \notin X. \end{cases}$$

The saddle subdifferential operator $T_{\mathcal{L}_a}(x, \lambda_a)$ with $\text{dom}(T_{\mathcal{L}_a}) \neq \emptyset$ and $\text{dom}(T_{\mathcal{L}_a}) \subseteq \mathbf{R}^{n+mL}$ of the Lagrange function $\mathcal{L}_a(x,\, \lambda_a)$ in (5.22) can be written as:

$$T_{\mathcal{L}_a}(x, \lambda_a) = \begin{bmatrix} \partial_x \mathcal{L}_a(x, \lambda_a) \\ \partial_{\lambda_a}(-\mathcal{L}_a)(x, \lambda_a) \end{bmatrix} = \begin{bmatrix} F(x) + N_X(x) + \Phi^T \lambda_a \\ -\Phi x + b_a + N_{\mathcal{A}_\lambda}(\lambda_a) \end{bmatrix}, \tag{5.23}$$

where $b_a = (b_1, \ldots, b_L) \in \mathbf{R}^{mL}$, $\Phi = \text{diag}(A_1, \ldots, A_L) \in \mathbf{R}^{mL\times n}$.

Proposition 5.4.1 *For any $(x^*, \lambda_a^*) \in \text{zer}(T_{\mathcal{L}_a})$ with $T_{\mathcal{L}_a}$ being defined in* (5.23)*, we have $\lambda_a^* \in \{1_L \otimes \lambda^* \,|\, \lambda^* \in \mathbf{R}^m\}$, i.e., $\lambda_i^* = \lambda_j^* = \lambda^*, \forall i, j \in [L]$. Furthermore, $(x^*, \lambda^*) \in \text{zer}(T_{\mathcal{L}})$ with $T_{\mathcal{L}}$ being defined in* (5.11)*, which means (x^*, λ^*) is an optimal primal-dual pair of the optimization problem* (5.1)*.*

Proposition 5.4.2 *Operator $T_{\mathcal{L}_a}(x, \lambda_a)$ as defined in* (5.23) *is maximally monotone.*

Proof Observing that

$$T_{\mathcal{L}_a}(x,\lambda_a) = \underbrace{\begin{bmatrix} 0 & \Phi^T \\ -\Phi & 0 \end{bmatrix}\begin{bmatrix} x \\ \lambda_a \end{bmatrix} + \begin{bmatrix} F(x) \\ b_a \end{bmatrix}}_{T_1(x,\lambda_a)} + \underbrace{\begin{bmatrix} N_X(x) \\ N_{\mathcal{A}_\lambda}(\lambda_a) \end{bmatrix}}_{T_2(x,\lambda_a)}.$$

The first part of $T_1(x,\lambda_a)$ is maximally monotone since it an affine mapping with skew-symmetric matrix, the second part of $T_1(x,\lambda_a)$ is maximally monotone since individually, $F(x)$ and b_a are maximally monotone, and concatenation preserves maximally monotonicity. $T_2(x,\lambda_a)$ is maximally monotone since individually, $N_X(x)$, $N_{\mathcal{A}_\lambda}(\lambda_a)$ are all maximally monotone operators in their respective domains (in particular, $N_{\mathcal{A}_\lambda}(\lambda_a)$ is maximally monotone because $\mathcal{A}_\lambda$ is a closed convex set). ■

To find a saddle point of $T_{\mathcal{L}_a}(x,\lambda_a)$, instead of evaluating the standard resolvent $R_{\mathcal{L}_a}$ or the generalized resolvent $\widehat{R}_{\mathcal{L}_a}$ directly, we first notice that $\mathcal{L}_a(x,\,\lambda_a)$ can be written as $\mathcal{L}(x,\,\lambda_a) = \mathcal{L}_1(x,\,\lambda_a) + \mathcal{L}_2(x,\,\lambda_a)$ where

$$\mathcal{L}_1(x,\,\lambda_a) = \sum_{i\in[L]} \left(f_i(x_i) + \lambda_i^T A_i x_i - \lambda_i^T b_i\right),$$

$$\mathcal{L}_2(x,\,\lambda_a) = \mu_{X\times\mathcal{A}_\lambda}(x,\lambda_a).$$

Note that $\mathcal{L}_1(x,\,\lambda_a)$ is closed since it is lower semicontinuous in x for each fixed λ_a and upper semicontinuous in λ_a for each fixed x; it is also proper because its effective domain is nonempty. In addition, $\mathcal{L}_2(x,\,\lambda_a)$ is also a closed and proper saddle function since X and $\mathcal{A}_\lambda$ are both nonempty closed convex sets. Thus $T_{\mathcal{L}_1}$ and $T_{\mathcal{L}_2}$ are also maximally monotone operators, whose resolvents $R_{\mathcal{L}_1}$ and $R_{\mathcal{L}_2}$ are averaged operators.

We adopt the Douglas-Rachford splitting method. Starting from any $z^0 = (x^0,\,\lambda_a^0)$, for $k = 0, 1, \ldots$, let

$$w^{k+1} = R_{\mathcal{L}_2}(z^k);$$

$$z^{k+1} = z^k + 2\alpha\left(R_{\mathcal{L}_1}(2w^{k+1} - z^k) - w^{k+1}\right),$$

where $\alpha \in (0,1)$; $R_{\mathcal{L}_1}$ and $R_{\mathcal{L}_2}$ are the resolvent operator corresponding to saddle functions $\mathcal{L}_1$ and $\mathcal{L}_2$, respectively. Next, we show how $R_{\mathcal{L}_1}$ and $R_{\mathcal{L}_2}$ can be computed in a distributed fashion by each agent.

Further notice that $\mathcal{L}_1(x,\,\lambda_a) = \sum_{i=1}^{L}\mathcal{L}_{1,i}(x_i,\lambda_i)$, where $\mathcal{L}_{1,i}(x_i,\lambda_i) = f_i(x_i) + \lambda_i^T A_i x_i - \lambda_i^T b_i$. From Proposition 5.2.3 and the fact that $\mathcal{L}_{1,i}(x_i,\lambda_i)$'s are closed proper saddle functions, we know $R_{\mathcal{L}_1}(x,\,\lambda_a)$ can be computed in a distributed fashion through each agent by evaluating $R_{\mathcal{L}_{1,i}}(x_i,\,\lambda_a)$ independently. For the computation of $(p_i, q_i) = R_{\mathcal{L}_{1,i}}(x_i,\,\lambda_i)$, notice that (5.10a) becomes

$$\begin{cases} 0 \in \partial f_i(p_i) + A_i^T q_i + (p_i - x_i)/\beta \\ A_i p_i - b_i = (q_i - \lambda_i)/\beta. \end{cases}$$

The second equation is equivalent to $q_i = \lambda_i + \beta(A_i p_i - b_i)$. Plugging into the first equation yields $0 \in \partial f_i(p_i) + (p_i - x_i)/\beta + A_i^T\lambda_i + \beta A_i^T(A_i p_i - b_i)$. Therefore, $(p_i, q_i) = R_{\mathcal{L}_{1,i}}(x_i,\lambda_i)$ can be evaluated as

$$\begin{cases} p_i &= \arg\min_s \left(f_i(s) + \lambda_i^T A_i s + \frac{\beta}{2}\|A_i s - b_i\|^2 + \frac{1}{2\beta}\|s - x_i\|^2 \right), \\ q_i &= \lambda_i + \beta(A_i p_i - b_i). \end{cases}$$

As for the computation of $R_{\mathcal{L}_2}$, we have $R_{\mathcal{L}_2}(x,\,\lambda_a) = (P_X(x), P_{\mathcal{A}_\lambda}(\lambda_a))$, where $P_X(x) = P_{X_1}(x_1) \times \cdots \times P_{X_L}(x_L)$, which can be computed by each agent independently. Here, P_C denotes the orthogonal projection operator onto C. In addition, $P_{\mathcal{A}_\lambda}(\lambda_a) = 1_L \otimes (\lambda_1 + \cdots + \lambda_L)/L$, which will be computed by a central coordinator. The proposed *dual averaging via Doulas-Rachford splitting* algorithm is summarized in Algorithm 2.

Theorem 5.4.1 *Under Assumptions 2.1.1, 2.1.2 and 2.1.3, the sequence* $(\overline{x}^k,\,\overline{\lambda}^k)_{k\geq 0}$ *generated by Algorithm 2 converges to some* $(x^*,\lambda^*) \in \mathrm{zer}(T_{\mathcal{L}})$*, and* x^* *is an optimal solution to the original problem* (5.1)*.*

Proof Combing results from Proposition 3.6.1, Proposition 5.4.1, Proposition 5.4.2 yields the desired statement. ■

Compared with the proximal parallel ADMM algorithm in Algorithm 1, this algorithm has the following advantages: 1) The information exchanged between the coordinator and agents solely contains copies of dual variables, and local optimization step (5.25c) does not directly involve primal decision variables from other agents, thus the privacies of individual

Algorithm 2 Dual Averaging via Douglas-Rachford Splitting

1: Initialize $(x^0,\, \lambda_a^0,\, \overline{\lambda}^0)$, set $k = 0$;

2: **while** stopping criterion is not satisfied **do**

3: Each agent i sends λ_i^k to Coordinator;

4: Coordinator calculates $\overline{\lambda} = (\lambda_1^k + \cdots + \lambda_L^k)/L$ and sends it back to each agent;

5: Each agent i updates local decision variables (in parallel) according to

$$\overline{x}_i^k = P_{X_i}(x_i^k), \quad \overline{\lambda}^k = \overline{\lambda}; \tag{5.25a}$$

$$\widehat{x}_i^k = 2\overline{x}_i^k - x_i^k, \quad \widehat{\lambda}_i^k = 2\overline{\lambda}^k - \lambda_i^k; \tag{5.25b}$$

$$\widehat{x}_i^{k+1} = \arg\min_{x_i} \Big(f_i(x_i) + (\widehat{\lambda}_i^k)^T A_i x_i + \frac{\beta}{2}\|A_i x_i - b_i\|^2 + \frac{1}{2\beta}\|x_i - \widehat{x}_i^k\|^2 \Big); \tag{5.25c}$$

$$\widehat{\lambda}_i^{k+1} = \widehat{\lambda}_i^k + \beta(A_i \widehat{x}_i^{k+1} - b_i); \tag{5.25d}$$

$$x_i^{k+1} = x_i^k + 2\alpha \left(\widehat{x}_i^{k+1} - \overline{x}_i^k\right); \tag{5.25e}$$

$$\lambda_i^{k+1} = \lambda_i^k + 2\alpha \left(\widehat{\lambda}_i^{k+1} - \overline{\lambda}_i^k\right); \tag{5.25f}$$

6: $k \leftarrow k + 1$;

7: **end while**

agents are better protected. 2) Although there are more local variables that each agent needs to keep track of, i.e., $\overline{x}_i^k$, $\widehat{x}_i^k$, $\overline{\lambda}_i^k$, $\widehat{\lambda}_i^k$, x_i^k, and λ_i^k, the local computation cost for each agent might be reduced. This is because in the proximal parallel ADMM algorithm, each agent needs to solve a constrained convex optimization problem as in (5.12), whereas here each agent only needs to complete a projection (5.25a), an unconstrained convex optimization problem (5.25c), and some other linear operations. If the local objective function is quadratic, (5.25c) reduces to linear operations.

Notice that in the Douglas-Rachford splitting method, the roles of the two resolvent operators can be switched, thus giving us a slightly different algorithm, which is summarized in Algorithm 3.

Algorithm 3 Dual Averaging via Douglas-Rachford Splitting, Second Version

1: Initialize $(x^0,\, \lambda_a^0)$, set $k = 0$;

2: **while** stopping criterion is not satisfied **do**

3: Each agent i updates local decision variables (in parallel) according to

$$\widehat{x}_i^k = \arg\min_{x_i} \left(f_i(x_i) + (\lambda_i^k)^T A_i x_i + \frac{\beta}{2}\|A_i x_i - b_i\|^2 + \frac{1}{2\beta}\|x_i - x_i^k\|^2 \right);$$
$$\widehat{\lambda}_i^k = \lambda_i^k + \beta(A_i \widehat{x}_i^k - b_i);$$
$$\widehat{x}_i^{k+1} = 2\widehat{x}_i^k - x_i^k, \quad \widehat{\lambda}_i^{k+1} = 2\widehat{\lambda}_i^k - \lambda_i^k;$$

4: Each agent i sends $\widehat{\lambda}_i^{k+1}$ to Coordinator;

5: Coordinator calculates $\overline{\lambda}^{k+1} = (\widehat{\lambda}_1^{k+1} + \cdots + \widehat{\lambda}_L^{k+1})/L$ and sends it back to each agent;

6: Each agent i updates local decision variables (in parallel) according to

$$\overline{x}_i^{k+1} = P_{X_i}(\widehat{x}_i^{k+1}), \quad \overline{\lambda}_i^{k+1} = \overline{\lambda}^{k+1};$$
$$x_i^{k+1} = x_i^k + 2\alpha\left(\overline{x}_i^{k+1} - \widehat{x}_i^k\right);$$
$$\lambda_i^{k+1} = \lambda_i^k + 2\alpha\left(\overline{\lambda}_i^{k+1} - \widehat{\lambda}_i^k\right);$$

7: $k \leftarrow k + 1$;

8: **end while**

Theorem 5.4.2 *Under Assumptions 2.1.1, 2.1.2 and 2.1.3, the sequence* $(\widehat{x}^k,\, \widehat{\lambda}^k)_{k\geq 0}$ *generated by Algorithm 3 converges to some* $(x^*, \lambda^*) \in \mathrm{zer}(T_{\mathcal{L}_a})$ *with* $T_{\mathcal{L}_a}$ *being defined in* (5.23)*, and* x^* *is an optimal solution to the original problem* (5.1)*.*

Remark 5.4.1 *The name of the proposed algorithm in this section, "Dual Averaging", comes from the fact that the coordinator calculates the average of local copies of dual variables from all agents at each iteration. It should not be confused with a similar notion in [66, 1, 27, 43, 96], where "dual averaging" refers to taking the average of subgradients.*

The two synchronous algorithms proposed in this chapter have a common drawback: a central coordinator or an agent that is able to communicate with every other agent is

required to update certain variable that is used in every agent's local update. This may not be feasible for some applications. Even if it is feasible, the algorithm's convergence is vulnerable against the single point failure on the central coordinator. Distributed algorithms that does not require a central coordinator and central data aggregation give more flexibility and resilience against single point failure. The next chapter will delve into more details.

6. DISTRIBUTED SYNCHRONOUS ALGORITHMS WITHOUT COORDINATOR

In the dual averaging algorithm via Douglas-Rachford splitting in Chapter 5, an indicator function of the dual variables' consensus subspace $\mathcal{A}_\lambda := \{\lambda_a \,|\, \lambda_1 = \cdots = \lambda_L\}$ was added to the Lagrange function to guarantee the agreement among all agents on the value of dual variables. The resolvent operator of this indicator function is the projection onto the set $\mathcal{A}_\lambda$, which is the averaging of all λ_i, $\forall i \in [L]$, and has to be evaluated by a central coordinator.

In order to eliminate the need for a central coordinator and make each agent's local updates involve only its own decision variables and variables from certain neighbors, we introduce an undirected *multiplier graph*, over which agents can exchange information and reach consensus on λ_i's. The multiplier graph is defined as $\mathcal{G}_\lambda = ([L], \mathcal{E}_\lambda)$, $(j,i) \in \mathcal{E}_\lambda$ if agent i can receive certain information from agent j. Agent i's *agent neighbors* are defined as $\mathcal{N}_i = \{j \in [L] \,|\, (j,i) \in \mathcal{E}_\lambda\}$. Let W be a weighted adjacency matrix associated with the multiplier graph $\mathcal{G}_\lambda$ where $W_{ji} > 0$ if $j \in \mathcal{N}_i$, and $W_{ji} = 0$ if otherwise. It is assumed $W_{ii} = 0$, $\forall i \in [L]$, i.e., no self loop exists. Let $\deg(i) := \sum_{j \in \mathcal{N}_i} W_{ji}$ be the weighted degree of agent i. Let $L_\mathcal{G}$ be the corresponding weighted graph Laplacian matrix, i.e., $L_\mathcal{G} = \text{diag}(\deg(1), \ldots, \deg(L)) - W$.

Assumption 6.0.1 *Multiplier graph $\mathcal{G}_\lambda$ is connected.*

With the introduction of this connected, undirected multiplier graph, we can design alternative ways to enforce the consensus among λ_i's.

6.1 Dual Consensus via Operator Augmentation with Graph Laplacian Matrix

For any undirected and connected graph, its graph Laplacian matrix $L_\mathcal{G} \in \mathbf{R}^{L \times L}$ has an eigenvector 1_L corresponding to eigenvalue 0. Therefore, for $p \in \mathbf{R}^{mL}$, $(L_\mathcal{G} \otimes I_m)p = 0_{mL}$

always implies that $p \in \{1_L \otimes q \,|\, q \in \mathbf{R}^m\}$, i.e., vector p always contains L identical sub-vectors. Therefore, we can concatenate the saddle subdifferential operator (5.23) with $(L_{\mathcal{G}} \otimes I_m)\lambda_a$, and correspondingly introduce a set of node auxiliary variables $y_i \in \mathbf{R}^m$ for each agent, and denote $y = (y_1, \ldots, y_L) \in \mathbf{R}^{mL}$. The new operator can be written as

$$T_{\mathcal{L}_a}(x, y, \lambda_a) = \begin{bmatrix} F(x) + N_X(x) + \Phi^T \lambda_a \\ (L_{\mathcal{G}} \otimes I_m)\lambda_a \\ -\Phi x + b_a \end{bmatrix},$$

where $F(x) := (\partial_{x_1} f_1(x_1), \ldots, \partial_{x_L} f_L(x_L)) \in \mathbf{R}^n$, $\lambda_a = (\lambda_1, \ldots, \lambda_L) \in \mathbf{R}^{mL}$, $b_a = (b_1, \ldots, b_L) \in \mathbf{R}^{mL}$, $\Phi = \text{diag}(A_1, \ldots, A_L) \in \mathbf{R}^{mL \times n}$. For any $(x^*, y^*, \lambda_a^*) \in \text{zer}(T_{\mathcal{L}_a})$, we always have $\lambda_a^* \in \mathcal{A}_\lambda$. However, this new operator as it is is not maximally monotone. We can make a small modification and obtain the following new augmented operator $T_{\mathcal{L}_a}$ with $\text{dom}(T_{\mathcal{L}_a}) \neq \emptyset$ and $\text{dom}(T_{\mathcal{L}_a}) \subseteq \mathbf{R}^{n+2mL}$ as

$$T_{\mathcal{L}_a}(x, y, \lambda_a) = \begin{bmatrix} F(x) + N_X(x) + \Phi^T \lambda_a \\ (L_{\mathcal{G}} \otimes I_m)\lambda_a \\ -\Phi x - (L_{\mathcal{G}} \otimes I_m) y + b_a \end{bmatrix}. \tag{6.1}$$

Proposition 6.1.1 *Operator $T_{\mathcal{L}_a}$ as defined in* (6.1) *is maximally monotone.*

Proof Notice that the operator in (6.1) can be decomposed as

$$T_{\mathcal{L}_a}(x, y, \lambda_a) = T_1 + T_2 \tag{6.2}$$

$$= \begin{bmatrix} F(x) + N_X(x) \\ 0 \\ b_a \end{bmatrix} + \begin{bmatrix} 0 & 0 & \Phi^T \\ 0 & 0 & L_{\mathcal{G}} \otimes I_m \\ -\Phi & -L_{\mathcal{G}} \otimes I_m & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ \lambda_a \end{bmatrix}. \tag{6.3}$$

It is apparent that $F(x)+N_X(x)$ is maximally monotone since individual parts are maximally monotone based on previous assumptions. Both 0 and b_a are maximally monotone since they are constant-valued operators. Therefore, T_1 is maximally monotone as it is the concatenation of three maximally monotone operators.

