DISTRIBUTED CONTROL AND OPTIMIZATION IN MULTI-AGENT SYSTEMS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Xuan Wang

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

August 2020

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Shaoshuai Mou, Chair

School of Aeronautics and Astronautics

Dr. Dengfeng Sun

School of Aeronautics and Astronautics

Dr. Martin J. Corless

School of Aeronautics and Astronautics

Dr. Shreyas Sundaram

School of Electrical and Computer Engineering

Approved by:

Dr. Gregory A. Blaisdell

Head of the School Graduate Program

ACKNOWLEDGMENTS

First, I would like to express my sincere gratitude towards my advisor, Prof. Shaoshuai Mou, for his guidance and support during my doctoral study. He provided me the chance to join Purdue University and work in Autonomous & Intelligent Multi-agent Systems (AIMS) Lab. He is an inspiring mentor who provides constructive instructions on my research path, and teaches me how to be a rigorous researcher. I appreciate every discussion I have had with him, which spans academic, daily life, and future career plans. Thank you Prof. Mou, for all the things you have done for me!

In addition, I would like to thank my committee members, Prof. Dengfeng Sun, Prof. Martin Corless, and Prof. Shreyas Sundaram for their suggestions and collaborations during my Ph.D. study. I would also like to show my sincere thank to Prof. Brian Anderson for his collaboration on my research projects and valuable support on my job applications. My lab mates and friends Wanxin Jin, Jiazhi Song, Jingqiu Zhou, Jeffrey Hall, Paulo Heredia Aguilar, Zihao Liang, Mark Duntz, Jason King Lo, Nicholas Schultz, Tianyu Zhou, Kevin Shi, Arthur de Waleffe, David Bambrick, Jiazhen Zhou, Dawei Sun, Chuyu Zhang also helped me a lot during my four years at Purdue, I would like to thank all of them.

Finally, I would like to thank my mom and dad, Lijuan Zhou and Zengmin Wang, for their endless love, support, and encouragement. Last but not least, I would like to thank my girlfriend, Shuang Wu, for her patience and support of my academic endeavor.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

SYMBOLS

$\mathbf{1}_r$	the vector in $\mathbb{R}^r$ with all entries equal to 1
I_r	$r \times r$ identity matrix
$\otimes$	the Kronecker product
$M^\top$	the transpose of a matrix M
$M > 0$	the symmetric matrix M is positive definite
$M \geq 0$	the symmetric matrix M is positive semi-definite
eig (M)	the set of all eigenvalues of a matrix M
image M	the image of a matrix M
$\ker M$	the kernel of a matrix M
diag $\{A_1, A_2, \cdots, A_r\}$	the block diagonal matrix with A_i the ith diagonal block entry
col $\{A_1, A_2, \cdots, A_r\}$	a column stack of matrices A_i with the index in a top-down ascending order

ABSTRACT

Wang, Xuan PhD, Purdue University, August 2020. Distributed Control and Optimization in Multi-agent Systems. Major Professor: Shaoshuai Mou.

In recent years, the collective behaviors in nature have motivated rapidly expanding research efforts in the control of *multi-agent* systems. A multi-agent system is composed of multiple interacting subsystems (agents). In order to seek approaches that respect the network nature of multi-agent systems, *distributed* algorithms has recently received a significant amount of research attention, the goal of which is allowing multi-agent systems to accomplish global objectives through only local coordination. Under this scope, we consider three major problems in this dissertation, namely, distributed computation, distributed optimization, and the resilience of distributed algorithms. First, for distributed computation, we devise distributed algorithms for solving linear equations, which can eliminate the initialization step for agents; converge to the minimum l_1 and l_2 solutions of under-determined linear equations; achieve ultimate scalability inters of agents' local storage and local states. Second, for distributed optimization, we introduce a new method for algorithm discretization so that the agents no longer have to carefully choose their step-size. We also introduce a new distributed optimization approach that can achieve better convergence rate with lower bandwidth requirement. Finally, for the resilience of distributed algorithms, we propose a new approach that allow normal agents in the multi-agent system to automatically isolate any false information from malicious agents without identification process. Though out the dissertation, all mentioned results are theoretically guaranteed and numerically validated.

1. INTRODUCTION

1.1 Background and Motivation

Through the past few decades, with the development of techniques for control and optimization, researchers have achieved significant accomplishments on autonomy. We allows robots to understand the environment, make human-like reasoning, and then do decisions. However, for single robots, due to the considerations on cost, size and mobility, they are usually equipped with limited hardware, so that can only offer restricted functionality and a lower level of autonomy. In such a situation, as the current manufacturing technology is not sufficient to integrate various hardware-dependent functionalities into one small robot, a natural way to improve the capability of the system is to introduce multiple robots (agents) [1] to work together, so that they can offer collective intelligence, a wider range and type of operations, and therefore a higher-level of autonomy. However, these mentioned benefits cannot be directly obtained by simply putting these robots together. Instead, an essential gap that naturally arises is how one can guarantee that all the agents are cooperative, say they are working towards a same goal as a cohesive whole. To achieve corporation, the agents in the system must be interactive. However, such interaction are usually subject to a network constraints to communications among agents, which makes multi-agent systems more challenging to control. Specifically, the network constraints, to the relations between agents involving sensing, communication or control, usually prohibit the application of traditional methods from controlling the multi-agent systems in a centralized manner. In order to seek new control approaches that respect the network nature of multi-agent systems, *distributed control* has recently received a significant amount of research attention, the goal of which is to allow multi-agent systems to accomplish global objectives through only local coordination. Here, the

word 'local' contexts interaction between any given agent and a limited number of associated 'neighbor' agents, often physically adjacent. Towards this end, the idea of *consensus* [2], the aim of which is to drive all agents in the network to reach an agreement regarding a certain quantity, has served as a basis in deriving many distributed algorithms for multi-agent systems such as motion synchronization [1]; multi-robot path planning/formation control [3]; flocking of mobile robots [4]; and cooperative sensing [5].

In general, the key benefits of consensus-based distributed algorithms can be summarized from two perspectives. One perspective is Top-down Analysis, where given a sophisticated task, we care about how to decompose the task into small pieces so that the local sub-tasks do not exceed the limited capability of individual agents. The examples for this include the Large-scale computation [6], where you may want to find a solution subjection to multiple constraints. In this case you can assign each constraint to an agent and let them work together to find the solution to the original problem. Another example is the Swarm control of autonomous vehicles [7], where it may be difficult to use a single computer to control all the vehicles running on the road. In this case, we allow the vehicles to control themselves, but at the meanwhile maintain an agreement on higher-level decisions such as lane change, lane assignment, priority in front of traffic lights. Except for top down analysis, another prospective is Bottom-up Synthesis, where we assume each agent has some local data and our goal to synthesize these scattered knowledges into a collective intelligence. Examples for this include the information fusion in multi-sensor network [8] where we want to get more accurate estimation of the overall system. Or in Collaborative learning [9], where the objective can be local risk function, and the goal is to find a parameter set that optimizes the learning model in a global sense.

1.2 Key Challenges

Under the scope of consensus-based distributed coordination for multi-agent systems, the key challenges arises from the following aspects.

- **Scalability**. Given a complex task to the overall system, we care about how we can decompose the task into smaller pieces, such that each piece scales well with the limited capability of the individual agent.
- **Resilience**, which has become a very popular topic for cyber-physical systems in recent years. As we have introduced, multi-agent systems have network natures, which leaves the system a large attacking surface and great vulnerability to cyber-attacks. In this case, we care about how to maintain the functionality of the systems, even in the presence of sophisticated attacks.

In this dissertation, our goal is to introduce some methods, trying to fill the gap of multi-agent system by addressing the challenges we have identified. In the following, I will explain how one can formulate different types of multi-agent coordination problems into consensus-based distributed optimization problems and then solve them in scalable and resilient ways. Particularly, this dissertation covers three major types of problems, which are briefly introduced in the following.

1.3 Literature Review

1.3.1 Distributed Computation

Distributed computation features prominently in many of today's most pressing engineering and data science challenges including cooperative machine learning [9]; multi-robot coordination [10]; large-scale power grid regulation [11]; complex numerical computation [6] (computational fluid dynamics, finite-element analysis); and so on. Among all numerical computations, perhaps the most fundamental problem is solving a system of linear algebraic equations. Efforts to develop distributed algorithms to solve such systems have been underway for a long time. The objective is to

achieve better efficiency by decomposing a large system of linear equations into smaller ones and then being cooperatively solved by multiple computational agents [12–15]. In these results, a commonly used idea is a so-called "agreement principle" [12], in which each agent limits the update of its state to satisfy its own equation while trying to reach a consensus with its nearby neighbors' states. Based on this idea, if the underlying network jointly strongly connected, the distributed algorithm developed in [12] is able to converge to an exact solution to the original linear equation with exponential convergence speed. As [12] being one of the classical works in this area, many related works in the literature further improves the results of this paper. For example in [16], by adding extra modification terms to the algorithm in [12], one can remove the initialization step that is required by the algorithm. Another modification of [12] can be found in [17], which takes into account the sparsity of the known matrix of the linear equation in order to reduce the agents' state size. Furthermore, the result in [18] studies the eigenvalue that characterize the exponential convergence parameter of [12]. [19] studies the weakest possible conditions for the exponential convergence of the algorithm, from a graphical prospective. [20] shows that if the communication graph is random (instead of a strict guarantee on strongly connectedness), the algorithm has almost surely convergence. Except for the mentioned works that mainly explore the algorithm of [12] in terms of converge rate, some other works consider the effectiveness of the algorithm under different network condition. [21] shows that the data transmitting delays do not influence the result of [12]. [22] studies the security issues, which introduces a verification method to protect the algorithm against the local data manipulation in agents. In [23], an event trigger based algorithm is studied. By the trigger mechanism, the communication overhead of the network can be reduced. In addition, a component dropping mechanism is developed [24] which also aims to achieve better communication-efficiency.