Notice that $T_2 = Hw$ is a linear operator in $w = (x, y, \lambda_a)$, with H being a skew symmetric matrix, since $L_{\mathcal{G}} \otimes I_m$ is symmetric. Thus T_2 is maximally monotone as it is the summation of two maximally monotone operators. ■

Remark 6.1.1 *The augmented operator $T_{\mathcal{L}_a}(x, y, \lambda_a)$ as defined in* (6.1) *can be interpreted as the saddle subdifferential operator of the saddle function*

$$\widehat{\mathcal{L}}_a(x, y, \lambda_a) = \sum_{i=1}^{L} \left(f_i(x_i) + \mathbf{1}_{X_i}(x_i)\right) + \lambda_a^T \Phi x - \lambda_a^T b_a + \lambda_a^T (L_{\mathcal{G}} \otimes I_m) y.$$

which is a convex function in (x, y) for fixed λ_a, and a concave function in λ_a for fixed (x, y). It is straightforward to show that $\widehat{\mathcal{L}}_\lambda(x, y, \lambda_a)$ is closed and proper. The variables $y \in \mathbf{R}^{mL}$ can be thought of as augmented primal variables.

Proposition 6.1.2 *For any $(x^*, y^*, \lambda_a^*) \in \mathrm{zer}(T_{\mathcal{L}_a})$ with $T_{\mathcal{L}_a}$ being defined in* (6.1)*, we have $\lambda_a^* \in \{1_L \otimes \lambda^* \,|\, \lambda^* \in \mathbf{R}^m\}$, i.e., $\lambda_i^* = \lambda_j^* = \lambda^*, \forall i, j \in [L]$. Furthermore, $(x^*, \lambda^*) \in \mathrm{zer}(T_{\mathcal{L}})$ with $T_{\mathcal{L}}$ being defined in* (5.11)*, which means x^* is an optimal solution to the original optimization problem* (2.1)*.*

Proof Suppose $(x^*, y^*, \lambda_a^*) \in \mathrm{zer}(T_{\mathcal{L}_a})$. Then

$$\begin{bmatrix} F(x^*) + N_X(x^*) + \Phi^T \lambda_a^* \\ (L_{\mathcal{G}} \otimes I_m)\lambda_a^* \\ -\Phi x^* + b_a - (L_{\mathcal{G}} \otimes I_m) y^* \end{bmatrix} \ni \begin{bmatrix} 0_n \\ 0_{mL} \\ 0_{mL} \end{bmatrix},$$

Second row block $(L_{\mathcal{G}} \otimes I_m)\lambda_a^* = 0_{mL}$ implies that $\lambda_a^* \in \{1_L \otimes \lambda^* \,|\, \lambda^* \in \mathbf{R}^m\}$ since $L_{\mathcal{G}}$ is the weighted Laplacian matrix of the connected multiplier graph $\mathcal{G}_\lambda$. Therefore, $\lambda_i^* = \lambda_j^* = \lambda^*, \forall i, j \in [L]$.

From the first row block we have $0_n \in F(x^*) + N_X(x^*) + \Phi^T(1_L \otimes \lambda^*)$. Combining with $\Phi = \mathrm{diag}(A_1, \ldots, A_L)$ and $N_X(x^*) = N_{X_1}(x_1^*) \times \cdots \times N_{X_L}(x_L^*)$ yields

$$0_{n_i} \in \partial f_i(x_i^*) + N_{X_i}(x_i^*) + A_i^T \lambda^*, \quad \forall i \in [L]. \tag{6.4}$$

Pre-multiplying both sides of the thrid row block by $1_L^T \otimes I_m$ yields $0_m = \sum_{i \in [L]} (b_i - A_i x_i^*) - (1_L^T \otimes I_m)(L_{\mathcal{G}} \otimes I_m) y^*$. Since $(1_L^T \otimes I_m)(L_{\mathcal{G}} \otimes I_m) y^* = \left((1_L^T L_{\mathcal{G}}) \otimes (I_m)^2\right) y^* = 0_m$, we have

$$0_m = -Ax^* + b. \tag{6.5}$$

(6.4) and (6.5) implies that $(x^*, \lambda^*) \in \mathrm{zer}(T_{\mathcal{L}})$, which means x^* is an optimal solution to the original optimization problem (2.1). This completes the proof. ■

Remark 6.1.2 *The above theorem indicates that every zero point of the augmented operator $T_{\mathcal{L}_a}$ corresponds to one zero point of $T_{\mathcal{L}}$. From the third row block of operator $T_{\mathcal{L}_a}$, it should be noted that essentially we are decomposing the task of finding x^* such that $Ax^* - b = 0_m$ into L smaller tasks of finding x_i^*'s such that $A_i x_i^* - b_i = z_i^* = \sum_{j\in\mathcal{N}_i^\lambda} w_{ij}(y_i^* - y_j^*)$ with $\sum_{i\in[L]} z_i^* = 0_m$. Therefore, auxiliary variables y_i's can be thought of as coordination variables that help to find the proper decomposition of z_i^*'s.*

Inspired by the work in [98, 100], we define a matrix P as follows:

$$P = \begin{bmatrix} \Upsilon & 0 & 0 \\ 0 & \Gamma^{-1} & L_a \\ 0 & L_a & \Sigma^{-1} \end{bmatrix}, \tag{6.6}$$

where $L_a = L_{\mathcal{G}} \otimes I_m$, $\Upsilon = \text{diag}(\upsilon_1 I_{n_1}, \ldots, \upsilon_L I_{n_L})$, $\Gamma = \text{diag}(\gamma_1 I_m, \ldots, \gamma_L I_m)$, $\Sigma = \text{diag}(\sigma_1 I_m, \ldots, \sigma_L I_m)$. Notice that matrix P is symmetric because L_a and each of the diagonal block is symmetric.

Proposition 6.1.3 *Suppose the parameters υ_i, γ_i, and σ_i, $\forall i \in [L]$ satisfy that $\upsilon_i > 0$, $0 < \gamma_i < \frac{1}{2\deg(i)}$, and $0 < \sigma_i < \frac{1}{2\deg(i)}$, $\forall i \in [L]$. Then the matrix P defined in* (6.6) *is positive definite.*

Proof The above conditions guarantee that matrix P is diagonally dominant with positive diagonal entries, thus positive definite. ■

Although the resolvent of $T_{\mathcal{L}_a}(x, y, \lambda_a)$ can be evaluated, it does not yield a distributed updating rule with which agent only utilizes neighboring agents' information. Instead, we will apply the generalized resolvent iteration with matrix P as defined in (6.6). Let $z = (x, y, \lambda_a)$, $z^{k+1} = \widehat{R}_{\mathcal{L}_a}(z^k)$ gives

$$\begin{aligned} & Pz^{k+1} + T(z^{k+1}) \ni Pz^k, \\ \iff & \begin{bmatrix} \Upsilon x^{k+1} \\ \Gamma^{-1} y^{k+1} + L_a \lambda_a^{k+1} \\ L_a y^{k+1} + \Sigma^{-1} \lambda_a^{k+1} \end{bmatrix} + \begin{bmatrix} F(x^{k+1}) + N_X(x^{k+1}) + \Phi^T \lambda_a^{k+1} \\ L_a \lambda_a^{k+1} \\ b_a - \Phi x^{k+1} - L_a y^{k+1} \end{bmatrix} \ni \begin{bmatrix} \Upsilon x^k \\ \Gamma^{-1} y^k + L_a \lambda_a^k \\ L_a y^k + \Sigma^{-1} \lambda_a^k \end{bmatrix}. \end{aligned}$$

The second row block results in

$$y^{k+1} = y^k + \Gamma L_a(\lambda_a^k - 2\lambda_a^{k+1}). \tag{6.7}$$

The third row block simplifies to

$$\lambda_a^{k+1} = \lambda_a^k + \Sigma(L_a y^k + \Phi x^{k+1} - b_a). \tag{6.8}$$

Substitute the above equation into the first row block yields

$$F(x^{k+1}) + N_X(x^{k+1}) + \Phi^T \lambda_a^k + \Phi^T \Sigma(\Phi x^{k+1} - b_a + L_a y^k) + \Upsilon(x^{k+1} - x^k) \ni 0. \tag{6.9}$$

The key reason that the updates for x, y and λ_a can all be completed by each agent using only local information and information from neighboring agents is that for any $s = (s_1, \ldots, s_L) \in \mathbf{R}^{mL}$, $q = (q_1, \ldots, q_L) \in \mathbf{R}^{mL}$, where $s_i, q_i \in \mathbf{R}^m$, $\forall i, j \in [L]$, $s = L_a q$ can be evaluated in a distributed fashion as

$$s_i = \sum_{j \in \mathcal{N}_i} w_{ij}(q_i - q_j), \quad \forall i \in [L].$$

The proposed algorithm, called *dual consensus via operator augmentation with graph Laplacian matrix*, is outlined in Algorithm 4. Each iteration of this algorithm runs in a certain order: each agent i first communicates and obtains y_j^k from neighboring agents and calculates a weighted local consensus error $d_{i,y}^k$ on variable y, which will be then used in its local update of x_i and λ_i in (6.10b) and (6.10c), respectively. Once all agents obtain updated x_i^{k+1} and λ_i^{k+1}, another round of communication takes place during which each agent i collects λ_j^{k+1} from its neighboring agents. Then, each agent calculates a weighted local consensus error $d_{i,\lambda}^k$ on variable λ, which will be used in its local update of y_i in (6.11b). In total, two rounds of synchronization and communication are required during each iteration. The convergence of Algorithm 4 is summarized in the following theorem.

Theorem 6.1.1 *Under Assumptions 2.1.1, 2.1.2, 2.1.3 and 6.0.1, and suppose υ_i, γ_i, and σ_i, $\forall i \in [L]$ are chosen according to Proposition 6.1.3 such that matrix P in* (6.6) *is positive definite. Then, the sequence $(x^k, y^k, \lambda_a^k)_{k \geq 0}$ generated by Algorithm 4 converges to some $(x^*, y^*, \lambda_a^*) \in \mathrm{zer}(T_{\mathcal{L}_a})$ with $T_{\mathcal{L}_a}$ being defined in* (6.1)*, and x^* is an optimal solution to the original problem* (2.1).

Algorithm 4 Dual Consensus via Operator Augmentation with Graph Laplacian Matrix

1: Initialize (x^0, y^0, λ_a^0) and $(d_{i,\lambda}^0)_{i\in[L]}$, set $k = 0$;

2: **while** stopping criterion is not satisfied **do**

3: Each agent i collects y_j^k from neighboring agents $j \in \mathcal{N}_i$;

4: Each agent i updates local decision variables (in parallel) according to

$$d_{i,y}^k = \sum_{j\in\mathcal{N}_i} w_{ij}(y_i^k - y_j^k); \tag{6.10a}$$

$$x_i^{k+1} = \arg\min_{x_i\in X_i} \Big(f_i(x_i) + (\lambda_i^k)^T A_i x_i + \frac{\sigma_i}{2}\|A_i x_i - b_i + d_{i,y}^k\|^2 + \frac{\upsilon_i}{2}\|x_i - x_i^k\|^2 \Big); \tag{6.10b}$$

$$\lambda_i^{k+1} = \lambda_i^k + \sigma_i \left(A_i x_i^{k+1} - b_i + d_{i,y}^k \right); \tag{6.10c}$$

5: Each agent i collects λ_j^{k+1} from neighboring agents $j \in \mathcal{N}_i$;

6: Each agent i updates node auxiliary variables (in parallel) according to

$$d_{i,\lambda}^{k+1} = \sum_{j\in\mathcal{N}_i} w_{ij}(\lambda_i^{k+1} - \lambda_j^{k+1}); \tag{6.11a}$$

$$y_i^{k+1} = y_i^k + \gamma_i \left(d_{i,\lambda}^k - 2d_{i,\lambda}^{k+1} \right); \tag{6.11b}$$

7: $k \leftarrow k + 1$;

8: **end while**

Proof Combing results from Proposition 4.1.3, Proposition 6.1.1, Proposition 6.1.2 and Proposition 6.1.3 yields the desired statement. ■

Remark 6.1.3 *A similar but different algorithm, called proximal dual consensus ADMM was proposed in [18]. The proximal dual consensus ADMM was derived from the standard ADMM algorithm, which is special case of the Douglas-Rachford splitting method, while our algorithm is an instance of the generalized resolvent iteration* (4.4)*. Our convergence proof based on monotone operator theory and fixed point iteration is much more elegant and simpler than the proof in [18]. The proximal dual consensus ADMM requires part of each agent's local constraint set X_i to be a polyhedra constraint of the form $C_i x_i \leq d_i$, whereas*

we do not impose such a requirement. Another difference is that the weighted adjacency matrix is utilized in our algorithm while the standard adjacency matrix was used in [18].

6.2 Dual Consensus via Operator Augmentation with Incidence Matrix

Instead of using the graph Laplacian matrix, the incidence matrix can also be used to augment the saddle subdifferential operator in (5.23) for the purpose of eliminating the central coordinator. We use a similar multiplier graph as defined in the previous section $\mathcal{G}_\lambda = ([L], \mathcal{E}_\lambda)$, $(j, i) \in \mathcal{E}_\lambda$ if agent i can receive certain information from agent j. Suppose there are M edges between agents ($|\mathcal{E}_\lambda| = M$), which are labeled with e_p, $p = 1, \ldots, M$. Without loss of generality, we assume $e_p = (i, j)$ is ordered as from agent/node i to agent/node j. An edge e_p belongs to the in-edge-neighbor set of agent i, $\mathcal{E}_i^{in}$, if i is the ending point of edge e_p; and it belongs to the out-edge-neighbor set of agent i, $\mathcal{E}_i^{out}$, if i is the starting point of edge e_p. Then denote $\mathcal{E}_i = \mathcal{E}_i^{in} \cup \mathcal{E}_i^{out}$ as the edge neighbor set of agent i. Note that $\mathcal{N}_i = \{j \in [L] \,|\, \mathcal{E}_i \cap \mathcal{E}_j \neq \emptyset\}$. Additionally, we define the in-agent-neighbor set, and out-agent-neighbor set of agent i as $\mathcal{N}_i^{in} = \{j \in [L] \,|\, \mathcal{E}_i^{in} \cap \mathcal{E}_j^{out} \neq \emptyset\}$, $\mathcal{N}_i^{out} = \{j \in [L] \,|\, \mathcal{E}_i^{out} \cap \mathcal{E}_j^{in} \neq \emptyset\}$, respectively. We also have $\mathcal{N}_i = \mathcal{N}_i^{in} \cup \mathcal{N}_i^{out}$ and $\mathcal{N}_i^{in} \cap \mathcal{N}_i^{out} = \emptyset$.

Define the incidence matrix of graph $\mathcal{G}_\lambda$ as $V \in \mathbf{R}^{L \times M}$ where

$$V_{ip} = \begin{cases} 1, & \text{if } e_p \in \mathcal{E}_i^{in}; \\ -1, & \text{if } e_p \in \mathcal{E}_i^{out}; \\ 0, & \text{if } e_p \notin \mathcal{E}_i. \end{cases} \tag{6.12}$$

For any graph, we have $1_L^T V = 0_M^T$. Since the graph $\mathcal{G}_\lambda$ is assumed to be connected, we also have $V^T p = 0_M$ if and only if $p \in \{q 1_L \,|\, q \in \mathbf{R}\}$. Therefore, $(V \otimes I_m)^T s = 0_{mM}$ always implies that $s \in \{1_L \otimes q \,|\, q \in \mathbf{R}^m\}$, i.e., vector p always contains L identical sub-vectors. These are the key features of incidence matrix that enables us to use it as a replacement for the graph Laplacian matrix to augment the saddle subdifferential operator in (5.23). We introduce a set of edge auxiliary variables $y_p \in \mathbf{R}^m$ for each edge e_p, and denote $y = (y_1, \ldots, y_M) \in \mathbf{R}^{mM}$.

Assumption 6.2.1 *Without loss of generality, we assume that the auxiliary variable y_p, $p \in [M]$ is maintained and updated by agent i if $e_p \in \mathcal{E}_i^{out}$.*

We can then define another augmented operator $T_{\mathcal{L}_a}$ with $\mathrm{dom}(T_{\mathcal{L}_a}) \neq \emptyset$ and $\mathrm{dom}(T_{\mathcal{L}_a}) \subseteq \mathbf{R}^{n+mL+mM}$ as

$$T_{\mathcal{L}_a}(x, y, \lambda_a) = \begin{bmatrix} F(x) + N_X(x) + \Phi^T \lambda_a \\ (V \otimes I_m)^T \lambda_a \\ -\Phi x - (V \otimes I_m) y + b_a \end{bmatrix} \tag{6.13}$$
$$= \begin{bmatrix} F(x) + N_X(x) \\ 0 \\ b_a \end{bmatrix} + \begin{bmatrix} 0 & 0 & \Phi^T \\ 0 & 0 & (V \otimes I_m)^T \\ -\Phi & -V \otimes I_m & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ \lambda_a \end{bmatrix}.$$

Proposition 6.2.1 *Operator $T_{\mathcal{L}_a}$ as defined in* (6.13) *is maximally monotone.*

Proof This follows from similar arguments as the proof of Proposition 6.1.1. ■

Remark 6.2.1 *The augmented operator $T_{\mathcal{L}_a}(x, y, \lambda_a)$ as defined in* (6.13) *can be interpreted as the saddle subdifferential operator of the saddle function*

$$\widehat{\mathcal{L}}_a(x, y, \lambda_a) = \sum_{i=1}^{L} \left(f_i(x_i) + \mathbf{1}_{X_i}(x_i) \right) + \lambda_a^T \Phi x - \lambda_a^T b_a + y^T (V \otimes I_m)^T \lambda_a,$$

which is a convex function in (x, y) for fixed λ_a, and a concave function in λ_a for fixed (x, y). It is straightforward to show that $\widehat{\mathcal{L}}_\lambda(x, y, \lambda_a)$ is closed and proper. The variables $y \in \mathbf{R}^{mM}$ can be thought of as augmented primal variables.