Note that the algorithm developed in [12] is only able to achieve a feasible solution to linear equations. If the equation has multiple solutions, one usually cares about some special solutions in the solution space. For example, motivated by compres-

sion sensing applications, [25] developed a distributed algorithm based on Fillipov Set Value maps, which is able to achieve the minimum l_1 norm solution of underdetermined linear equations in finite-time; motivated by energy minimization, the algorithm developed in [16] can achieve a particular solution that has the minimum l_2 norm by a special state initialization on [12]; a same result can be achieve by [26] through a different approach namely, M-Fejer mappings.

Except for the results that are closely related to [12], other distributed algorithms for achieving the exact solution of a linear equation can also be found in the existing literature. For example, for positive definite linear equations, the result in [27] partitioned the linear equation such that a local square matrix is assigned to each agent of the network, and the summation of all local matrices equals the original equation. Then it provides an algorithm that allows the agents in the network to solve the overall equation. During the solving process, the agents can join and leave the network at any time, for infinitely many times, and lose all its memory upon leaving. In [28], by adding some partial centralized information and an adaptive momentum to agents' state update, one can achieve accelerated convergence rate on distributed algorithms for solving linear equations. [29] considers linear equations that are generalized diagonal dominant, then by developing a distributed Gauss-Seidel, one can achieve discrete-time finite time convergence for acyclic graphs and exponential convergence for cyclic graphs. In [13], when the known matrix of the equation is square and nonsingular, a continuous update is proposed to achieve the unique solution exponentially fast under fixed, undirected and connected graph. [30] considers a more complicated matrix equation, namely, $AXB = F$, where A, X, B, F are matrices of proper size. By adding four sets of dual variables, it develops continuous-time algorithms that can solve this equation in distributed manner with undirected graph. In [31], instead of partitioning the equation into rows as in [12], it partitions the overall linear equation into columns. By doing this, the state of each agent does not have to be a complete copy of the overall equation, but only a small section of the overall unknown vector. Under this setup, an algorithm is proposed, which can achieve the solution

exponential fast under fixed undirected connected graphs. By combining the ideas of [31] and [12], in [6], a linear equation can be partitioned into rows and columns simultaneously, such that each agent only knows a small square block of the equation, and can cooperatively achieve the solution to the linear equation exponentially fast.

1.3.2 Distributed Optimization

Apart from distributed computation, another major problem in multi-agent control is distributed optimization.Though only local coordination, the goal of distributed optimization is to find a common decision vector which optimizes a global objective function that can be written as an aggregate of agents' local functions. By associating these functions with the application specified objectives, the technique of distributed optimization has been extensively applied to various types of multi-agent systems, including the synchronization of coupled oscillators [11, 32], multi-robot formation control [7], flocking of mobile robots [33], cooperative sensing [34] and multi-agent learning [35, 36].

To establish algorithms for distributed optimization, one can similarly combine the idea of *consensus* with the *gradient descent* method [37]. Different from the case in distributed computation where all agents' constraints must process at least one common solution, for distributed optimization, the local objective functions usually do not have a common minimizer. This means in general, the consensus and the gradient descent updates will have different equilibria so that the states in all agents can never converge to a same point. To circumvent this difficulty, the work in [37–40] applies a diminishing step-size (i.e. $1/t$) to the gradient term in the update equations. As a side effect, this time-variant step-size must be shared by all the agents in the network, and the convergence rate of the algorithm will be degraded most commonly to $\mathcal{O}(1/\sqrt{t})$ (from exponential). In order to improve the convergence rate, many recent works have shown that the exponential convergence rate can be achieved by doubling the dimension of the state vector. See for example the continuous-time update introduced

in [41], where the role of the extra vector is played by the Lagrangian dual vector for the consensus error, and the discrete-time update introduced in [42, 43], where the extra vector performs gradient tracking. In these algorithms, the extra states have to be exchanged across the network, necessitating duplication of the network bandwidth requirement.

In many practical applications, due to feasibility reasons, the distributed optimization problems are additionally associated with local constraints [37–39, 44–46]. To solve such a constrained optimization problem, one usually needs an algorithm that integrates *consensus*, *gradient descent* and *projection* all together. In [37], an algorithm for constrained optimization is developed by assuming that the constraints in all the agents are identical. Further, in [38], the result of [37] is generalized by removing the extra assumption. Note that both the algorithms in [37,38] are obtained by applying a projection operator to an existing distributed optimization algorithm with diminishing step-sizes, they are discrete-time and have a convergence rate of $\mathcal{O}(1/\sqrt{t})$. In addition to the algorithms based on diminishing step-sizes, another line of research for constrained optimization stems from applying projection operators to the algorithm [41] based on doubled state dimensions. This has led to the recent achievements proposed in [44–46]. Although these algorithms can potentially achieve a faster convergence rate than the ones based on diminishing step-sizes, yet the exponential convergence of these algorithms has never been theoretically guaranteed.

1.3.3 Resilient Consensus-based Distributed Algorithms

Simple and powerful as the idea of consensus for developing distributed computation/optimization algorithms, its effectiveness heavily depends on the assumption that all agents in the network are sharing trustful information with their neighbors. In practice, however, the presence of a large number of diverse agents in multi-agent systems (provided by different vendors, with differing levels of trust) introduces the potential for maliciously compromised agents (Byzantine) to inject misinformation

that propagates throughout the network and prevents the entire system from achieving a proper consensus-based coordination. As shown in [47], even one misbehaving agent in the network can alter the states of other agents to any value it desires. To fix such vulnerability, traditional information-security approaches (built on the pillars of confidentiality, integrity, and availability) focus on protecting the data itself, but are not directly applicable to distributed control algorithms, which involve not only data gathering but also data processing, information sharing and coordination between highly-mobile agents. Thus, instead of narrowly focusing on techniques such as data encryption that aim to prevent attackers, a new line of research arises, which aims to seek fundamentally new approaches that allow the network to reliably perform consensus even after the adversary has successfully compromised certain agents in the system. Towards this end, the main challenges arise from the special characteristics of distributed systems where all agents in the network are with high mobility and no agent has access to the global information. As a result, the paradigm of fault-tolerant control [48] equipped with state-observers are not sufficient to capture the sophisticated manners the malicious agents take to avoid detection.

In order to allow the normal agents in multi-agent systems to utilize only their local information to mitigate or fully isolate the influence of malicious attackers, in the existing literature, different attempts have been made, dating back to the results in [49,50], which shows that a resilient consensus can be asynchronously achieved among m agents with at most $(m-1)/3$ malicious attackers. These results are applicable to binary problems where the state of each agent either take 0 or 1. In reality, however, engineering applications usually require the multi-agent system to reach a consensus (synchronization) on a continuous domain that has infinite many choices. To solve the problems of this type, the algorithm in [51] is developed, based on the idea of letting each normal agent to run distributed consensus by excluding the most extreme values that are far away from its other neighbors. A similar result has been parallelly proposed in [52], which can be used for resilient distributed optimization with applications in machine learning. Both of these results have introduced certain

restrictions to the behavior of malicious agents, which means if the malicious agents can intentionally choose the states they send to the normal agents that maliciously prevent them from reaching a proper consensus, these resilient algorithms may fail. Such worst case scenario can be handled by the algorithm proposed in [53], which guarantees a safe consensus even the malicious agents send arbitrary information to different normal agents. The only limitation of this algorithm [53] lies in the requirement that the agents' states must be scalar. Although in the multi-dimensional case, one can run the algorithms separately on each entry of the state vector, such a scheme will violate the convex combination property of distributed consensus (i.e. the agents states in the new time-step are not obtained by the convex combination of the ones at the current time-step.) and becomes ineffective if the normal agents are subject to local state constraints [6, 12]. To reach a completely theoretically safe distributed consensus and also respect the convex combination property among agents states, the methods based on Tverberg-points have been proposed for both centralized [54] and distributed cases [55, 56]. According to [57], the computational complexity of achieving Tverberg points are usually high. To lower the computational complexity, new approaches are developed in [58, 59], which are based on an idea called *resilient convex combination*. The difference between [58] and [59] is that in [59], each normal agent employs a special treatment on its own state so that have a relaxed requirement on its neighbor's state. But apart from such differences, the key advantage of both [58] and [59], is that their resilient convex combination are locally solvable via linear or quadratic programming methods.

1.4 Contributions

The purpose of this dissertation is to develop novel distributed algorithms for multi-agent systems, namely, distributed computation, distributed optimization, and the resilience of distributed algorithms.

For distributed computation, we developed a distributed algorithm for solving linear equations [16], which can achieve better numerical stability against round-off errors and can eliminate the initialization step for agents. Such initialization is necessary for traditional approaches. Second, if the linear equation system is under-determined, traditional algorithm can only achieve an arbitrary solution. Here, by introducing new algorithms, we allow the agents to converge to a particular solutions with the minimum l_1-norm [25,60] and minimum l_2-norm [61]. Finally, compared with traditional approaches where each agent has to know at least one complete row of the original linear equation, by combining the idea of consensus and conservation, we proposed a new distributed framework for solving linear equations that can achieve ultimate scalability inters of agents' local storage and local states.