Proposition 6.2.2 *Suppose $(x^*, y^*, \lambda_a^*) \in \mathrm{zer}(T_{\mathcal{L}_a})$ with $T_{\mathcal{L}_a}$ being defined in* (6.13)*, we have $\lambda_a^* \in \{1_L \otimes \lambda^* \,|\, \lambda^* \in \mathbf{R}^m\}$, i.e., $\lambda_i^* = \lambda_j^* = \lambda^*$, $\forall i, j \in [L]$. Furthermore, $(x^*, \lambda^*) \in \mathrm{zer}(T_{\mathcal{L}})$ with $T_{\mathcal{L}}$ being defined in* (5.11)*, which means x^* is an optimal solution to the original optimization problem* (2.1)*.*

Proof Uses the facts: 1) $(V \otimes I_m)^T s = 0_{mM}$ implies that $s \in \{1_L \otimes q \,|\, q \in \mathbf{R}^m\}$; 2) $(1_L^T \otimes I_m)(V \otimes I_m) y^* = \left((1_L^T V) \otimes (I_m)^2 \right) y^* = 0_m$, $\forall y^* \in \mathbf{R}^{mM}$. Then follows similar arguments as the proof of Proposition 6.1.2. ■

Remark 6.2.2 *There is an intuitive interpretation of the auxiliary variables y_p's. If b in the coupling constraint $Ax = b$ is thought of as the shared resources among all agents, then y_p can be regarded as the resource "flow" on edge e_p between the starting and ending agents to reach the balance or satisfaction of the coupling resource constraints.*

Inspired by the work in [99], we define a matrix P as follows:

$$P = \begin{bmatrix} \Upsilon & 0 & 0 \\ 0 & \Gamma^{-1} & V_a^T \\ 0 & V_a & \Sigma^{-1} \end{bmatrix}, \tag{6.14}$$

where $V_a = V \otimes I_m$, $\Upsilon = \text{diag}(\upsilon_1 I_{n_1}, \ldots, \upsilon_L I_{n_L})$, $\Gamma = \text{diag}(\gamma_1 I_m, \ldots, \gamma_M I_m)$, $\Sigma = \text{diag}(\sigma_1 I_m, \ldots, \sigma_L I_m)$. Notice that matrix P is symmetric.

Proposition 6.2.3 *Suppose the parameters υ_i, σ_i, $\forall i \in [L]$, and γ_p, $\forall p \in [M]$ satisfy that $\upsilon_i > 0$, $0 < \sigma_i < \frac{1}{\deg(i)}$, $\forall i \in [L]$, and $0 < \gamma_p < 0.5$, $\forall p \in [M]$, then the matrix P defined in* (6.14) *is positive definite.*

Proof The above conditions guarantee that matrix P is diagonally dominant with positive diagonal entries, thus positive definite. ■

Similarly, we will apply the generalized resolvent iteration with matrix P as defined in (6.14). Let $z = (x, y, \lambda_a)$, $z^{k+1} = \widehat{R}_{\mathcal{L}_a}(z^k)$ gives

$$\begin{aligned} & Pz^{k+1} + T(z^{k+1}) \ni Pz^k, \\ \iff & \begin{bmatrix} \Upsilon x^{k+1} \\ \Gamma^{-1} y^{k+1} + V_a^T \lambda_a^{k+1} \\ V_a y^{k+1} + \Sigma^{-1} \lambda_a^{k+1} \end{bmatrix} + \begin{bmatrix} F(x^{k+1}) + N_X(x^{k+1}) + \Phi^T \lambda_a^{k+1} \\ V_a^T \lambda_a^{k+1} \\ b_a - \Phi x^{k+1} - V_a y^{k+1} \end{bmatrix} \ni \begin{bmatrix} \Upsilon x^k \\ \Gamma^{-1} y^k + V_a^T \lambda_a^k \\ V_a y^k + \Sigma^{-1} \lambda_a^k \end{bmatrix}. \end{aligned}$$

The second row block results in

$$y^{k+1} = y^k + \Gamma V_a^T (\lambda_a^k - 2\lambda_a^{k+1}).$$

The third row block simplifies to

$$\lambda_a^{k+1} = \lambda_a^k + \Sigma(V_a y^k + \Phi x^{k+1} - b_a).$$

Substitute the above equation into the first row block yields

$$F(x^{k+1}) + N_X(x^{k+1}) + \Phi^T \lambda_a^k + \Phi^T \Sigma(\Phi x^{k+1} - b_a + V_a y^k) + \Upsilon(x^{k+1} - x^k) \ni 0.$$

The key reasons that allow the updates on x, y and λ_a to be performed by each agent in parallel using only local information and information from neighboring agents are for any $s = (s_1, \ldots, s_L) \in \mathbf{R}^{mL}$, $q = (q_1, \ldots, q_M) \in \mathbf{R}^{mM}$, where $s_i, q_p \in \mathbf{R}^m$, $\forall i \in [L]$ and $\forall p \in [M]$:

1. Operation $s = V_a q$ can be evaluated in a distributed fashion as

$$s_i = \sum_{e_p \in \mathcal{E}_i} V_{ip} q_p, \quad \forall i \in [L].$$

2. Operation $q = V_a^T s$ can be evaluated in a distributed fashion as

$$q_p = s_j - s_i, \quad e_p = (i, j) \in \mathcal{E}_\lambda, \quad \forall p \in [M].$$

The proposed algorithm, called *dual consensus via operator augmentation with incidence matrix*, is outlined in Algorithm 5. Each iteration of this algorithm runs in a certain order: each agent i first communicates and obtains y_p^k from its in-agent-neighbors $\mathcal{N}_i^{in}$ and calculates a $d_{i,y}^k$ according to (6.15a), which will be then used in its local update of x_i and λ_i in (6.15b) and (6.15c), respectively. Once all agents obtain updated x_i^{k+1} and λ_i^{k+1}, another round of communication takes place during which each agent i collects λ_j^{k+1} from its out-agent-neighbors $\mathcal{N}_i^{out}$. Then, each agent calculates $d_{p,\lambda}^{k+1}$ according to (6.16a), which will be used in its local update of edge auxiliary variables y_p in (6.16b). Obviously, two rounds of synchronization and communication are required during each iteration. The convergence of Algorithm 5 is summarized in the following theorem.

Remark 6.2.3 *Intuitively, auxiliary variables $d_{i,y}^k \in \mathbf{R}^m$ represents the aggregate amount of resource change in agent i at the kth iteration, while auxiliary variables $d_{p,\lambda}^k \in \mathbf{R}^m$ can be thought of as the resource "flow" on edge e_p at the kth iteration. At steady state, $d_{p,\lambda}^k$ goes to 0, $\forall p \in [M]$, and $d_{i,y}^k$ goes to certain fixed values depending on the network topology and also the allocation of b_i's in b.*

Algorithm 5 Dual Consensus via Operator Augmentation with Incidence Matrix

1: Initialize (x^0, y^0, λ_a^0) and $(d_{p,\lambda}^0)_{p\in[M]}$, set $k = 0$;

2: **while** stopping criterion is not satisfied **do**

3: Each agent i collects y_p^k from in-agent-neighbors $j \in \mathcal{N}_i^{in}$ where $e_p = (j, i)$;

4: Each agent i updates its local decision variables (in parallel) according to

$$d_{i,y}^k = \sum_{e_p\in\mathcal{E}_i} V_{ip} y_p^k = \sum_{e_p\in\mathcal{E}_i^{in}} y_p^k - \sum_{e_p\in\mathcal{E}_i^{out}} y_p^k; \tag{6.15a}$$

$$x_i^{k+1} = \arg\min_{x_i\in X_i} \Big(f_i(x_i) + (\lambda_i^k)^T A_i x_i + \frac{\sigma_i}{2}\|A_i x_i - b_i + d_{i,y}^k\|^2 + \frac{\upsilon_i}{2}\|x_i - x_i^k\|^2 \Big); \tag{6.15b}$$

$$\lambda_i^{k+1} = \lambda_i^k + \sigma_i \left(A_i x_i^{k+1} - b_i + d_{i,y}^k \right); \tag{6.15c}$$

5: Each agent i collects λ_j^{k+1} from out-agent-neighbors $j \in \mathcal{N}_i^{out}$;

6: Each agent i updates its edge auxiliary variables (in parallel) according to

$$d_{p,\lambda}^{k+1} = \lambda_j^{k+1} - \lambda_i^{k+1}, \text{ where } e_p = (i, j),\ \forall j \in \mathcal{N}_i^{out}; \tag{6.16a}$$

$$y_p^{k+1} = y_p^k + \gamma_p \left(d_{p,\lambda}^k - 2d_{p,\lambda}^{k+1} \right), \quad \forall e_p \in \mathcal{E}_i^{out}; \tag{6.16b}$$

7: $k \leftarrow k + 1$;

8: **end while**

Theorem 6.2.1 *Under Assumptions 2.1.1, 2.1.2, 2.1.3, 6.0.1 and 6.2.1, and suppose* υ_i, σ_i, $\forall i \in [L]$, *and* γ_p, $\forall p \in [M]$ *are chosen according to Proposition 6.2.3 such that matrix* P *in* (6.14) *is positive definite. Then, the sequence* $(x^k, y^k, \lambda_a^k)_{k\geq 0}$ *generated by Algorithm 5 converges to some* $(x^*, y^*, \lambda_a^*) \in \mathrm{zer}(T_{\mathcal{L}_a})$ *with* $T_{\mathcal{L}_a}$ *being defined in* (6.13)*, and* x^* *is an optimal solution to the original problem* (2.1).

Proof Combing results from Proposition 4.1.3, Proposition 6.2.1, Proposition 6.2.2 and Proposition 6.2.3 yields the desired statement. ■

Remark 6.2.4 *Algorithm 5 should only be applied to network where* $M \approx L$. *Because when incidence matrix is utilized in the augmented operator* (6.13)*, an additional edge*

auxiliary variable $y_p \in \mathbf{R}^m$ is introduced for each edge e_p. In the scenario where $M \gg L$, such as when the multiplier graph is nearly fully connected, this will lead to an excessive number of auxiliary variables. In situation like this, one should consider apply Algorithm 4, where the graph Laplacian matrix is utilized, and a fixed number (L) of auxiliary variables need to be introduced.

6.3 Dual Consensus via Operator Augmentation and Splitting

For either dual consensus via operator augmentation with graph Laplacian matrix or with incidence matrix, the local updates of each agent contain the following step

$$x^{k+1} = \underset{x_i \in X_i}{\arg\min} \left(f_i(x_i) + (\lambda_i^k)^T A_i x_i + \frac{\sigma_i}{2} \|A_i x_i - b_i + d_{i,y}^k\|^2 + \frac{\alpha_i}{2} \|x - x_i^k\|^2 \right), \quad (6.17)$$

which is special case of

$$x^{k+1} = \underset{x_i}{\arg\min} \Big(f_{1,i}(x_i) + f_{2,i}(x_i) + (\lambda_i^k)^T A_i x_i + \frac{\sigma_i}{2} \|A_i x_i - b_i + d_{i,y}^k\|^2 + \frac{\alpha_i}{2} \|x - x_i^k\|^2 \Big), \quad (6.18)$$

by letting $f_{1,i}(x_i) = f_i(x_i)$ and $f_{2,i}(x_i) = \mathbf{1}_{X_i}(x_i)$. For some other problems where $X = \mathbf{R}^n$, but local objective function of each agent contains two parts with different smoothness properties, the local x update step can also be written as (6.18). It is highly likely that this step is the bottleneck of the algorithm from a computational perspective since all other steps of the updates are linear operations. The algorithm could be more efficient if we could break (6.18) into two steps including one step of unconstrained optimization problem with $f_{1,i}(x_i)$ and one step of proximal operator evaluation of $f_{2,i}(x_i)$.

Instead of problem (2.1), we consider the following equivalent formulation

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) = \sum_{i \in [L]} \left(f_{1,i}(x_i) + f_{2,i}(x_i) \right) \\ \text{subject to} \quad & Ax = b, \quad i \in [L]. \end{aligned} \quad (6.19)$$

We can write out the Lagrange function of (6.19)

$$\mathcal{L}(x, \lambda) = \sum_{i \in [L]} \left(f_{1,i}(x_i) + f_{2,i}(x_i) + \lambda^T A_i x_i - \lambda^T b_i \right), \quad (6.20)$$

which is a closed proper saddle function under the same assumptions as before. Thus, the saddle points of (6.20) are optimal primal-dual pairs of (6.19). We can write out the saddle subdifferential operator of (6.20), and augment it with the graph Laplacian matrix using the same strategy as before and obtain the following augmented operator

$$T_{\mathcal{L}_a}(x, y, \lambda_a) = \begin{bmatrix} F_1(x) + F_2(x) + \Phi^T \lambda_a \\ (L_{\mathcal{G}} \otimes I_m)\lambda_a \\ -\Phi x - (L_{\mathcal{G}} \otimes I_m) y + b_a \end{bmatrix}, \tag{6.21}$$

where $F_r(x) := (\partial_{x_1} f_{r,1}(x_1), \ldots, \partial_{x_L} f_{r,L}(x_L))$, $r = 1, 2$. Similarly, we can show that $T_{\mathcal{L}_a}(x, y, \lambda_a)$ as defined in (6.21) is maximally monotone, and for any $(x^*, y^*, \lambda_a^*) \in \mathrm{zer}(T_{\mathcal{L}_a})$, we have $\lambda_a^* \in \{1_L \otimes \lambda^* \,|\, \lambda^* \in \mathbf{R}^m\}$, i.e., $\lambda_i^* = \lambda_j^* = \lambda^*$, $\forall i, j \in [L]$, and x^* is an optimal solution to the original optimization problem (6.19).

Define matrix P as follows

$$P = \begin{bmatrix} \Upsilon & 0 & 0 \\ 0 & \Gamma^{-1} & L_a/2 \\ 0 & L_a/2 & \Sigma^{-1} \end{bmatrix}, \tag{6.22}$$

where $L_a = L_{\mathcal{G}} \otimes I_m$, $\Upsilon = \mathrm{diag}(\upsilon_1 I_{n_1}, \ldots, \upsilon_L I_{n_L})$, $\Gamma = \mathrm{diag}(\gamma_1 I_m, \ldots, \gamma_L I_m)$, $\Sigma = \mathrm{diag}(\sigma_1 I_m, \ldots, \sigma_L I_m)$. Notice that matrix P is symmetric since L_a is symmetric.

Proposition 6.3.1 *Suppose the parameters υ_i, γ_i, and σ_i, $\forall i \in [L]$ satisfy that $\upsilon_i > 0$, $0 < \gamma_i < \frac{1}{\deg(i)}$, and $0 < \sigma_i < \frac{1}{\deg(i)}$, $\forall i \in [L]$, then the matrix P defined in* (6.22) *is positive definite.*

Proof The above conditions guarantee that matrix P is diagonally dominant with positive diagonal entries, thus positive definite. ■

Instead of directly applying the generalized reolvent iteration (4.4) on operator $T_{\mathcal{L}_a}$, we note that $T_{\mathcal{L}_a}(x, y, \lambda_a)$ can be written as

$$T_{\mathcal{L}_a}(x, y, \lambda_a) = T_1 + T_2$$

$$= \begin{bmatrix} F_1(x) + \Phi^T \lambda_a \\ \frac{1}{2}(L_{\mathcal{G}} \otimes I_m)\lambda_a \\ -\Phi x - \frac{1}{2}(L_{\mathcal{G}} \otimes I_m)y + b_a \end{bmatrix} + \begin{bmatrix} F_2(x) \\ \frac{1}{2}(L_{\mathcal{G}} \otimes I_m)\lambda_a \\ -\frac{1}{2}(L_{\mathcal{G}} \otimes I_m)y \end{bmatrix},$$

with T_1 and T_2 both being maximally monotone operators. We adopt the generalized Douglas-Rachford splitting method (4.5) with matrix P defined in (6.22). Starting from any $z^0 = (x^0, y^0, \lambda_a^0)$, for $k = 0, 1, \ldots$, let

$$\begin{aligned} w^{k+1} &= \widehat{R}_{T_1}(z^k); \\ z^{k+1} &= z^k + 2\alpha \left(\widehat{R}_{T_2}(2w^{k+1} - z^k) - w^{k+1} \right), \end{aligned}$$

where $\alpha \in (0, 1)$; $\widehat{R}_{T_1}$ and $\widehat{R}_{T_2}$ are the generalized resolvent operator with matrix P in (6.22) of T_1 and T_2, respectively. Then by Proposition 4.2.1, the sequence $(w^k)_{k\geq 0}$ converges to some $w^* \in \mathrm{zer}(T_{\mathcal{L}_a})$. Following similar derivations as in previous sections, we can obtain distributed solutions for $\widehat{R}_{T_1}$ and $\widehat{R}_{T_2}$ at each iteration (omitted here for brevity). Combining them together, we will obtain the *dual consensus via operator augmentation and splitting with graph Laplacian matrix* algorithm, which is outlined in Algorithm 6.

In Algorithm 6, Step 1 corresponds to the evaluation of $\widehat{R}_{T_1}$, Step 2 corresponds to the evaluation of $2w^{k+1} - z^k$, Step 3 corresponds to the evaluation of $\widehat{R}_{T_2}$, and Step 4 corresponds to the evaluation of $z^{k+1} = z^k + 2\alpha\big(\widehat{R}_{T_2}(2w^{k+1} - z^k) - w^{k+1}\big)$. Since we need to evaluate two generalized resolvent operators during each iteration, a total of four rounds of synchronization, communication and information exchange are required, which happen in lines 3, 5, 7, and 9. As expected, the constrained optimization step (6.17) has been broken into two steps: 1) an unconstrained optimization problem involving $f_{1,i}(x_i)$ (second line of Step 1a) 2) the evaluation of the proximal operator corresponding to $f_{2,i}(x_i)$ (Step 3a), and some additional linear operations.