For distributed optimization, we developed an algorithm for distributed least-squares approximation [62, 63]. Compared with traditional approaches that needs to carefully chose step-sizes for convergence, our approach does not involve any small or time-varying step sizes. In addition, for more general constrained distributed optimization [6] problems, we developed a new distributed algorithm by incorporating the idea of integral feedback. Compared with traditional approaches, the algorithm has better convergence rate and can save the communication bandwidth by 50%.

For the resilience of distributed algorithms, we have developed an algorithm that allows the normal agents to compute a *resilient convex combination* [59]. Different from traditional approached that are mainly based on identification, this approach can automatically isolate any malicious information manipulated by cyber-attacks without any identification process. In terms of implementation, it has low computational complexity, only based on local information and can be integrated into any consensus-based distributed algorithms [64].

1.5 Dissertation Overview

The rest part of the dissertation is organized as follows. In Chapter 2, we first introduced an existing distributed algorithm for solving linear equations called DALE. Then we modified DALE in the following ways: eliminate the requirement for initialization; allows DALE to achieve a minimum l_2 norm solution; allow DALE to achieve a minimum l_1 norm solution. In Chapter 3, we still focus on distributed algorithms for solving linear equations. But compared with the ones in Chapter 2 that are based on the idea of consensus, the algorithms in this chapter integrates another idea called conservation. By introducing a double-layered network that actively combines the ideas of consensus and conversation, algorithms are developed that can achieve much better scalability. Chapter 4 introduces two algorithms for distributed optimization. The first one considers a least square minimization problem while the second one considers a more general convex optimization problem with local constraints. In Chapter 5, we discuss the resilience for consensus-based distributed algorithms, by introducing a new concept named resilient convex combination, we allow multi-agent systems to achieve automated resilience towards cyber attacks. At last, conclusion remarks and future directions are summarized in Chapter 6.

2. CONSENSUS-BASED DISTRIBUTED ALGORITHMS FOR SOLVING LINEAR EQUATIONS [1]

2.1 Introduction

Solving linear equation is a fundamental problem in many engineering applications. In recent years, the modeling and controlling of large scale systems (such as power supply management; water quality management; and swarm control of autonomous robots), calls for distributed algorithms for solving linear algebraic equations [12, 13, 16, 40, 62], which achieves efficiency by decomposing a large system of linear equations into smaller ones that can be cooperatively solved by a network of agents. Compared with distributed equation solvers such as Gaussian Belief Propagation [66] and Neumann series approximation [67], which require the linear equation to be either diagonally dominant or positive definite, the consensus-based distributed linear equation solvers [12,13,16,40,62] are applicable to a much larger class of linear equations.

Particularly, consider a network of m agents in which each agent i is able to communicate with certain other nearby agents called its neighbors, denoted by $\mathcal{N}_i$. The neighbor relations can be described by a graph $\mathbb{G}$ such that there is an edge from j to i if and only if $j \in \mathcal{N}_i$. In this chapter, we assume $\mathbb{G}$ is *directed*. Suppose each agent i knows a local matrix $A_i \in \mathbb{R}^{n_i \times n}$ and a corresponding solution vector $b_i \in \mathbb{R}^{n_i}$, which are some partitions of an original linear equation $Ax = b$ such that

$$\begin{bmatrix} A & b \end{bmatrix} = \begin{bmatrix} A_1 & b_1 \\ A_2 & b_2 \\ \vdots & \vdots \\ A_m & b_m \end{bmatrix}, \quad A \in \mathbb{R}^{\bar{n} \times n} \tag{2.1}$$

[1] Resarch in this section has been published in the papers [16,65] with me as the leading author.

Suppose each agent i controls a state vector $x_i(t) \in \mathbb{R}^n$, which could be looked as the estimate of agent i to the solution of the overall equation $Ax = b$. The goal is to develop distributed algorithms that allows the states in all agents to converge to a common vector x^*, which is a common solution satisfying all agents' local linear equations, i.e.

$$x_1 = x_2 = \cdots = x_m = x^* \tag{2.2}$$

$$A_i x_i = b_i \tag{2.3}$$

Towards this end, in existing literature, the authors of [12] have devised the following **D**istributed **A**lgorithm for solving **L**inear **E**quations (**DALE**):

1. **Initialization:** At $t = 0$, each agent i initializes $x_i(0)$ such that $A_i x_i(0) = b_i$;

2. **Update:** At $t + 1$, $t = 0, 1, ...,$, each agent i receives $x_j(t)$ from certain other agents denoted by $\mathcal{N}_i(t)$, and then update its own state to be

$$x_i(t+1) = x_i(t) - P_i \left(x_i(t) - \frac{1}{d_i(t)} \sum_{j \in \mathcal{N}_j(t)} x_j(t) \right) \tag{2.4}$$

   Here, P_i is the orthogonal projection matrix to the kernel of A_i; $\mathcal{N}_i(t)$ denotes the set of agent i's neighbors at time t, from which agent i can receive information. We always assume that each agent is a neighbor of itself, that is, $i \in \mathcal{N}_i(t)$; $d_i(t)$ denotes the cardinality of $\mathcal{N}_i(t)$.