Theorem 6.3.1 *Under Assumptions 2.1.1, 2.1.2, 2.1.3 and 6.0.1, and suppose υ_i, γ_i, and σ_i, $\forall i \in [L]$ are chosen according to Proposition 6.3.1 such that matrix P in* (6.22) *is positive definite. Then, the sequence $(\overline{x}^k, \overline{y}^k, \overline{\lambda}_a^k)_{k\geq 0}$ generated by Algorithm 6 converges to some $(x^*, y^*, \lambda_a^*) \in \mathrm{zer}(T_{\mathcal{L}_a})$ with $T_{\mathcal{L}_a}$ being defined in* (6.21)*, and x^* is an optimal solution to the original problem* (6.19)*.*

Algorithm 6 Dual Consensus via Operator Augmentation and Splitting with Graph Laplacian Matrix

1: Initialize (x^0, y^0, λ_a^0) and $(d_{i,\lambda}^0)_{i\in[L]}$, set $k = 0$;

2: **while** stopping criterion is not satisfied **do**

3: Each agent i collects y_j^k from neighboring agents $j \in \mathcal{N}_i$;

4: Each agent i updates local decision variables (in parallel) according to

$$\text{Step 1a: } d_{i,y}^k = \sum_{j\in\mathcal{N}_i} w_{ij}(y_i^k - y_j^k)/2;$$

$$\overline{x}_i^k = \arg\min_{x_i} \left(f_{1,i}(x_i) + (\lambda_i^k)^T A_i x_i + \frac{\sigma_i}{2}\|A_i x_i - b_i + d_{i,y}^k\|^2 + \frac{\upsilon_i}{2}\|x_i - x_i^k\|^2 \right);$$

$$\overline{\lambda}_i^k = \lambda_i^k + \sigma_i \left(A_i \overline{x}_i^k - b_i + d_{i,y}^k \right);$$

5: Each agent i collects $\overline{\lambda}_j^k$ from neighboring agents $j \in \mathcal{N}_i$;

6: Each agent i updates local decision variables (in parallel) according to

$$\text{Step 1b: } \overline{d}_{i,\lambda}^k = \sum_{j\in\mathcal{N}_i} w_{ij}(\overline{\lambda}_i^k - \overline{\lambda}_j^k)/2,\ \overline{y}_i^k = y_i^k + \gamma_i \left(d_{i,\lambda}^k - 2\overline{d}_{i,\lambda}^k \right);$$

$$\text{Step 2: } \widehat{x}_i^k = 2\overline{x}_i^k - x_i^k,\ \widehat{\lambda}_i^k = 2\overline{\lambda}_i^k - \lambda_i^k,\ \widehat{y}_i^k = 2\overline{y}_i^k - y_i^k,\ \widehat{d}_{i,\lambda}^k = 2\overline{d}_{i,\lambda}^k - d_{i,\lambda}^k,$$

7: Each agent i collects $\widehat{y}_j^k$ from neighboring agents $j \in \mathcal{N}_i$;

8: Each agent i updates local decision variables (in parallel) according to

$$\text{Step 3a: } \widehat{x}_i^{k+1} = \operatorname*{prox}_{f_{2,i}/\alpha_i} \left(\widehat{x}_i^k \right),\ \widehat{d}_{i,y}^k = \sum_{j\in\mathcal{N}_i} w_{ij}(\widehat{y}_i^k - \widehat{y}_j^k)/2,\ \widehat{\lambda}_i^{k+1} = \widehat{\lambda}_i^k + \sigma_i \widehat{d}_{i,y}^k$$

9: Each agent i collects $\widehat{\lambda}_i^{k+1}$ from neighboring agents $j \in \mathcal{N}_i$;

10: Each agent i updates local decision variables (in parallel) according to

$$\text{Step 3b: } \widehat{d}_{i,\lambda}^{k+1} = \sum_{j\in\mathcal{N}_i} w_{ij}(\widehat{\lambda}_i^{k+1} - \widehat{\lambda}_j^{k+1})/2,\ \widehat{y}_i^{k+1} = \widehat{y}_i^k + \gamma_i \left(\widehat{d}_{i,\lambda}^k - 2\widehat{d}_{i,\lambda}^{k+1} \right)$$

$$\text{Step 4: } x_i^{k+1} = x_i^k + 2\alpha(\widehat{x}_i^{k+1} - \overline{x}_i^k),\ \lambda_i^{k+1} = \lambda_i^k + 2\alpha(\widehat{\lambda}_i^{k+1} - \overline{\lambda}_i^k)$$

$$y_i^{k+1} = y_i^k + 2\alpha(\widehat{y}_i^{k+1} - \overline{y}_i^k),\ d_{i,\lambda}^{k+1} = d_{i,\lambda}^k + 2\alpha(\widehat{d}_{i,\lambda}^{k+1} - \overline{d}_{i,\lambda}^k)$$

11: $k \leftarrow k + 1$;

12: **end while**

Proof Straightforward. ■

Remark 6.3.1 *If $f_{1,i}(x_i)$ is a quadratic function, then the corresponding unconstrained optimization problem has an explicit analytical solution, which reduces to linear operations. If $f_{2,i}(x_i)$ is a convex indicator function, its proximal operator is simply the projection onto the underlying convex set. Furthermore, the evaluations of certain functions' proximal operators have explicit formulae, such as $\|\cdot\|_1$ and $\|\cdot\|_\infty$; and for many other functions, the proximal operators can be evaluated in efficient ways. More details can be found in Chapter 6 of [69]. Therefore, with the application of the generalized Douglas-Rachford splitting, although more synchronization and information exchange are required, the computation time and complexity are expected to be significantly reduced for many problems.*

Remark 6.3.2 *Notice that we can switch the roles between $\widehat{R}_{T_1}$ and $\widehat{R}_{T_2}$ and obtain a slightly different algorithm with the same convergence results. In addition, instead of graph Laplacian matrix, we can also use incidence matrix in the augmented operator* (6.21)*. Following similar derivations as before, we will be able to obtain two new algorithms. Details of these algorithms are omitted for brevity.*

6.4 Further Extensions

In the previous section, we applied generalized Douglas-Rachford splitting technique on problem (6.19), where each local objective function has two parts. In some other problems, each local objective function may have three parts $f_{1,i} + f_{2,i} + f_{3,i}$. One special case is that local objective function is the summation of a smooth function and a nonsmooth function, subject to some local convex constraint set. If one of the functions is Lipschitz continuous, then its subdifferential is cocoercive. This is referred to as Baillon-Haddad Theorem [5, 8]. In these cases, the generalized Davis-Yin three operator splitting technique (4.6) could potentially be applied, we leave this as future work.

7. DISTRIBUTED ASYNCHRONOUS ALGORITHM WITHOUT COORDINATOR

The synchronous algorithms we discussed in the previous sections assume that there is a synchronization clock among all agents. During each iteration, there is one or multiple rounds of synchronization required. Because of the differences in the computational power as well as computation schedules among agents, the times required to perform a single update for different agents might be vastly different. Therefore, all the agents need to wait for the slowest one, resulting much idle time and waste of computational resource. Moreover, in many practical applications, a synchronization clock among all agents might not be feasible. Therefore, it is desirable to design an asynchronous version of the algorithm.

In this chapter, we will employ the tools from randomized block-coordinate fixed point iteration of nonexpansive operators to design distributed asynchronous algorithms for our multi-agent optimization problem. Each iteration of the previously proposed distributed synchronous algorithms can be written as a special case of $z^{k+1} = \mathcal{T}(z^k)$ where $\mathcal{T}$ is an averaged or nonexpansive operator. Instead of evaluating the whole operator at the same time, only the coordinates that correspond to a certain activated agent will be updated while others remain unchanged, and we denote the new operator corresponding to agent i's update as $\mathcal{T}_i$. In the presence of delayed information, an extra step size η_i needs to be added towards the direction of update, $z^k - \mathcal{T}_i(z^k)$, for relaxation. That is, the new update rule is $z^{k+1} = z^k - \eta_i(\widehat{z}^k - \mathcal{T}_i(\widehat{z}^k))$ where $\widehat{z}^k$ characterizes the delayed information. Under the assumption that the information delay is uniformly bounded by a finite value, it is shown that η_i's can be chosen small enough to still guarantee the algorithm's convergence to some fixed point of the operator $\mathcal{T}$ in a probabilistic sense.

For the synchronous algorithms proposed in Chapter 6, the updates of x_i, y_p and λ_i need to follow a certain sequence. For example, in Algorithm 5, for agent i, the computation of its local edge auxiliary variable y_p^{k+1} require information about $d_{p,\lambda}^{k+1}$, which depends on

λ_j^{k+1} from neighboring agent $j \in \mathcal{N}_i^{out}$. This type of coupling is fine in the synchronous algorithms as we can impose multiple rounds of synchronization between neighbors in each iteration, however, it prevents the implementation of an asynchronous algorithm as it requires neighboring agents of an activated agent to also perform updates, which contradicts the idea of asynchrony. Therefore, inspired by [99] we start from modifying the P matrix in the dual consensus algorithm via operator augmentation with incidence matrix, so that the local updates of x_i, y_p, λ_i only need information of local and neighboring variables from the last iteration.

7.1 Modified Synchronous Dual Consensus via Operator Augmentation with Incidence Matrix

With the incidence matrix V defined in (6.12), we can compute the *edge Laplacian* matrix $L^e = V^T V = (L^e)^T \in \mathbf{R}^{M\times M}$, and it can be verified that

$$L^e_{pq} = \begin{cases} 2, & \text{if } p = q; \\ 1, & \text{if } e_p \text{ and } e_q \text{ share the same starting or ending node;} \\ -1, & \text{if the ending node of } e_p\ (e_q) \text{ is the starting node of } e_q\ (e_p); \\ 0, & \text{if } e_p \text{ and } e_q \text{ do not intersect.} \end{cases}$$

Interested readers can find more details regarding incidence matrix and edge Laplacian matrix in [60]. Define the edge neighbor set of edge e_p as $\mathcal{N}_p^e = \{e_q \,|\, L^e_{pq} \neq 0\}$. It is straightforward to see $\mathcal{N}_p^e = \mathcal{E}_i \cup \mathcal{E}_j$ if $e_p = (i, j)$.

Assumption 7.1.1 *Although auxiliary variable y_p associated with edge $e_p = (i, j) \in \mathcal{E}_i^{out}$ is updated by agent i, we assume that it is also accessible to agent j.*

We use the same augmented operator as defined in (6.13). Define a new matrix $\widehat{P}$ [99] as follows:

$$\widehat{P} = \begin{bmatrix} \Upsilon & 0 & \Phi^T \\ 0 & \Gamma^{-1} & V_a^T \\ \Phi & V_a & \Sigma^{-1} \end{bmatrix}, \tag{7.1}$$

where $V_a = V \otimes I_m$, $\Upsilon = \text{diag}(\upsilon_1 I_{n_1}, \ldots, \upsilon_L I_{n_L})$, $\Gamma = \text{diag}(\gamma_1 I_m, \ldots, \gamma_M I_m)$, $\Sigma = \sigma I_{mL}$, $\Phi = \text{diag}(A_1, \ldots, A_L)$. Notice that matrix P is symmetric.

Proposition 7.1.1 *Suppose the parameters σ, υ_i, $\forall i \in [L]$, and γ_p, $\forall p \in [M]$ satisfy that $0 < \sigma < \frac{1}{\delta}$, $0 < \upsilon_i < \frac{1}{\rho_i}$, $\forall i \in [L]$, and $0 < \gamma_p < 0.5$, $\forall p \in [M]$, where*

$$\delta = \max_{i\in[L]} \left\{ \deg(i) + \max_{j\in[L]} \left\{ \sum_{t\in[n_i]} |(A_i)_{jt}| \right\} \right\},$$
$$\rho_i = \max_{j\in[n_i]} \left\{ \sum_{t\in[L]} |(A_i^T)_{jt}| \right\}, \quad \forall i \in [L],$$

then the matrix $\widehat{P}$ defined in (7.1) *is positive definite.*

Proof The above conditions guarantee that matrix $\widehat{P}$ is diagonally dominant with positive diagonal entries, thus positive definite. ■

Remark 7.1.1 *In Proposition 7.1.1, $(A_i)_{jt}$ and $(A_i^T)_{jt}$ denote the entries at the j-th row, t-th column of matrices A_i and A_i^T, respectively.*

Let $z = (x, y, \lambda_a)$, the generalized resolvent iteration with matrix $\widehat{P}$ can be written as

$$\begin{aligned}
&\widehat{P}z^{k+1} + T(z^{k+1}) \ni \widehat{P}z^k, \\
\iff &\begin{bmatrix} \Upsilon x^{k+1} + \Phi^T \lambda_a^{k+1} \\ \Gamma^{-1} y^{k+1} + V_a^T \lambda_a^{k+1} \\ \Phi x^{k+1} + V_a y^{k+1} + \Sigma^{-1} \lambda_a^{k+1} \end{bmatrix} + \begin{bmatrix} F(x^{k+1}) + N_X(x^{k+1}) + \Phi^T \lambda_a^{k+1} \\ V_a^T \lambda_a^{k+1} \\ b_a - \Phi x^{k+1} - V_a y^{k+1} \end{bmatrix} \\
&\qquad\qquad \ni \begin{bmatrix} \Upsilon x^k + \Phi^T \lambda_a^k \\ \Gamma^{-1} y^k + V_a^T \lambda_a^k \\ \Phi x^k + V_a y^k + \Sigma^{-1} \lambda_a^k \end{bmatrix}.
\end{aligned}$$

The third row block simplifies to

$$\lambda_a^{k+1} = \lambda_a^k + \sigma(V_a y^k + \Phi x^k - b_a). \tag{7.2}$$

The second row block results in

$$y^{k+1} = y^k + \Gamma V_a^T (\lambda_a^k - 2\lambda_a^{k+1})$$

$$\begin{aligned}
\Longrightarrow \quad & y^{k+1} = y^k + \Gamma V_a^T \left(-\lambda_a^k - 2\sigma(V_a y^k + \Phi x^k - b_a)\right) \\
\Longrightarrow \quad & y^{k+1} = y^k - \Gamma V_a^T \lambda_a^k - 2\sigma\Gamma V_a^T V_a y^k - 2\sigma\Gamma V_a^T(\Phi x^k - b_a) \\
\Longrightarrow \quad & y^{k+1} = y^k - \Gamma V_a^T \lambda_a^k - 2\sigma\Gamma L^e y^k - 2\sigma\Gamma V_a^T(\Phi x^k - b_a). \qquad (7.3)
\end{aligned}$$

The first row block results in

$$F(x^{k+1}) + N_X(x^{k+1}) + \Phi^T(2\lambda_a^{k+1} - \lambda_a^k) + \Upsilon(x^{k+1} - x^k) \ni 0,$$

then substitute $\lambda_a^{k+1} = \lambda_a^k + \sigma(V_a y^k + \Phi x^k - b_a)$ into the above relation yields

$$F(x^{k+1}) + N_X(x^{k+1}) + \Phi^T \lambda_a^k + 2\sigma\Phi^T(\Phi x^k - b_a + V_a y^k) + \Upsilon(x^{k+1} - x^k) \ni 0. \quad (7.4)$$

The updates of x_i and λ_i require the information of $(y_p)_{e_p \in \mathcal{E}_i} = \left((y_p)_{e_p \in \mathcal{E}_i^{out}}, (y_p)_{e_p \in \mathcal{E}_i^{in}}\right)$, of which $(y_p)_{e_p \in \mathcal{E}_i^{in}}$ are from agent i's in-agent-neighbors $\mathcal{N}_i^{in}$. The update of $(y_p)_{e_p \in \mathcal{E}_i^{out}}$ requires $(\lambda_j)_{j \in \mathcal{N}_i^{out}}$, $(A_j x_j - b_j)_{j \in \mathcal{N}_i^{out}}$, and $(y_q)_{e_q \in \mathcal{N}_p^e} = (y_q)_{q \in \{q \,|\, L_{pq}^e \neq 0\}}$, which contains $(y_q)_{e_q \in \mathcal{E}_i^{in}}$ from agent i's in-agent-neighbors $\mathcal{N}_i^{in}$, $\left((y_q)_{e_q \in \mathcal{E}_j^{out}}\right)_{j \in \mathcal{N}_i^{out}}$ from agent i's out-agent-neighbors $\mathcal{N}_i^{out}$, and $\left((y_q)_{e_q \in \mathcal{E}_j^{in}}\right)_{j \in \mathcal{N}_i^{out}}$ from the in-agent-neighbors of agent i's out-agent-neighbors $\mathcal{N}_i^{out}$.

Therefore, the local updates of agent i call for variables maintained by its one-hop and two-hop node/agent neighbors. Since it is assumed that regardless of edge directions that we arbitrarily assigned, the corresponding auxiliary edge variables are accessible to both the starting and ending nodes/agents, for agent i, all the information needed for local updates can be received from its immediate neighbors $\mathcal{N}_i$. Thus, (7.2), (7.3) and (7.4) can all be carried in a distributed fashion. This modified distributed synchronous dual consensus algorithm via operator augmentation with incidence matrix is summarized in Algorithm 7. The convergence result is similar to Theorem 6.2.1 as long as parameters in matrix $\widehat{P}$ are chosen according to Proposition 7.1.1.

Notice that the updates of x_i, y_p and λ_i are independent, i.e., no particular order needs to be followed. In addition, only one round of synchronization is required at the beginning of each iteration, and the local updates of an agent only utilize local and neighbor's variables from the last iteration. These are key features allowing the implementation of

Algorithm 7 Modified Dual Consensus via Operator Augmentation with Incidence Matrix

1: Initialize (x^0, y^0, λ_a^0), set $k = 0$;

2: **while** stopping criterion is not satisfied **do**

3: Each agent i collects λ_j^k, $A_j x_j^k - b_j$ from its out-agent-neighbors $j \in \mathcal{N}_i^{out}$, and $(y_q)_{e_q \in \mathcal{N}_p^e}$ from its neighbors $j \in \mathcal{N}_i$;

4: Each agent i updates its local decision variables (in parallel) according to

$$d_{i,y}^k = \sum_{e_p \in \mathcal{E}_i} V_{ip} y_p^k = \sum_{e_p \in \mathcal{E}_i^{in}} y_p^k - \sum_{e_p \in \mathcal{E}_i^{out}} y_p^k;$$

$$x_i^{k+1} = \arg\min_{x_i \in X_i} \Big(f_i(x_i) + \left(\lambda_i^k + 2\sigma_i (A_i x_i^k - b_i + d_{i,y}^k)\right)^T A_i x_i + \frac{\alpha_i}{2} \|x_i - x_i^k\|^2 \Big);$$

$$\lambda_i^{k+1} = \lambda_i^k + \sigma_i \left(A_i x_i^k - b_i + d_{i,y}^k\right);$$

$$d_{p,e}^k = \sum_{e_q \in \mathcal{N}_p^e} L_{pq}^e y_q^k, \quad \forall e_p \in \mathcal{E}_i^{out};$$

$$y_p^{k+1} = y_p^k - \gamma_p(\lambda_j^k - \lambda_i^k) - 2\gamma_p \sigma (A_j x_j^k - b_j - A_i x_i^k + b_i) - 2\gamma_p \sigma d_{p,e}^k, \quad \forall e_p = (i,j) \in \mathcal{E}_i^{out};$$

5: $k \leftarrow k + 1$;

6: **end while**

an asynchronous version of the algorithm. We will build upon Algorithm 7 and develop a distributed asynchronous algorithm in the next section.

7.2 Asynchronous Dual Consensus via Operator Augmentation with Incidence Matrix Considering Delays

In an asynchronous setting, each agent has its individual clock and update its local variables independently. We use a different iteration increment rule. As opposed to the synchronous setting where the iteration counter increases by one after all agents finish one round of updating of all their local variables (or the slowest agent finishes), the iteration

counter increases by one whenever an arbitrary agent completes an update on its local variable in the asynchronous setting.