A major result of the DALE is the following theorem:

Theorem 2.1.1 *[12] Suppose $Ax = b$ has at least one solution and assume that the sequence of neighbor graphs $\mathbb{G}(t)$, $t \geq 1$, is repeatedly jointly strongly connected. Then the DALE drives all $x_i(t)$ converges exponentially fast to the same solution to $Ax = b$ as $t \to \infty$.*

We refer to [12] for the introduction of "repeatedly jointly strongly connected", which has been shown to be not only the sufficient but also necessary requirement of the network connectivity for the above theorem to hold.

The DALE developed in [12] is applicable to all linear equations that have at least one solution; converges exponentially fast; works for a large class of time-varying directed graphs; operates asynchronously; and does not involve any time-varying step-size. The aim of this chapter is to achieve further improvement to the DALE by addressing the following two problems:

- The DALE heavily depends on that each state vector $x_i(t)$ is initialized to be a solution to $A_i x = b_i$. When there are round-off errors in such initialization step [68], the DALE might fail to achieve the exact solution to $Ax = b$. Moreover, such initialization step requires each agent to be capable of finding an exact solution to $A_i x = b_i$, which may be out of the computation capability of low-cost agents. Motivated by this, the Section 2.2 of this chapter aims to eliminate such initialization step and allows $x_i(0)$ can be chosen arbitrarily from $\mathbb{R}^n$.

- When $Ax = b$ has more than one solutions, the DALE achieves one of them. However, it is unclear that which particular solution it converges to. In Sections 2.3 and 2.4, we will make certain modifications that allow the DALE to achieve specific solutions with have minimum l_2 or l_1 norms. The significance of such solutions are that, if x represents the state vector needed to be controlled in industrial process [10, 69], a minimum l_2 norm solution ensures the system to achieve its goal with the lowest cost; if x represents the state vector corresponding to an out signal from a compression sensing process [70–72], a minimum l_1 norm solution ensures the algorithm to achieve the original sparse signal one wants to recover.

2.2 A Modified DALE without Initialization

2.2.1 The Problem

The DALE in [12] requires an initialization step in which each $x_i(0)$ is initialized to be a solution to $A_i x = b_i$. In this section, the problem of interest is to eliminate the

initialization step of DALE. To do this, we modify the DALE by adding one additional term to (2.4), which drives all $x_i(t)$ to the manifold $Ax = b$ even when they are not initialized to be so. Please note that a similar idea has also been discussed in [13].

2.2.2 Main Result

Let $\left[\bar{A}_i\ \bar{b}_i\right]$ denotes a submatrix of $[A_i\ b_i]$ such that $\ker A_i = \ker \bar{A}_i$ and $\bar{A}_i\bar{A}_i'$ is non-singular, which also implies that $A_ix = b_i$ if and only if $\bar{A}_ix = \bar{b}_i$. By doing this, the orthogonal projection matrix P_i to the kernel of A_i can be expressed as $P_i = I - \bar{A}_i'(\bar{A}_i\bar{A}_i')^{-1}\bar{A}_i$. The modified DALE will be: At $t = 0$, $x_i(0)$ is any vector in $\mathbb{R}^n$; The update for each agent i at $t+1$ is

$$\begin{aligned} x_i(t+1) &= x_i(t) - P_i\left(x_i(t) - \frac{1}{d_i(t)}\sum_{j\in\mathcal{N}_j(t)} x_j(t)\right) \\ &\quad -\bar{A}_i'(\bar{A}_i\bar{A}_i')^{-1}(\bar{A}_ix_i(t) - \bar{b}_i) \end{aligned} \tag{2.5}$$

Remark 2.2.1 *It is worth mentioning that by multiplying $\bar{A}_i$ to both sides of algorithm (2.5), one has $\bar{A}_ix_i(t+1) = \bar{A}_ix(t) - (\bar{A}_ix_i(t) - \bar{b}_i)$ because of $\bar{A}_iP_i = 0$. Then $\bar{A}_ix_i(t+1) = \bar{b}_i$ for $t = 0, 1, 2....$ This observation implies that $x_i(t)$ for $t \geq 1$ is always a solution to $\bar{A}_ix = \bar{b}_i$ even if $x_i(0)$ is not.*