Assumption 7.2.1 *There is a virtual global iteration counter k, which increases by one whenever an arbitrary agent finishes an update on its local variable.*

Recall that the synchronous algorithm in Algorithm 7 roots from evaluating the generalized resolvent operator of $T_{\mathcal{L}_a}(x, y, \lambda_a)$ as defined in (6.13) with matrix $\widehat{P}$ in (7.1). For notation simplicity, we will denote the generalized resolvent operator of $T_{\mathcal{L}_a}$ as $\mathcal{T} := \widehat{R}_{T_a}$. In Algorithm 7, the entire operator $\mathcal{T}$ is evaluated at each iteration. Since $z = (x, y, \lambda_a) \in \mathbf{R}^{n+mL+mM}$ is the aggregate global variable, we can define L index vectors $\mathcal{I}^i \in \mathbf{R}^{n+mL+ML}$, $\forall i \in [L]$, where

$$\mathcal{I}^i_j = \begin{cases} 1, & \text{if } z_j \text{ is a coordinate of agent } i\text{'s local variable, } \big(x_i, (y_p)_{e_p \in \mathcal{E}_i^{out}}, \lambda_i\big); \\ 0, & \text{otherwise.} \end{cases}$$

Notice that z_j and $\mathcal{I}^i_j$ denote the jth element of vectors z and $\mathcal{I}^i$, respectively. Index vectors $\mathcal{I}^i$, $\forall i \in [L]$ select the coordinates of each agent's local variables from the global variable z. Next, we define operator $\mathcal{T}_i : \mathbf{R}^{n+mL+mM} \to \mathbf{R}^{n+mL+mM}$, which corresponds to the update on the global variable whenever agent i is activated:

$$\mathcal{T}_i(z)_j := \begin{cases} \mathcal{T}(z)_j, & \text{if } \mathcal{I}^i_j = 1; \\ z_j, & \text{if } \mathcal{I}^i_j = 0, \end{cases}$$

where $\mathcal{T}(z)_j$ and $\mathcal{T}_i(z)_j$ denote the jth element of vectors $\mathcal{T}(z)$ and $\mathcal{T}_i(z)$, respectively. Note that $\mathcal{T}_i$ only updates $\big(x_i, (y_p)_{e_p \in \mathcal{E}_i^{out}}, \lambda_i\big)$ and leaves local variables of other agents unchanged. Then, we define $\mathcal{S} := \text{Id} - \mathcal{T}$, and $\mathcal{S}_i := \text{Id} - \mathcal{T}_i$, which represents the "direction" of change for the global variable z when the corresponding operator is applied. We have

$$\mathcal{S}_i(z)_j := \begin{cases} z_j - \mathcal{T}(z)_j, & \text{if } \mathcal{I}^i_j = 1; \\ 0, & \text{if } \mathcal{I}^i_j = 0. \end{cases}$$

Therefore, without considering delay, the update performed by each agent i when active can be characterized by

$$z^{k+1} = z^k - \mathcal{S}_i(z^k) = \mathcal{T}_i(z^k).$$

In order to guarantee convergence, we introduce an extra step size $\eta_i \in (0, 1)$ such that instead of moving to $z^k - \mathcal{S}_i(z^k)$ directly after agent i's update, the global variable moves to

$$z^{k+1} = z^k - \eta_i \mathcal{S}_i(z^k),$$

which is an instance of the randomized block-coordinate fixed point iteration for the corresponding operator $\mathcal{T}$. The convergence results have been investigated in [11] when $\mathcal{T}$ is an averaged operator and in [22] when $\mathcal{T}$ is nonexansive. However, neither of those two cases consider information delays in a practical asynchronous setting. Instead, we resort to the recently development in [72, 71, 95], which not only deals with randomized block-coordinate update for nonexpansive operators, but also handles potentially out-of-date information.

Next, we characterize delayed information. At iteration k, let $\tau^k \in \mathbf{R}_+^L$ be the delay vector for variables x^k and λ_a^k, of which $\tau_i^k \in \mathbf{R}_+$ is the delay time for x_i^k and λ_i^k, $\forall i \in [L]$; let $\delta^k \in \mathbf{R}_+^M$ be the delay vector for variables y^k, of which $\delta_p^k \in \mathbf{R}_+$ is the delay time for y_p^k, $\forall p \in [M]$.

Assumption 7.2.2 (Uniform Delay Upper Bound) *There is an uniform upper bound of delay time $0 < \tau < +\infty$ at any iteration k such that $\tau_i^k \leq \tau$, $\forall i \in [L]$, and $\delta_p^k \leq \tau$, $\forall p \in [M]$.*

Remark 7.2.1 *Since $\left(x_i^k, (y_p^k)_{e_p \in \mathcal{E}_i^{out}}, \lambda_i^k\right)$ is agent i's local variables, they are assumed to share the same delay time, i.e., $\tau_i^k = \delta_p^k$, $\forall e_p \in \mathcal{E}_i^{out}$. The reason we introduce δ_p^k as the delay time of variable y^k is that as discussed before, agent i's local updates involve not only $(y_p^k)_{e_p \in \mathcal{E}_i^{out}}$, but also $(y_p^k)_{e_p \in \mathcal{E}_i^{in}}$, $\left((y_q)_{e_q \in \mathcal{E}_j^{out}}\right)_{j \in \mathcal{N}_i^{out}}$ and $\left((y_q)_{e_q \in \mathcal{E}_j^{in}}\right)_{j \in \mathcal{N}_i^{out}}$. The latter three are received from $\mathcal{N}_i$, thus have different delay times than τ_i^k in general. We keep both notations τ_i^k and δ_p^k so that so that the asynchronous distributed algorithm to be presented later has better clarity regarding delay times.*

Remark 7.2.2 *If agent i is activated at iteration k, it has the most up-to-date information of its own local variables, i.e., $x_i^{k-\tau_i^k} = x_i^{k-\tau_i^k+1} = \cdots = x_i^k$, $\lambda_i^{k-\tau_i^k} = \lambda_i^{k-\tau_i^k+1} = \cdots = \lambda_i^k$, and $y_p^{k-\delta_p^k} = y_p^{k-\delta_p^k+1} = \cdots = y_p^k$, $\forall e_p \in \mathcal{E}_i^{out}$, because the activation of agent i at iteration k implies that it is the k-th (most recent) agent completes an update.*

With the definitions of delay vectors, the delayed global variable used in iteration k can be defined as

$$\widehat{z}^k := (x^{k-\tau^k}, y^{k-\delta^k}, \lambda_a^{k-\tau^k}) \in \mathbf{R}^{n+m(M+L)}.$$

Then, the update applied to the global variable at iteration k with delayed information can be characterized as

$$z^{k+1} = z^k - \eta_{i_k}(\widehat{z}^k - \mathcal{T}_{i_k}(\widehat{z}^k)) = z^k - \eta_{i_k}\mathcal{S}_{i_k}(\widehat{z}^k), \tag{7.6}$$

where i_k is a random agent index number from $[L]$. Iteration (7.6) is an instance of randomized block-coordinate iteration of nonexpansive operator based on delayed variables [72, 71, 95]. With proper choices of step sizes, η_{i_k}, and some other assumptions to be stated next, the convergence of iteration (7.6) to fixed point of the nonexpansive operator $\mathcal{T}$, which is also optimal solution to original problem (2.1), can be guaranteed. Under Assumption 7.2.3, the relation between the delayed global variable $\widehat{z}^k$ and z^k can be characterized by the following proposition [72].

Proposition 7.2.1 *Under Assumption 7.2.2, it holds that*

$$\widehat{z}^k = z^k + \sum_{d \in J(k)} (z^d - z^{d+1}), \tag{7.7}$$

where $J(k) \subseteq \{k-\tau, \ldots, k-1\}$ is an index set.

Proof See [72, Section 1.2]. ■

Assumption 7.2.3 (Independent Random Agent Activation) *The probability that agent i is responsible for the increment of global iteration counter from k to $k+1$, i.e., agent i completes the k-th block coordinate update on global variable is Prob$(i_k = i \,|\, \mathcal{Z}^k)$ = Prob$(i_k = i) := p_i > 0$, $\forall i \in [L]$ and $\forall k$, where $\mathcal{Z}^k$ is the σ-algebra generated by z_0, $\widehat{z}^0, \ldots, z_k$, $\widehat{z}^k$. Let $p_{min} = \min_{i\in[L]} p_i$.*

As mentioned in [95], Prob$(i_k = i \mid \mathcal{Z}^k) =$ Prob$(i_k = i)$ implies that the agent responsible for the k-th block coordinate update is independent of of the delays in different rows of $\widehat{z}^k$, which may not be practical for all the cases, but is a key assumption for the convergence proof to go through.

Assumption 7.2.4 *It is assumed that each agent: 1) has dedicated local buffer to store required information from neighbors; 2) is frequently communicating with or probing neighboring agents, and whenever new information is received, corresponding local buffer is instantly updated; 3) sends out information to its appropriate neighbors instantly after it completes a new round of local update (there is no other agent completing local update before the current agent sends out updated information).*

The proposed *asynchronous dual consensus algorithm via operator augmentation with incidence matrix considering delays* is summarized in Algorithm 8, and its convergence result is stated in the following theorem.

Theorem 7.2.1 *Under Assumptions 2.1.1, 2.1.2, 2.1.3, 6.0.1, 6.2.1, 7.1.1, 7.2.1, 7.2.2, 7.2.3 and 7.2.4, suppose* σ, υ_i, $\forall i \in [L]$, *and* γ_p, $\forall p \in [M]$ *are chosen according to Proposition 7.1.1 such that matrix* $\widehat{P}$ *in* (7.1) *is positive definite. Suppose step sizes* η_i *satisfies that*

$$0 < \eta_i < \frac{p_{min}}{p_i \left(2\tau\sqrt{\kappa p_{min}} + \kappa\right)} = \eta_{i,max}, \quad \forall i \in [L],$$

where $\kappa := \frac{\lambda_{max}(\widehat{P})}{\lambda_{min}(\widehat{P})}$ *is the condition number of matrix* $\widehat{P}$. *Then, the sequence* $(z^k)_{k\geq 0}$ *generated by Algorithm 8 converges to a* z^**-valued random variable with probability 1, for some* $z^* = (x^*, y^*, \lambda_a^*) \in \mathrm{zer}(T_{\mathcal{L}_a})$ *with* $T_{\mathcal{L}_a}$ *being defined in* (6.13)*, and* x^* *is an optimal solution to the original problem* (2.1)*.*

Proof Under the independent random agent activation assumption, each iteration of Algorithm 8 can be characterized by (7.6), which is a special case of the randomized block-coordinate fixed point iteration of nonexpansive operator based on delayed variables. If η_i is smaller than $\eta_{i,max}$ as defined above, according to [95, Theorem 4], (7.6) will converge to a random variable supported by $z^* = (x^*, y^*, \lambda_a^*) \in \mathrm{zer}(T_{\mathcal{L}_a})$ almost surely. ■

Algorithm 8 Asynchronous Dual Consensus via Operator Augmentation with Incidence Matrix Considering Delays

1: Initialize $(x^0,\, y^0,\, \lambda^0)$, set $k = 0$;

2: **while** stopping criterion is not satisfied **do**

3: Agent $i \in [L]$ independently update its local variables using the most recent information received from its neighbors:

$$\text{Step 1: } d_{i,y}^k = \sum_{e_p \in \mathcal{E}_i} V_{ip} y_p^{k-\delta_p^k};$$

$$\widetilde{x}_i^k = \arg\min_{x_i \in X_i} \Big(f_i(x_i) + \big(\lambda_i^{k-\tau_i^k} + 2\sigma_i(A_i x_i^{k-\tau_i^k} - b_i + d_{i_k,y}^k)\big)^T A_i x_i + \frac{\alpha_i}{2}\|x_i - x_i^{k-\tau_i^k}\|^2 \Big);$$

$$\widetilde{\lambda}_i^k = \lambda_i^{k-\tau_i^k} + \sigma\Big(A_i x_i^{k-\tau_i^k} - b_i + d_{i_k,y}^k\Big);$$

$$d_{p,e}^k = \sum_{e_q \in \mathcal{N}_p^e} L_{pq}^e y_q^{k-\delta_q^k}, \quad \forall e_p \in \mathcal{E}_i^{out};$$

$$\widetilde{y}_p^k = y_p^{k-\delta_p^k} - \gamma_p(\lambda_j^{k-\tau_j^k} - \lambda_i^{k-\tau_i^k}) - 2\gamma_p\sigma(A_j x_j^{k-\tau_j^k} - b_j - A_i x_i^{k-\tau_i^k} + b_i) - 2\gamma_p\sigma d_{p,e}^k, \quad \forall e_p = (i,j) \in \mathcal{E}_i^{out};$$

$$\text{Step 2: } x_i^{k+1} = x_i^k + \eta_i(\widetilde{x}_i^k - x_i^k);$$

$$\lambda_i^{k+1} = \lambda_i^k + \eta_i(\widetilde{\lambda}_i^k - \lambda_i^k);$$

$$y_p^{k+1} = y_p^k + \eta_i(\widetilde{x}_i^k - x_i^k), \quad \forall e_p \in \mathcal{E}_i^{out};$$

4: Virtual global counter increases from k to $k+1$;

5: Agent i sends out $\big(A_i x_i^{k+1} - b_i, (y_p^{k+1})_{e_p \in \mathcal{E}_i^{out}},\, \lambda_i^{k+1}\big)$ and $(y_p^{k-\delta_p^k})_{e_p \in \mathcal{E}_i^{in}}$ to appropriate neighbors;

6: **end while**

Remark 7.2.3 *Several observations can be made regarding step sizes η_i's:*

1. *In order to correctly set the step size η_i, each agent i needs to be aware of its own activation probability p_i as it appears in the equation defining $\eta_{i,max}$, but this may not be practical for some applications.*

2. *In general, longer delays leads to smaller $\eta_{i,max}$, and slower convergence speed.*

3. *The more ill-conditioned the matrix $\widehat{P}$ is (smaller condition number κ), the larger $\eta_{i,max}$ is.*

4. *Let $j = \arg\min_{i\in[L]} p_i$, then larger p_j implies larger $\eta_{i,max}$, $\forall i \in [L]$ and $i \neq j$. In other words, the more frequent the least active agent performs updates, the more aggressive other agents can be.*

5. *Fixing the least active agent and its activation probability p_{min}, for other agents, the less active (smaller p_i) they are, the larger step size upper bounds they have.*

7.3 Asynchronous Parallel Coordinate Updates of Nonexpansive Operators with Uniform Step Size Upper Bound

ARock [72] is a general algorithmic framework for asynchronous parallel coordinate updates of nonexpansive operators in Euclidean norm, which also handles delayed information. The distributed asynchronous algorithm proposed in the previous section builds upon the theoretical results in [95], which is an extension of ARock to nonexpansive operators under the more general norm $\|\cdot\|_P$ where $P \succ 0$. Both algorithms in [72, 95] has the form

$$z^{k+1} = z^k - \eta_{i_k}\mathcal{S}_{i_k}(\widehat{z}^k) = z^k - \frac{\eta}{Lp_{i_k}}\mathcal{S}_{i_k}(\widehat{z}^k). \tag{7.8}$$

Obviously, each agent must be aware its activation probability p_i in order to correctly set local step size η_i. However, this assumption may not be practical or realistic for many problems unless all agents have uniform activation probability $1/L$. In ARock [72], the upper bound for η is $\frac{p_{min}}{2\tau\sqrt{p_{min}}+1}$, thus the upper bound for the step size of agent i is $\frac{p_{min}}{p_i\left(2\tau\sqrt{p_{min}}+1\right)}$.

In this section, we modify the ARock algorithm framework to the following

$$z^{k+1} = z^k - \eta\mathcal{S}_{i_k}(\widehat{z}^k), \tag{7.9}$$

where all agents utilize the same step size η. An uniform upper bound will be derived for η such that the almost sure convergence of the asynchronous parallel coordinate updates

algorithm to a random variable supported by some fixed point of the underlying nonexpansive operator $\mathcal{T}$ can still be guaranteed. This new algorithm (7.9) eliminates individual agent activation probability from the step size, thus is more practical for some applications. For all development in this section, we will assume operator $\mathcal{T}$ is nonexpansive on $\mathcal{H} = \mathbf{R}^{n+mL+mM}$ under the Euclidean norm instead of the norm $\|\cdot\|_P$.

We define the full update (all agents perform updates) at the k-th iteration:

$$\overline{z}^{k+1} =: z^k - \eta \mathcal{S}(\widehat{z}^k). \tag{7.10}$$

Notice that (7.10) is defined for the purpose of analysis only, and is never computed in the asynchronous setting. Recall the following result we reviewed in Chapter 3.

Lemma 7.3.1 *Operator $\mathcal{T} : \mathcal{H} \to \mathcal{H}$ is nonexpansive if and only if $\mathcal{S} = Id - \mathcal{T}$ is $1/2$-cocoercive, i.e., $\langle x - y, \mathcal{S}(x) - \mathcal{S}(y)\rangle \geq \frac{1}{2}\|\mathcal{S}(x) - \mathcal{S}(y)\|^2$, $\forall x, y \in \mathcal{H}$.*

Let $H \in \mathbf{R}^{n+mL+mM}$ be a diagonal matrix with diagonal elements defined as $H_{jj} = \frac{1}{p_i}$ if $\mathcal{I}_j^i = 1$, $\forall j \in [n + mL + mM]$. Due to the construction method of $\mathcal{I}_j^i$, all diagonal elements are positive, thus H is symmetric postive definite.

Proposition 7.3.1 *Let $(z^k)_{k\geq 0}$ be the sequence generated by iteration* (7.9)*. Then for any $z^* \in \mathrm{Fix}(\mathcal{T})$ and $\gamma > 0$, it holds that*

$$\begin{aligned}\mathbb{E}\left(\|z^{k+1} - z^*\|_H^2 \,|\, \mathcal{Z}^k\right) \leq \|z^k - z^*\|_H^2 + \gamma \sum_{d=k-\tau}^{k-1} \|z^d - z^{d+1}\|^2 \\ + \left(1 + \frac{|J(k)|}{\gamma} - \frac{1}{\eta}\right)\|z^k - \overline{z}^{k+1}\|^2.\end{aligned} \tag{7.11}$$

Proof We have

$$\begin{aligned}\mathbb{E}\left(\|z^{k+1} - z^*\|_H^2 \,|\, \mathcal{Z}^k\right) &\overset{(7.9)}{=} \mathbb{E}\left(\|z^k - \eta \mathcal{S}_{i_k}(z^k) - z^*\|_H^2 \,|\, \mathcal{Z}^k\right) \\ &= \|z^k - z^*\|_H^2 + \mathbb{E}\left(2\eta\langle S_{i_k}(\widehat{z}^k), z^* - z^k\rangle_H + \eta^2\|\mathcal{S}_{i_k}(\widehat{z}^k)\|_H^2 \,|\, \mathcal{Z}^k\right) \\ &= \|z^k - z^*\|_H^2 + \sum_{i=1}^{L} 2\eta p_i \langle S_i(\widehat{z}^k), z^* - z^k\rangle_H + \eta^2 \sum_{i=1}^{L} p_i \|\mathcal{S}_i(\widehat{z}^k)\|_H^2 \\ &= \|z^k - z^*\|_H^2 + 2\eta\langle S(\widehat{z}^k), z^* - z^k\rangle + \eta^2 \sum_{i=1}^{L} p_i \|\mathcal{S}_i(\widehat{z}^k)\|_H^2.\end{aligned} \tag{7.12}$$

Note that

$$\sum_{i=1}^{L} p_i \|\mathcal{S}_i(\widehat{z}^k)\|_H^2 = \|\mathcal{S}(\widehat{z}^k)\|^2 \overset{(7.10)}{=} \frac{1}{\eta^2}\|z^k - \overline{z}^{k+1}\|^2, \tag{7.13}$$

and

$$\begin{aligned}
\langle \mathcal{S}(\widehat{z}^k), z^* - z^k\rangle &\overset{(7.7)}{=} \left\langle \mathcal{S}(\widehat{z}^k), z^* - \widehat{z}^k + \sum_{d\in J(k)} (z^d - z^{d+1}) \right\rangle \\
&= \langle \mathcal{S}(\widehat{z}^k), z^* - \widehat{z}^k\rangle + \sum_{d\in J(k)} \langle \mathcal{S}(\widehat{z}^k), z^d - z^{d+1}\rangle \\
&\overset{(7.10)}{=} \langle \mathcal{S}(\widehat{z}^k) - \mathcal{S}(z^*), z^* - \widehat{z}^k\rangle + \frac{1}{\eta}\sum_{d\in J(k)} \langle z^k - \overline{z}^{k+1}, z^d - z^{d+1}\rangle \\
&\leq -\frac{1}{2}\|\mathcal{S}(\widehat{z}^k) - \mathcal{S}(z^*)\|^2 + \frac{1}{\eta}\sum_{d\in J(k)} \langle z^k - \overline{z}^{k+1}, z^d - z^{d+1}\rangle \\
&\leq -\frac{1}{2}\|\mathcal{S}(\widehat{z}^k) - \mathcal{S}(z^*)\|^2 + \frac{1}{2\eta}\sum_{d\in J(k)} \left(\frac{1}{\gamma}\|z^k - \overline{z}^{k+1}\|^2 + \gamma\|z^d - z^{d+1}\|^2\right) \\
&\overset{(7.10)}{=} -\frac{1}{2\eta^2}\|z^k - \overline{z}^{k+1}\|^2 + \frac{|J(k)|}{2\gamma\eta}\|z^k - \overline{z}^{k+1}\|^2 + \frac{\gamma}{2\eta}\sum_{d\in J(k)} \|z^d - z^{d+1}\|^2,
\end{aligned} \tag{7.14}$$

where the third equality is due to $\mathcal{S}(z^*) = z^* - \mathcal{T}(z)^* = 0$, for any $z^* \in \mathrm{Fix}(\mathcal{T})$; the first inequality is due to the $1/2$-cocoercivity of operator $\mathcal{S}$; the second inequality is because of the fact that $\frac{1}{\gamma}\|p\|^2 + \gamma\|q\|^2 \geq 2\langle p, q\rangle$, $\forall p, q \in \mathcal{H}$. Plugging (7.13) and (7.14) into (7.12) yields (7.11). ■

Let $\mathcal{H}^{\tau+1} = \prod_{i=0}^{\tau} \mathcal{H}$ be a product space. Then define a matrix $\widehat{U} \in \mathbf{R}^{(\tau+1)\times(\tau+1)}$ as:

$$\widehat{U} = \begin{bmatrix}
1+\tau & -\tau & & & & \\
-\tau & 2\tau-1 & 1-\tau & & & \\
 & 1-\tau & 2\tau-3 & 2-\tau & & \\
 & & \ddots & \ddots & \ddots & \\
 & & & -2 & 3 & -1 \\
 & & & & -1 & 1
\end{bmatrix},$$

and define $U = \widehat{U} \otimes H$, which is symmetric and positive definite since both $\widehat{U}$ and H are symmetric and positive definite. We define

$$\mathbf{z}^k := (z^k, z^{k-1}, \ldots, z^{k-\tau}) \in \mathcal{H}^{\tau+1}, \quad k \geq 0,$$

$$\mathbf{z}^* := 1_{\tau+1} \otimes z^* \in \mathcal{H}^{\tau+1},$$

where $z^k = z^0$ for $k < 0$. It can be verified that

$$\|\mathbf{z}^{k+1} - \mathbf{z}^*\|_U^2 = \|z^k - z^*\|_H^2 + \sum_{d=k-\tau}^{k-1} \big(d - (k-\tau) + 1\big)\|z^d - z^{d+1}\|_H^2. \tag{7.15}$$

Proposition 7.3.2 *Let $(z^k)_{k\geq 0}$ be the sequence generated by iteration* (7.9)*. Then for any $\mathbf{z}^* = 1_{\tau+1} \otimes z^* \in \mathcal{H}^{\tau+1}$ where $z^* \in \mathrm{Fix}(\mathcal{T})$, it holds that*

$$\mathbb{E}\left(\|\mathbf{z}^{k+1} - \mathbf{z}^*\|_U^2 \,|\, \mathcal{Z}^k\right) \leq \|\mathbf{z}^k - \mathbf{z}^*\|_U^2 + \left(\tau + 1 + p_{max}\tau - \frac{1}{\eta}\right)\|z^k - \overline{z}^{k+1}\|^2, \tag{7.16}$$

where $p_{max} = \max_{i\in[L]} p_i$.

Proof Let $\gamma = \frac{1}{p_{max}}$ in (7.11), we have

$$\begin{aligned}
&\mathbb{E}\left(\|\mathbf{z}^{k+1} - \mathbf{z}^*\|_U^2 \,|\, \mathcal{Z}^k\right) \\
&\overset{(7.15)}{=} \mathbb{E}\left(\|z^{k+1} - z^*\|_H^2 \,|\, \mathcal{Z}^k\right) + \sum_{d=k+1-\tau}^{k} \big(d - (k-\tau)\big)\mathbb{E}\left(\|z^d - z^{d+1}\|_H^2 \,|\, \mathcal{Z}^k\right) \\
&= \mathbb{E}\left(\|z^{k+1} - z^*\|_H^2 \,|\, \mathcal{Z}^k\right) + \sum_{d=k+1-\tau}^{k-1} \big(d - (k-\tau)\big)\|z^d - z^{d+1}\|_H^2 \\
&\qquad\qquad + \tau\mathbb{E}\left(\|z^k - z^{k+1}\|_H^2 \,|\, \mathcal{Z}^k\right) \\
&\overset{(7.9)}{=} \mathbb{E}\left(\|z^{k+1} - z^*\|_H^2 \,|\, \mathcal{Z}^k\right) + \sum_{d=k+1-\tau}^{k-1} \big(d - (k-\tau)\big)\|z^d - z^{d+1}\|_H^2 \\
&\qquad\qquad + \tau\eta^2\mathbb{E}\left(\|\mathcal{S}_{i_k}(\widehat{z}^k)\|_H^2 \,|\, \mathcal{Z}^k\right) \\
&\overset{(7.10)}{=} \mathbb{E}\left(\|z^{k+1} - z^*\|_H^2 \,|\, \mathcal{Z}^k\right) + \sum_{d=k+1-\tau}^{k-1} \big(d - (k-\tau)\big)\|z^d - z^{d+1}\|_H^2 + \tau\|z^k - \overline{z}^{k+1}\|^2 \\
&\leq \|z^k - z^*\|_H^2 + \left(\tau + 1 + p_{max}\tau - \frac{1}{\eta}\right)\|z^k - \overline{z}^{k+1}\|^2 + \frac{1}{p_{max}}\sum_{d=k-\tau}^{k-1}\|z^d - z^{d+1}\|^2 \\
&\qquad\qquad + \sum_{d=k+1-\tau}^{k-1} \big(d - (k-\tau)\big)\|z^d - z^{d+1}\|_H^2 \\
&\leq \|z^k - z^*\|_H^2 + \left(\tau + 1 + p_{max}\tau - \frac{1}{\eta}\right)\|z^k - \overline{z}^{k+1}\|^2 + \sum_{d=k-\tau}^{k-1}\|z^d - z^{d+1}\|_H^2
\end{aligned}$$

$$
\begin{aligned}
&\qquad\qquad + \sum_{d=k+1-\tau}^{k-1} \big(d-(k-\tau)\big)\|z^d - z^{d+1}\|_H^2 \\
&= \|z^k - z^*\|_H^2 + \sum_{d=k-\tau}^{k-1} \big(d-(k-\tau)+1\big)\|z^d - z^{d+1}\|_H^2 \\
&\qquad\qquad + \left(\tau + 1 + p_{max}\tau - \frac{1}{\eta}\right)\|z^k - \overline{z}^{k+1}\|^2 \\
&\overset{(7.15)}{=} \|\mathbf{z}^k - \mathbf{z}^*\|_U^2 + \left(\tau + 1 + p_{max}\tau - \frac{1}{\eta}\right)\|z^k - \overline{z}^{k+1}\|^2,
\end{aligned}
$$

where the first inequality is due to the plugging in of inequality (7.11) and the fact that $|J(k)| \leq \tau$; the second inequality is due to the fact that $H \succeq \frac{1}{p_{max}} I_{n+mL+mM}$. Therefore, inequality (7.16) holds. ■

Remark 7.3.1 *Suppose*

$$0 < \eta < \frac{1}{\tau(1+p_{max})+1} = \eta_{max}, \tag{7.17}$$

then we have

$$\mathbb{E}\left(\|\mathbf{z}^{k+1} - \mathbf{z}^*\|_U^2 \mid \mathcal{Z}^k\right) \leq \mathbb{E}\left(\|\mathbf{z}^k - \mathbf{z}^*\|_U^2 \mid \mathcal{Z}^k\right) = \|\mathbf{z}^k - \mathbf{z}^*\|_U^2,$$

i.e., the sequence $(\mathbf{z}^k)$ *is stochastic Fejer monotone. Notice that* $\mathbb{E}\left(\|\mathbf{z} - \mathbf{z}^*\|_U^2 \mid \mathcal{Z}^k\right)$ *can also be thought of as a Lyapunov function from the stability of a dynamical system's perspective.*

Theorem 7.3.1 *Suppose* $\mathcal{T}$ *is a nonexpansive operator under the Euclidean norm. Under Assumptions 7.2.2 and 7.2.3, the sequence* $(z^k)_{k\geq 0}$ *generated by iteration* (7.9)*, with* η *satisfying* (7.17) *converges to a* $\mathrm{Fix}(\mathcal{T})$*-valued random variable almost surely.*

Proof Notice that condition (7.17) guarantees that $\left(\tau + 1 + p_{max}\tau - \frac{1}{\eta}\right) < 0$, then follow the same argument in [72, Lemma 12 & 13, Theorem 14]. ■

In general, smaller delay upper bound τ leads to larger η_{max}. In the extreme case where there is no delay, i.e., $\tau = 0$, the uniform step size upper bound reduces to $\eta_{max} = 1$, and the global variable update becomes $z^{k+1} = z^k - \eta\mathcal{S}_{i_k}(z^k)$, for $\eta \in (0,1)$, which is exactly the randomized Krasnosel'skii Mann iterations of the averaged operator $(1-\eta)\mathrm{Id} + \eta\mathcal{T}$ studied

in [11, Theorem 3]. In contrast, the upper bound for agent i's step size in ARock reduces to $p_{min}/p_i \leq 1$ when $\tau = 0$, which is more conservative than ours. Another comment regarding the bound in (7.17) is that the smaller $p_{max} \in [1/L, 1)$ is, the larger η_{max} is. This implies that roughly speaking, less discrepancies in agents' activation probability, i.e., smaller variance of p_i's distribution, allows larger step sizes.

The investigation of (7.9) in the scenario where operator $\mathcal{T}$ is a nonexpansive operator with respect to the norm $\|\cdot\|_P$ is left as future work.

8. NUMERICAL EXAMPLES

In this chapter, we will test the effectiveness and performance of the proposed distributed algorithms through several numerical examples: exchange problem, L_1 regularized exchange problem, and resource allocation problem. For algorithms involved with multiplier graph $\mathcal{G}_\lambda$ (Algorithm 4-8), three possible graph topologies might be used: fully connected network, star network, and ring network, see Fig. 8.1 (a representation of the graph structures, and does not indicate the exact number of agents used in different numerical examples).

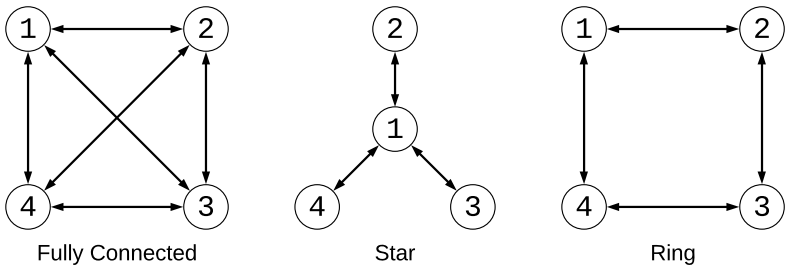

Fig. 8.1.: Graph topologies

When investigating the performance of a particular algorithm under different graph topologies, the weighted adjacency matrix W or the incidence matrix V are always first normalized such that the average weighted degree $\frac{1}{L}\sum_i^L \deg(i)$ is the same with different graph topologies.

8.1 Exchange Problem

Firstly, we study the commodity exchange problem. We consider a network of L agents exchanging m goods, and the local objective function or each agent is a quadratic function:

$$\underset{x}{\text{minimize}} \quad \sum_{i=1}^{L} \|C_i x_i - d_i\|^2 \tag{8.1}$$

$$\text{subject to} \quad \sum_{i=1}^{L} x_i = 0,$$

where $x_i \in \mathbf{R}^{n_i}$, $C_i \in \mathbf{R}^{p \times n_i}$, $d_i \in \mathbf{R}^p$ and $p < n_i$. The above problem is in the standard formulation (2.1) with $f_i(x_i) = \|C_i x_i - d_i\|^2 = x_i^T C_i^T C_i x_i - 2 d_i^T C_i x_i + d_i^T d_i$, $X_i = \mathbf{R}^{n_i}$, $A_i = I_{n_i}$ and $b_i = 0$. Apparently, for this particular problem, we have $m = n_i$, $\forall i \in [L]$. Since $p < n_i$, we have $C_i^T C_i \succeq 0$. Therefore, the global objective function $\sum_{i=1}^{L} \|C_i x_i - d_i\|^2$ is a convex function (but not strictly or strongly convex) in x.

The matrices and vectors in the above problem are randomly generated as follows. Firstly, randomly generate C_i's from normal distribution, then randomly generate $x_i^*, \forall i \in [L-1]$ from normal distribution and compute $x_L^* = -\sum_{i=1}^{L-1} x_i^*$ and $d_i = C_i x_i^*, \forall i \in [L]$. It is easy to see for any problem generated in this fashion, $x^* = (x_i^*)_{i \in [L]}$ is an optimal solution with optimal objective function value being 0. We choose $L = 20$, $n_i = m = 50$, $p = 20$, all the parameters and initial values are generated once and stored, then used under different simulation runs.

First, we test the proximal parallel ADMM algorithm (Algorithm 1) and the dual averaging algorithms via Douglas-Rachford splitting (Algorithm 2 and 3). The algorithm design parameters are manually chosen based on trial and error. For Algorithm 1, parameters are chosen from $\rho \in \{1,\, 5,\, 10\}$ and $\phi_i \in \{4\rho,\, 10\rho,\, 20\rho,\, 50\rho\}$, and it turns out $\rho = 5$, $\varphi_i = 20$ give the fastest convergence rates. Notice that $\varphi_i = 20$ is smaller than the bound provided in Proposition 5.3.3, which means that the sufficient conditions therein are conservative. For Algorithm 2 & 3, parameters are chosen from $\alpha \in \{0.05,\, 0.5,\, 0.95\}$ and $\beta \in \{0.1,\, 1,\, 10,\, 100\}$, and it turns out $\alpha = 0.5$, $\beta = 10$ is "optimal".

The global objective function values $\sum_{i=1}^{L} \|C_i x_i^k - d_i\|^2$ as well as the coupling constraint violations $\|\sum_{i=1}^{L} x_i^k\|$ after the first 800 iterations are plotted in Fig. 8.2. For Algorithm 2 and 3, since they are essentially two generalized resolvent iterations that originated from the same averaged operator, they generate the exact same variable values at each iteration k. Notice that the convergence speed of the dual averaging algorithms via Douglas-Rachford splitting is significantly faster than proximal parallel ADMM algorithm. This is possibly due to the fact that Algorithm 2 & 3 introduce more auxiliary variables (copies of dual

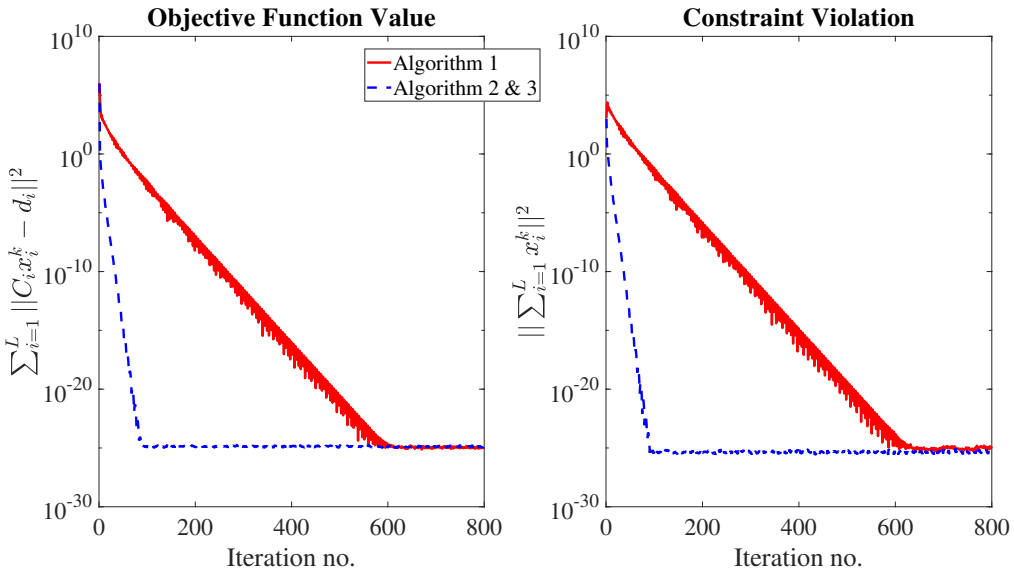

Fig. 8.2.: Exchange problem: convergence of Algorithm 1, 2 and 3

variables at each agent) which bring higher computation costs in each iteration, thus smaller number of iterations compared to Algorithm 1 are needed to reach the same accuracy level.

8.2 L_1-regularized Exchange Problem

We add an extra L_1 regularization term to the objective function of the exchange problem in (8.1) to impose sparsity in the optimal solution:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \sum_{i=1}^{L} \|C_i x_i - d_i\|^2 + \|x\|_1 \\ \text{subject to} \quad & \sum_{i=1}^{L} x_i = 0. \end{aligned} \tag{8.2}$$

Since the regularization term is separable, i.e., $\|x\|_1 = \sum_{i=1}^{L} \|x_i\|_1$, the above formulation can be equivalently written as

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \sum_{i=1}^{L} \left(\|C_i x_i - d_i\|^2 + \|x_i\|_1\right) \\ \text{subject to} \quad & \sum_{i=1}^{L} x_i = 0, \end{aligned} \tag{8.3}$$

which is a special case of the formulation in (6.19). Although this problem can be solved directly using Algorithms 1-5, as discussed in Section 6.3, it is expected that local optimization problem $\min_{x_i} (\|C_i x_i - d_i\|^2 + \|x_i\|_1 + \text{another quadratic function})$, may be the bottleneck of the algorithm because it does not have an explicit analytical solution.