To prove $x_i(t)$ converges to x^* where $Ax^* = b$, we prove the error vector $e_i(t) = x_i(t) - x^*$ converge to 0. From (2.5), $\bar{A}_ix^* = \bar{b}_i$, and $P_i = I - \bar{A}_i'(\bar{A}_i\bar{A}_i')^{-1}\bar{A}_i$, one has

$$\begin{aligned} e_i(t+1) =& x_i(t) - x^* \\ & - P_i\left(x_i(t) - x^* - \frac{1}{d_i(t)}\sum_{j\in\mathcal{N}_j(t)}(x_j(t) - x^*)\right) \\ & - \bar{A}_i'(\bar{A}_i\bar{A}_i')^{-1}(\bar{A}_ix_i(t) - \bar{A}_ix^*) \\ =& e_i(t) - P_i(e_i(t) - \frac{1}{d_i(t)}\sum_{j\in\mathcal{N}_j(t)} e_j(t)) - (I - P_i)e_i(t) \\ =& \frac{1}{d_i(t)}P_i\sum_{j\in\mathcal{N}_j(t)} e_j(t) \end{aligned}$$

which is the same as the error equation in [12], thus, one has the following lemma:

Lemma 2.2.1 *All $e_i(t)$ converge exponentially fast to be the same value e^*, where $e^* = 0$ when $Ax = b$ has a unique solution.*

From Lemma 2.2.1 and $e_i(t) = x_i(t) - x^*$, one has all $x_i(t)$ converges to be

$$z^* = e^* + x^*$$

When $Ax = b$ has a unique solution. One has $z^* = x^*$ and all $x_i(t)$ converges to the unique solution x^*. Since e^* may not be zero when $Ax = b$ has multiple solutions, we will need to show that z^* is actually also a solution to $Ax = b$. Note that z^* is the common value where all $x_i(t)$ converge to. Thus, z^* must satisfy equation (2.5). By replacing $x_i(t+1)$ and $x_i(t)$ by z^*, one has

$$\begin{aligned} z^* =& z^* - P_i(z^* - \frac{1}{d_i(t)} \sum_{j \in \mathcal{N}_j(t)} z^*) \\ & - \bar{A}_i'(\bar{A}_i \bar{A}_i')^{-1}(\bar{A}_i z^* - \bar{b}_i),\ i = 1, 2, ..., m \end{aligned} \tag{2.6}$$

which implies

$$\bar{A}_i'(\bar{A}_i \bar{A}_i')^{-1}(\bar{A}_i z^* - \bar{b}_i) = 0,\ i = 1, 2, ..., m \tag{2.7}$$

Multiplying $\bar{A}_i$ to both sides of (2.7) leads to

$$\bar{A}_i z^* - \bar{b}_i = 0,\ i = 1, 2, ..., m \tag{2.8}$$

Then

$$A_i z^* = b_i,\ i = 1, 2, .., m$$

Thus, z^* is a solution to $Ax = b$. To summarize, we have the following theorem:

Theorem 2.2.1 *The modified DALE (2.5) with arbitrary $x_i(0) \in \mathbb{R}^n$ drives all $x_i(t)$ to converge exponentially fast to be a common solution to $Ax = b$.*

2.2.3 Validation

To validate Theorem 2.2.1, consider the following linear equation,

$$\begin{bmatrix} A_1 \\ A_2 \\ A_3 \end{bmatrix} = \left[\begin{array}{ccccc} 19 & 0 & 12 & 5 & 10 \\ 12 & 16 & 10 & 5 & 5 \\ \hline 5 & 12 & 16 & 14 & 16 \\ 4 & 16 & 16 & 17 & 4 \\ \hline 13 & 4 & 5 & 7 & 4 \end{array}\right], \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \left[\begin{array}{c} 5 \\ 17 \\ \hline 99 \\ 9 \\ \hline 51 \end{array}\right] \tag{2.9}$$

with the unique solution $x^* = [0.96\ 2.77\ -12.46\ 7.25\ 10.10]'$.

Let $V(t) = \sum_{i=1}^{3} |x_i(t) - x^*|^2$. Then $V(t) = 0$ if and only if all $x_i(t) = x^*$. Suppose the three agents are all connected with each other with undirected edges. Fig. 2.1 shows that the modified DALE (2.5), without any initialization step, drives all agents exponentially fast to x^*.