On the other hand, if we apply Algorithm 6 to this problem, the local optimization problem will be broken into two steps: an unconstrained quadratic program and an evaluation of the proximal operator of L_1 norm function, both of which have explicit analytical solutions. Particularly, the proximal operator of the L_1 norm function at $y \in \mathbf{R}^n$ can be evaluated elementwise as

$$\left(\underset{\beta\|\cdot\|_1}{\text{prox}}(y)\right)_j = \text{sign}\,(y_j)\max\left(|y_j| - \beta, 0\right), \quad \forall j \in [n],$$

which is also called *soft thresholding*. Therefore, by applying Algorithm 6, local updates of all agents become linear expressions, which will be efficient to compute.

We choose $L = 20$, $n_i = m = 50$, $p = 20$. The parameter matrices are generated similarly as before, which are stored and used under different simulation runs. We firstly solve optimization problem (8.2) centrally and the optimal objective function value turns out to be 65.9877. All three graph topologies in Fig. 8.1 are utilized and compared under the same set of parameter values: $\upsilon_i = 10$, $\gamma_i = \sigma_i = \frac{1}{1.05\deg(i)} < \frac{1}{\deg(i)}$, $\forall i$. The global objective function values $\sum_{i=1}^{L} (\|C_i x_i - d_i\|^2 + \|x_i\|_1)$ as well as the coupling constraint violations $\|\sum_{i=1}^{L} x_i^k\|$ after the first 800 iterations are plotted in Fig. 8.3. Algorithm 6 converges to an optimal solution (with objective function value 65.9877) under all three graph topologies. As expected, the fully connected network renders the fastest convergence speed while the ring network gives slowest convergence speed under the same set of algorithm parameters and average node degree, but the convergence speed difference between fully connected network and star network is minimal.

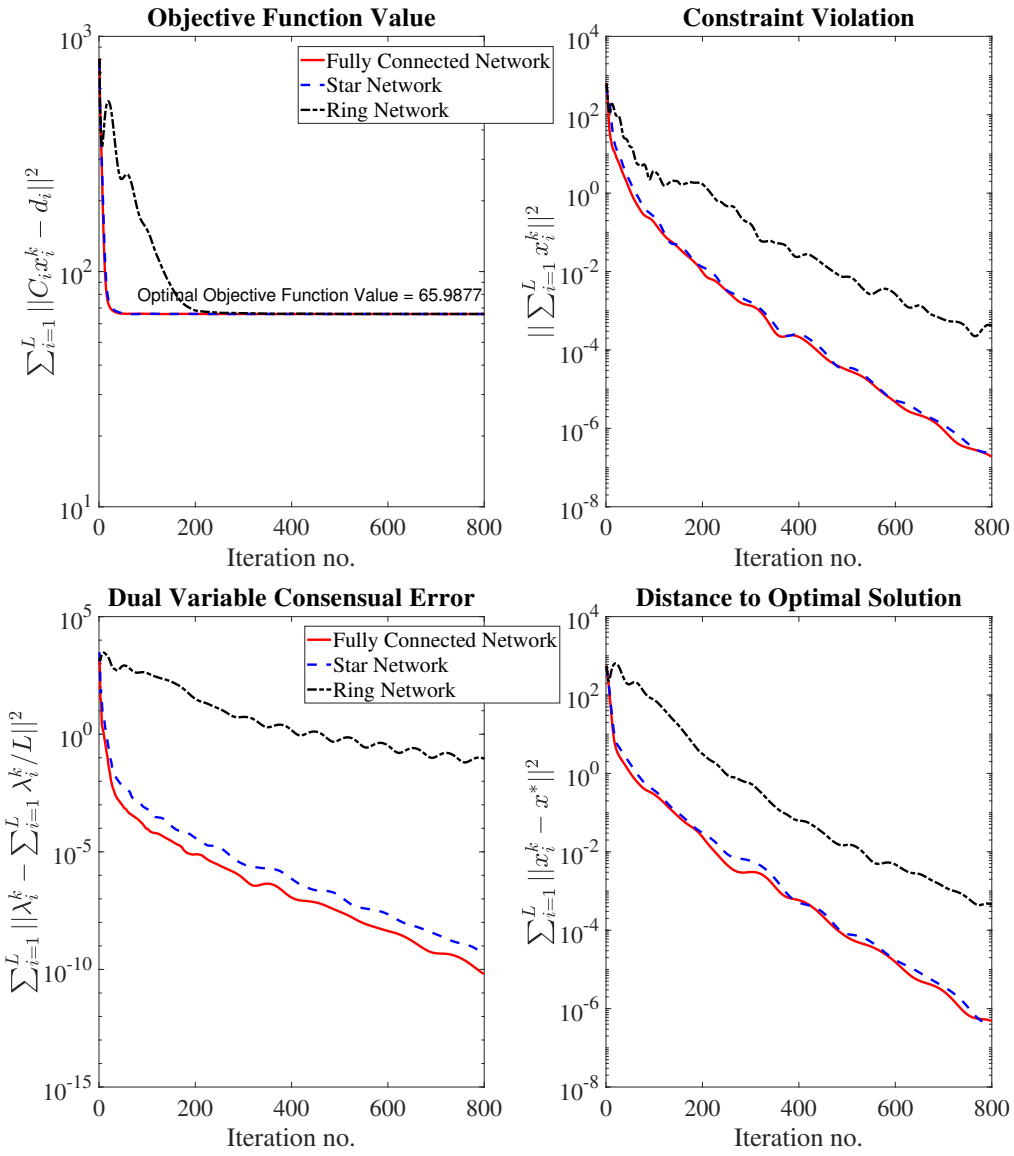

Fig. 8.3.: L_1 regularized exchange problem: convergence of Algorithm 6

8.3 Resource Allocation Problem

Modifying the global coupling constraint of the exchange problem in (8.1), we obtain the following resource allocation problem:

$$\underset{x}{\text{minimize}} \quad \sum_{i=1}^{L} \|C_i x_i - d_i\|^2 \tag{8.4}$$

$$\text{subject to} \quad \sum_{i=1}^{L} \left(x_i^T x_i - 2 \cdot 1_{n_i}^T x_i + n_i \right) \leq 360,$$

which is a special case of the convex inequality coupling constraints formulation in (2.6), with $g_i(x_i) = x_i^T x_i - 2 \cdot 1_{n_i}^T x_i + n_i = \|x_i - 1_{n_i}\|^2$, and $b_i = 360/L$. The problem in (8.4) is equivalent to the following problem

$$\begin{aligned} \underset{x,\, y}{\text{minimize}} \quad & \sum_{i=1}^{L} \|C_i x_i - d_i\|^2 \\ \text{subject to} \quad & \sum_{i=1}^{L} y_i = 360, \\ & \|x_i - 1_{n_i}\|^2 \leq y_i, \quad \forall i \in [L], \end{aligned} \tag{8.5}$$

where (x_i, y_i) is the local decision variable for agent i,

We will apply dual consensus algorithms via operator augmentation with graph Laplacian matrix (Algorithm 4) and incidence matrix (Algorithm 5) to solve (8.4). For Algorithm 4, all three multiplier graph topologies in Fig. 8.1 will be utilized and compared under the same set of parameters: $\upsilon_i = 1$, $\gamma_i = \sigma_i = \frac{1}{2.1 \deg(i)} < \frac{1}{2 \deg(i)}$, $\forall i \in [L]$. However, for Algorithm 5, only the star network and ring network will be utilized and compared under the same set of parameters: $\upsilon_i = 1$, $\sigma_i = \frac{1}{2.1 \deg(i)} < \frac{1}{2 \deg(i)}$, $\forall i \in [L]$, and $\gamma_p = 0.45 < 0.5$, $\forall p \in [M]$, because of the reason discussed in Remark 6.2.4. We choose $L = 20$, $n_i = m = 50$, $p = 20$. The parameter matrices are generated similarly as before, which are stored and used under different simulation runs. We firstly solve optimization problem (8.4) centrally and the optimal objective function value turns out to be 73.2774.

The global objective function values $\sum_{i=1}^{L} (\|C_i x_i - d_i\|^2)$, the "resource" constraint violations $\| \sum_{i=1}^{L} y_i^k - 360\|^2$, the dual variables consensual error $\sum_{i=1}^{L} \|\lambda_i^k - \sum_{i=1}^{L} \lambda_i^k / L\|^2$ and the distance to an optimal solution $\sum_{i=1}^{L} \|x_i^k - x_i^*\|^2$ after the first 100 iterations are plotted in Fig. 8.4 (Algorithm 4) and Fig. 8.5 (Algorithm 5). Algorithm 4 converges to an optimal solution with all three graph topologies. The fully connected network and star network render similar convergence speed while the ring network is slower under the same set of algorithm parameters and average node degree. Similarly, Algorithm 5 also converges

to an optimal solution with the two graph topologies utilized and the convergence speed under star network is faster.

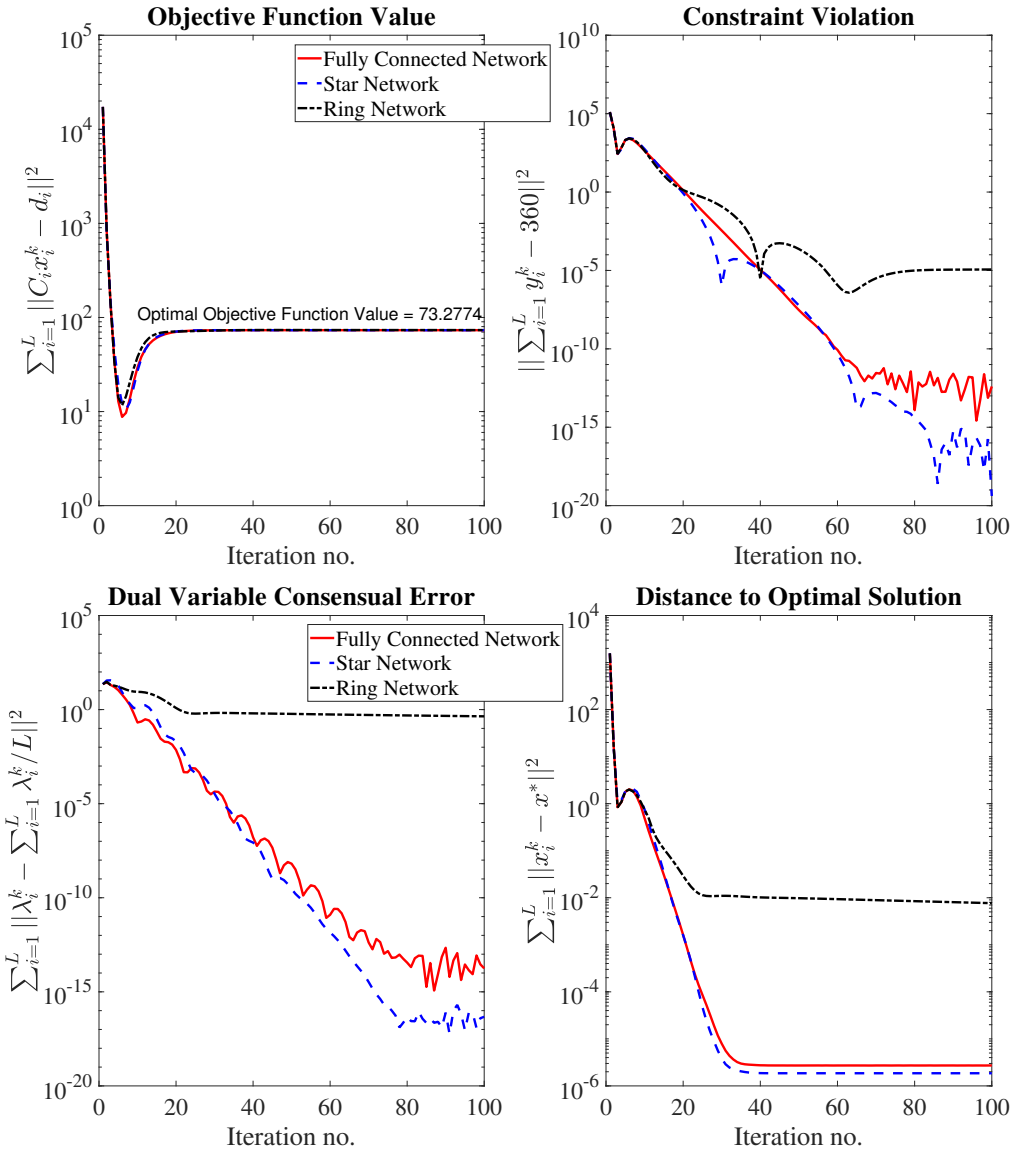

Fig. 8.4.: Resource allocation problem: convergence of Algorithm 4

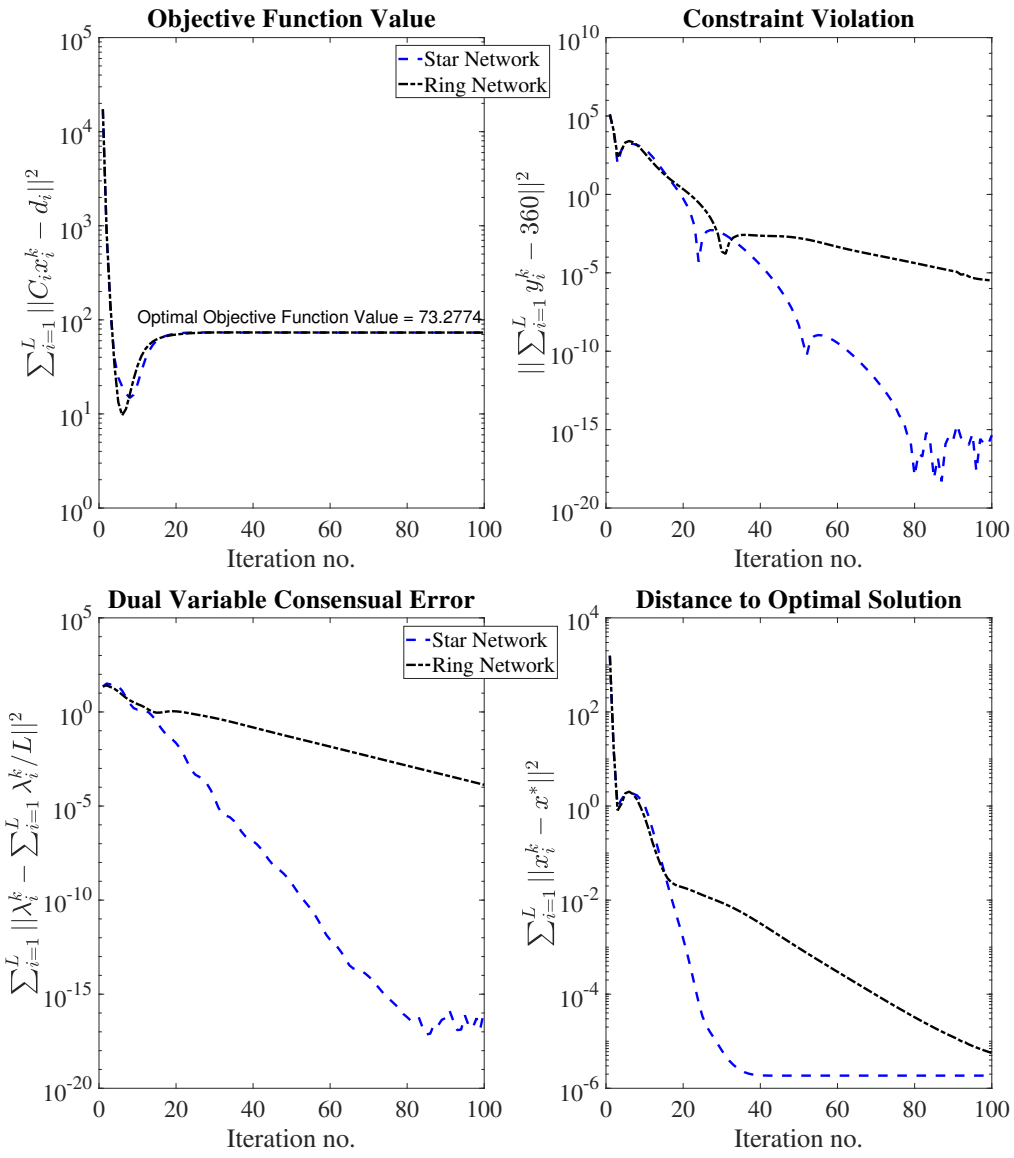

Fig. 8.5.: Resource allocation problem: convergence of Algorithm 5

9. CONCLUSION

The distributed solutions for a general class of convex multi-agent optimization problems with global shared resource couplings are studied in this dissertation. A series of synchronous and asynchronous distributed solution algorithms are proposed. Different from most of the distributed optimization algorithms in literature, our proposed algorithms are designed following a common procedure. Firstly, the optimal solution set of the multi-agent optimization problem is characterized by the zero sets of certain maximally monotone operators. Secondly, the equivalence of these zero sets with the fixed point sets of certain nonexpansive, or averaged operators are established. Then, iterative distributed algorithms are derived from (generalized) fixed point iterations of these nonexpansive, averaged operators. The most obvious advantage of this algorithm design procedure is that convergence to some optimal solution is automatically established when the algorithm is derived, while the biggest challenge being finding the proper operators that not only encapsulate the optimal solution set, but also whose fixed point iterations can be carried out in a distributed fashion. We summarize and highlight the main contributions of this dissertation before discussing some future directions.

Main Contributions

1. The convex multi-agent optimization problem formulation proposed is very general that many other formulations studied in the literature can be treated as special cases of our formulation, which not only naturally deals with the scenario where the local decision variables of different agents are of different dimensions, but also handles general convex shared resource coupling constraints.

2. The algorithms proposed in this dissertation require rather mild assumptions: local objective functions and local coupling functions are CCP, local constraint sets are

closed, convex and nonempty, there exists at least one point that satisfies that KKT conditions. No assumption needs to be made on the differentiability of local objective functions. Furthermore, we do not assume the boundness of agents' local constraint sets X_i's, which is a key assumption for many existing algorithms to guarantee the boundness of subgradients in their analysis.

3. All the algorithms proposed utilize fixed step sizes as opposed to diminishing step sizes, which are ubiquitous in literature. Diminishing step sizes are more difficult to tune from the perspective of achieving better convergence speed.

4. Since the algorithmic framework proposed in this dissertation is the fixed point iteration of certain nonexpansive, averaged operators, the adaptation of certain algorithm from a synchronous setting to an asynchronous setting with delayed information is achieved in Chapter 7. Many existing algorithms in the literature are based on duality theory, and it is not apparent how they can be modified to asynchronous algorithms with delays.

5. Our algorithmic framework provides better flexibility when the local objective function of an agent contains both smooth and nonsmooth parts. By applying operator splitting methods, the local constrained optimization problem can often be further broken down into simpler steps involving smooth part and nonsmooth part separately, e.g., Algorithm 6 in Chapter 6.

6. Generalized resolvent operator and generalized resolvent iteration are presented and studied. Although it is not exactly the first time they appear in literature, to the best our knowledge, it is the first time DR and DY splitting methods have been modified to utilize generalized resolvent operators.

7. The proximal parallel ADMM algorithm (Algorithm 1) proposed in Chapter 5 is an extension of the popular standard Gauss-Seidel (sequential) two-block ADMM algorithm to the Jacobi (parallel) multi-block scenario, which will be particularly useful for many large-scale classical machine learning training problems.

8. The general algorithmic framework for asynchronous parallel coordinate updates of nonexpansive operators proposed in [72] is modified to a more practical updating rule, and a new upper bound for the single relaxation step size is derived. The proposed modification eliminates individual agent activation probabilities from their step sizes.

Future Work

1. Most of the algorithms proposed in this dissertation are special cases of generalized resolvent iteration, or (generalized) DR splitting. The convergence orders of standard resolvent iteration (proximal point algorithm) and standard DR splitting method have been studied in [41, 24, 35], just to name a few. However, little work has been done to characterize the convergence rates of generalized resolvent operator and generalized DR splitting methods, or how to optimally select design parameter values, i.e., the P matrix associated with the generalized resolvent operator.

2. The multiplier graph is proposed in Chapter 6 over which agents can communicate and exchange information with neighbors. The convergence of subsequent algorithms under different graph topologies were compared in Chapter 8 through simulations, it will be interesting to quantitatively characterize how graph properties and network topologies affect the convergence speed. Currently, the multiplier graph is assumed to be time invariant. It would be interesting to study how existing algorithm framework can be modified to handle time-varying graphs.

3. In Chapter 7, we modified the asynchronous parallel coordinate updates algorithm proposed in [72] to allow a single step size to be used by all agents and eliminate the involvement of individual agent activation probabilities in their step sizes. A new uniform upper bound for the step size is derived with which probabilistic convergence to some fixed point can be guaranteed when the underlying operator is nonexpansive with respect to the Euclidean norm. The natural next step would be extending this result to the case when the underlying operator is nonexpansive with respect to the P-norm (defined in Chapter 4).

REFERENCES

REFERENCES

[1] A. Agarwal, M. J. Wainwright, and J. C. Duchi. Distributed dual averaging in networks. In *Advances in Neural Information Processing Systems*, pages 550–558, 2010.

[2] R. Aharoni and Y. Censor. Block-iterative projection methods for parallel computation of solutions to convex feasibility problems. *Linear Algebra and Its Applications*, 120:165–175, 1989.

[3] B. Anderson, S. Mou, A. S. Morse, and U. Helmke. Decentralized gradient algorithm for solution of a linear equation. *arXiv preprint arXiv:1509.04538*, 2015.

[4] I. Atzeni, L. G. Ordóñez, G. Scutari, D. P. Palomar, and J. R. Fonollosa. Demand-side management via distributed energy generation and storage optimization. *IEEE Transactions on Smart Grid*, 4(2):866–876, 2013.

[5] J.-B. Baillon and G. Haddad. Quelques propriétés des opérateurs angle-bornés etn-cycliquement monotones. *Israel Journal of Mathematics*, 26(2):137–150, 1977.

[6] S. Banach. Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fund. math*, 3(1):133–181, 1922.

[7] H. H. Bauschke and J. M. Borwein. On projection algorithms for solving convex feasibility problems. *SIAM review*, 38(3):367–426, 1996.

[8] H. H. Bauschke and P. L. Combettes. The baillon-haddad theorem revisited. *arXiv preprint arXiv:0906.0807*, 2009.

[9] H. H. Bauschke, P. L. Combettes, et al. *Convex analysis and monotone operator theory in Hilbert spaces*, volume 408. Springer, 2011.

[10] A. Beck, A. Nedic, A. Ozdaglar, and M. Teboulle. Optimal distributed gradient methods for network resource allocation problems. *IEEE Transactions on Control of Network Systems*, 1(1):64–74, 2014.

[11] P. Bianchi, W. Hachem, and F. Iutzeler. A coordinate descent primal-dual algorithm and application to distributed asynchronous optimization. *IEEE Transactions on Automatic Control*, 61(10):2947–2957, 2016.

[12] P. Bianchi and J. Jakubowicz. Convergence of a multi-agent projected stochastic gradient algorithm for non-convex optimization. *IEEE Transactions on Automatic Control*, 58(2):391–405, 2013.

[13] D. Blatt and A. O. Hero. Energy-based sensor network source localization via projection onto convex sets. *IEEE Transactions on Signal Processing*, 54(9):3614–3619, 2006.

[14] 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*, 3(1):1–122, 2011.

[15] H. Brézis and P. L. Lions. Produits infinis de résolvantes. *Israel Journal of Mathematics*, 29(4):329–345, 1978.

[16] Y. Cao, W. Yu, W. Ren, and G. Chen. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial informatics*, 9(1):427–438, 2013.

[17] S.-s. Chang, J. Kim, and X. Wang. Modified block iterative algorithm for solving convex feasibility problems in banach spaces. *Journal of Inequalities and Applications*, 2010(1):869684, 2010.

[18] T.-H. Chang. A proximal dual consensus admm method for multi-agent constrained optimization. *IEEE Transactions on Signal Processing*, 64(14):3719–3734, 2016.

[19] C. Chen, B. He, Y. Ye, and X. Yuan. The direct extension of admm for multi-block convex minimization problems is not necessarily convergent. *Mathematical Programming*, 155(1-2):57–79, 2016.

[20] P. Combettes. The convex feasibility problem in image recovery. *Advances in imaging and electron physics*, 95:155–270, 1996.

[21] P. L. Combettes*. Solving monotone inclusions via compositions of nonexpansive averaged operators. *Optimization*, 53(5-6):475–504, 2004.

[22] P. L. Combettes and J.-C. Pesquet. Stochastic quasi-fejér block-coordinate fixed point iterations with random sweeping. *SIAM Journal on Optimization*, 25(2):1221–1248, 2015.

[23] P. L. Combettes and I. Yamada. Compositions and convex combinations of averaged nonexpansive operators. *Journal of Mathematical Analysis and Applications*, 425(1):55–70, 2015.

[24] D. Davis. Convergence rate analysis of the forward-douglas-rachford splitting scheme. *SIAM Journal on Optimization*, 25(3):1760–1786, 2015.

[25] D. Davis and W. Yin. A three-operator splitting scheme and its optimization applications. *Set-Valued and Variational Analysis*, 25(4):829–858, 2017.

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

[27] J. C. Duchi, A. Agarwal, and M. J. Wainwright. Dual averaging for distributed optimization: Convergence analysis and network scaling. *IEEE Transactions on Automatic control*, 57(3):592–606, 2012.

[28] H. Everett III. Generalized lagrange multiplier method for solving problems of optimum allocation of resources. *Operations research*, 11(3):399–417, 1963.

[29] A. Falsone, K. Margellos, S. Garatti, and M. Prandini. Dual decomposition for multi-agent distributed optimization with coupling constraints. *Automatica*, 84:149–158, 2017.

[30] D. G. Feingold and R. Varga. Block diagonally dominant matrices and generalizations of the gerschgorin circle theorem. *Pacific Journal of Mathematics*, 12(4):1241–1250, 1962.

[31] D. Fullmer, J. Liu, and A. S. Morse. An asynchronous distributed algorithm for computing a common fixed point of a family of paracontractions. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pages 2620–2625. IEEE, 2016.

[32] D. Fullmer, L. Wang, and A. S. Morse. A distributed algorithm for computing a common fixed point of a family of paracontractions. *IFAC-PapersOnLine*, 49(18):552–557, 2016.

[33] D. Gabay. Applications of the method of multipliers to variational inequalities, in,(1983), 299. doi: 10.1016. *S0168-2024 (08)*, pages 70034–1, 1983.

[34] D. Gabay and B. Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & Mathematics with Applications*, 2(1):17–40, 1976.

[35] P. Giselsson and S. Boyd. Linear convergence and metric selection for douglas-rachford splitting and admm. *IEEE Transactions on Automatic Control*, 62(2):532–544, 2017.

[36] R. Glowinski and A. Marroco. Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de dirichlet non linéaires. *Revue française d'automatique, informatique, recherche opérationnelle. Analyse numérique*, 9(R2):41–76, 1975.

[37] C. Gu, Z. Wu, J. Li, and Y. Guo. Distributed convex optimization with coupling constraints over time-varying directed graphs. *arXiv preprint arXiv:1805.07916*, 2018.

[38] D. Han and X. Yuan. A note on the alternating direction method of multipliers. *Journal of Optimization Theory and Applications*, 155(1):227–238, 2012.

[39] B. He, L. Hou, and X. Yuan. On full jacobian decomposition of the augmented lagrangian method for separable convex programming. *SIAM Journal on Optimization*, 25(4):2274–2312, 2015.

[40] B. He, M. Tao, and X. Yuan. Alternating direction method with gaussian back substitution for separable convex programming. *SIAM Journal on Optimization*, 22(2):313–340, 2012.

[41] B. He and X. Yuan. On the convergence rate of douglas–rachford operator splitting method. *Mathematical Programming*, 153(2):715–722, 2015.

[42] A. O. Hero and D. Blatt. Sensor network source localization via projection onto convex sets (pocs). In *Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on*, volume 3, pages iii–689. IEEE, 2005.

[43] S. Hosseini, A. Chapman, and M. Mesbahi. Online distributed admm via dual averaging. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pages 904–909. IEEE, 2014.

[44] X. Hou, Y. Xiao, J. Cai, J. Hu, and J. Braun. A distributed model predictive control approach for optimal coordination of multiple thermal zones in a large open space. In *Proc. 4th International High Performance Buildings Conference*, pages 160–169, 2016.

[45] X. Hou, Y. Xiao, J. Cai, J. Hu, and J. E. Braun. Distributed model predictive control via proximal jacobian admm for building control applications. In *American Control Conference (ACC), 2017*, pages 37–43. IEEE, 2017.

[46] X. Hou, Y. Xiao, J. Joe, J. Cai, P. Karava, J. Hu, and J. Braun. An agent-based control implementation for the coordination of multiple rooftop units. In *Proc. 6th International High Performance Buildings Conference*, pages 324–333, 2018.

[47] Z. John Lu. The elements of statistical learning: data mining, inference, and prediction. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 173(3):693–694, 2010.

[48] M. Kraning, E. Chu, J. Lavaei, and S. Boyd. Message passing for dynamic network energy management. *arXiv preprint arXiv:1204.1106*, 2012.

[49] M. A. Krasnosel'skii. Two remarks on the method of successive approximations. *Uspekhi Matematicheskikh Nauk*, 10(1):123–127, 1955.

[50] S. Lee and A. Nedic. Distributed random projection algorithm for convex optimization. *IEEE Journal of Selected Topics in Signal Processing*, 7(2):221–229, 2013.

[51] S. Lee and A. Nedić. Asynchronous gossip-based random projection algorithms over networks. *IEEE Transactions on Automatic Control*, 61(4):953–968, 2016.

[52] V. Lesser, C. L. Ortiz Jr, and M. Tambe. *Distributed sensor networks: A multiagent perspective*, volume 9. Springer Science & Business Media, 2012.

[53] X. Li, X. Yi, and L. Xie. Distributed online optimization for multi-agent networks with coupled inequality constraints. *arXiv preprint arXiv:1805.05573*, 2018.

[54] P. Lin, W. Ren, and Y. Song. Distributed multi-agent optimization subject to nonidentical constraints and communication delays. *Automatica*, 65:120–131, 2016.

[55] I. Lobel and A. Ozdaglar. Distributed subgradient methods for convex optimization over random networks. *IEEE Transactions on Automatic Control*, 56(6):1291, 2011.

[56] I. Lobel, A. Ozdaglar, and D. Feijer. Distributed multi-agent optimization with state-dependent communication. *Mathematical programming*, 129(2):255–284, 2011.

[57] C. Lopes and A. H. Sayed. Distributed processing over adaptive networks. In *Proc. adaptive sensor array processing workshop*, pages 1–5, 2006.

[58] W. R. Mann. Mean value methods in iteration. *Proceedings of the American Mathematical Society*, 4(3):506–510, 1953.

[59] B. Martinet. Détermination approchée dâĂŹun point fixe dâĂŹune application pseudo-contractante. *CR Acad. Sci. Paris*, 274(2):163–165, 1972.

[60] M. Mesbahi and M. Egerstedt. *Graph theoretic methods in multiagent networks*, volume 33. Princeton University Press, 2010.

[61] S. Mou, J. Liu, and A. S. Morse. A distributed algorithm for solving a linear algebraic equation. *IEEE Transactions on Automatic Control*, 60(11):2863–2878, 2015.

[62] A. Nedic, A. Olshevsky, A. Ozdaglar, and J. N. Tsitsiklis. Distributed subgradient methods and quantization effects. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 4177–4184. IEEE, 2008.

[63] A. Nedic and A. Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.

[64] A. Nedic and A. Ozdaglar. Cooperative distributed multi-agent optimization. *Convex Optimization in Signal Processing and Communications*, 340, 2010.

[65] A. Nedic, A. Ozdaglar, and P. A. Parrilo. Constrained consensus and optimization in multi-agent networks. *IEEE Transactions on Automatic Control*, 55(4):922–938, 2010.

[66] Y. Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical programming*, 120(1):221–259, 2009.

[67] I. Notarnicola and G. Notarstefano. A duality-based approach for distributed optimization with coupling constraints. *IFAC-PapersOnLine*, 50(1):14326–14331, 2017.

[68] A. Ostrowski. Über die determinanten mit überwiegender hauptdiagonale. *Commentarii Mathematici Helvetici*, 10(1):69–96, 1937.

[69] N. Parikh, S. Boyd, et al. Proximal algorithms. *Foundations and Trends® in Optimization*, 1(3):127–239, 2014.

[70] Y. Peng, A. Ganesh, J. Wright, W. Xu, and Y. Ma. Rasl: Robust alignment by sparse and low-rank decomposition for linearly correlated images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2233–2246, 2012.

[71] Z. Peng, T. Wu, Y. Xu, M. Yan, and W. Yin. Coordinate friendly structures, algorithms and applications. *arXiv preprint arXiv:1601.00863*, 2016.

[72] Z. Peng, Y. Xu, M. Yan, and W. Yin. Arock: an algorithmic framework for asynchronous parallel coordinate updates. *SIAM Journal on Scientific Computing*, 38(5):A2851–A2879, 2016.

[73] E. Picard. Mémoire sur la théorie des équations aux dérivées partielles et la méthode des approximations successives. *Journal de Mathématiques pures et appliquées*, 6:145–210, 1890.

[74] W. Ren and R. W. Beard. *Distributed consensus in multi-vehicle cooperative control*. Springer, 2008.

[75] R. Rockafellar. On the maximality of sums of nonlinear monotone operators. *Transactions of the American Mathematical Society*, 149(1):75–88, 1970.

[76] R. T. Rockafellar. Monotone operators associated with saddle-functions and minimax problems. *Nonlinear functional analysis*, 18(Part 1):397–407, 1970.

[77] R. T. Rockafellar. Monotone operators and the proximal point algorithm. *SIAM journal on control and optimization*, 14(5):877–898, 1976.

[78] E. K. Ryu and S. Boyd. Primer on monotone operator methods. *Appl. Comput. Math*, 15(1):3–43, 2016.

[79] J. Sabater and C. Sierra. Reputation and social network analysis in multi-agent systems. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: Part 1*, pages 475–482. ACM, 2002.

[80] R. Scattolini. Architectures for distributed and hierarchical model predictive control–a review. *Journal of process control*, 19(5):723–731, 2009.

[81] G. Scutari and Y. Sun. Distributed nonconvex constrained optimization over time-varying digraphs. *arXiv preprint arXiv:1809.01106*, 2018.

[82] G. Shi and B. D. Anderson. Distributed network flows solving linear algebraic equations. In *American Control Conference (ACC), 2016*, pages 2864–2869. IEEE, 2016.

[83] N. Z. Shor. *Minimization methods for non-differentiable functions*, volume 3. Springer Science & Business Media, 2012.

[84] A. Simonetto and H. Jamali-Rad. Primal recovery from consensus-based dual decomposition for distributed convex optimization. *Journal of Optimization Theory and Applications*, 168(1):172–197, 2016.

[85] M. Sion et al. On general minimax theorems. *Pacific Journal of mathematics*, 8(1):171–176, 1958.

[86] M. Tao and X. Yuan. Recovering low-rank and sparse components of matrices from incomplete and noisy observations. *SIAM Journal on Optimization*, 21(1):57–81, 2011.

[87] T. Tatarenko and B. Touri. Non-convex distributed optimization. *IEEE Transactions on Automatic Control*, 62(8):3744–3757, 2017.

[88] J. N. Tsitsiklis. Problems in decentralized decision making and computation. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE LAB FOR INFORMATION AND DECISION SYSTEMS, 1984.

[89] H. Uzawa. Market mechanisms and mathematical programming. *Econometrica: Journal of the Econometric Society*, pages 872–881, 1960.

[90] H. Uzawa. Walras' tatonnement in the theory of exchange. *The Review of Economic Studies*, pages 182–194, 1960.

[91] A. N. Venkat, J. B. Rawlings, and S. J. Wright. Stability and optimality of distributed model predictive control. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pages 6680–6685. IEEE, 2005.

[92] D. Wang, Z. Wang, M. Chen, and W. Wang. Distributed optimization for multi-agent systems with constraints set and communication time-delay over a directed graph. *Information Sciences*, 438:1–14, 2018.

[93] E. Wei, A. Ozdaglar, and A. Jadbabaie. A distributed newton method for network utility maximization–i: Algorithm. *IEEE Transactions on Automatic Control*, 58(9):2162–2175, 2013.

[94] E. Wei, A. Ozdaglar, and A. Jadbabaie. A distributed newton method for network utility maximizationâĂŤpart ii: Convergence. *IEEE Transactions on Automatic Control*, 58(9):2176–2188, 2013.

[95] T. Wu, K. Yuan, Q. Ling, W. Yin, and A. H. Sayed. Decentralized consensus optimization with asynchrony and delays. *IEEE Transactions on Signal and Information Processing over Networks*, 4(2):293–307, 2018.

[96] L. Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, 11(Oct):2543–2596, 2010.

[97] Y. Xiao and J. Hu. Distributed solutions of convex feasibility problems with sparsely coupled constraints. In *Decision and Control (CDC), 2017 IEEE 56th Annual Conference on*, pages 3386–3392. IEEE, 2017.

[98] P. Yi and L. Pavel. A distributed primal-dual algorithm for computation of generalized nash equilibria with shared affine coupling constraints via operator splitting methods. *arXiv preprint arXiv:1703.05388*, 2017.

[99] P. Yi and L. Pavel. Asynchronous distributed algorithm for seeking generalized nash equilibria under full and partial decision information. *arXiv preprint arXiv:1801.02967*, 2018.

[100] P. Yi and L. Pavel. Distributed generalized nash equilibria computation of monotone games via double-layer preconditioned proximal-point algorithms. *IEEE Transactions on Control of Network Systems*, 2018.

[101] M. Zhu and S. Martínez. On distributed convex optimization under inequality and equality constraints. *IEEE Transactions on Automatic Control*, 57(1):151–164, 2012.

[102] M. Zhu and S. Martínez. An approximate dual subgradient algorithm for multi-agent non-convex optimization. *IEEE Transactions on Automatic Control*, 58(6):1534–1539, 2013.