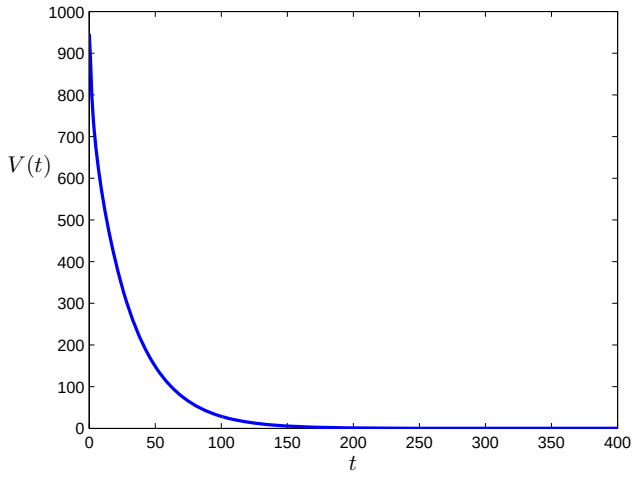

Figure 2.1. The speed $x_i(t)$ achieves x^* in Example 2

2.3 A Distributed Algorithm for minimum l_2 norm solution

2.3.1 The Problem

When $Ax = b$ has multiple solutions, the DALE developed in [12] enables each agent to achieve one of its solutions. However, which one is to be achieved is not clear. In this section, we will modify the initialization step of the DALE, so that it can achieve a specific minimum l_2 solution. Note that minimum l_2 norm solution is defined in the Euclidean distance, which is the solution being closest to the origin. Here, we generalize the concept of l_2 norm a little bit, by finding a solution that is closest to a specified point $q \in \mathbb{R}^n$. Specifically, consider

$$x^q_{\min} = \arg\min_{Ax=b} \frac{1}{2}|x - q|^2, \tag{2.10}$$

where $|\cdot|$ denotes the Euclidean norm. Obviously, when $q = 0$, $x^q_{\min}$ becomes the traditional minimum l_2 norm solution. Note that finding such a $x^q_{\min}$ can be formulated as a convex optimization problem and then being solved by the Dykstra's cyclic projection method [73]. Such cyclic projection method usually requires a centralized scheduling among all agents in the network.

2.3.2 Main Result

In this section we will show that under the same network connectivity requirement as in [12], and by utilizing the following special initialization step, the update (2.4) achieves $x^q_{\min}$ exponentially fast :

Initialization: Each agent i initializes its $x_i(0)$ to minimize

$$\frac{1}{2}|x - q|^2 \text{ subject to } A_i x = b_i \tag{2.11}$$

The above initialization (2.11) could be easily completed by solving a linear equation $A_i x = b_i$ and $P_i x = P_i q$ according to the following lemma:

Lemma 2.3.1 *$x = x^q_{\min}$ if and only if it satisfies $Ax = b$ and $P_A x = P_A q$, where P_A is the orthogonal projection to* $\ker A$.

Proof of Lemma 2.3.1: By the standard Lagrange multiplier method for convex optimization subject to linear constraints, there must exist a λ and $x^q_{\min}$ such that

$$Ax = b \tag{2.12}$$

$$x - q + A'\lambda = 0 \tag{2.13}$$

Multiplying P_A to (2.13), one has $P_A(x-q) + (AP_A)'\lambda = 0$ which and AP_A=0 imply

$$P_A x = P_A q \tag{2.14}$$

Thus $x^q_{\min}$ must satisfies linear equations (2.12) and (2.14), which has the unique solution since $\ker P_A \cap \ker A = 0$ because of the fact image $P_A = \ker A$.

To summarize, the point in $\mathbb{R}^n$ which is a solution to $Ax = b$ and minimizes $\frac{1}{2}|x-q|^2$ always exists, and must be the unique solution of (2.12) and (2.14). Thus Lemma 2.3.1 is true. ■

The hold of this Lemma leads to the following theorem

Theorem 2.3.1 *With the initialization* (2.11) *and the update* (2.4)*, all $x_i(t)$ converge exponentially fast to be $x^q_{\min}$.*

Proof of Theorem 2.3.1: Note that by the initialization (2.11) one still has $A_i x_i(0) = b_i$, which is a special case of the DALE. By Theorem 2.1.1, all $x_i(t)$ converges exponentially fast to be the same x^* such that $Ax^* = b$. To prove Theorem 2.3.1, we only need to prove that x^* additionally minimizes $\frac{1}{2}|x-q|^2$, or equivalently by Lemma 2.3.1, $P_A x^* = P_A q$ holds. Since x^* is the final value that all $x_i(t)$ converges to, it suffices to show

$$P_A x_i(t) = P_A q, \ i = 1, 2, ..., n \tag{2.15}$$

for all $t = 0, 1, 2, ...$, for which the induction method will be employed in the followings.

We first show

$$P_A x_i(0) = P_A q. \tag{2.16}$$

From image $P_A = \ker A$, image $P_i = \ker A_i$ and $\ker A \subset \ker A_i$, one has image $P_A \subset$ image P_i. Then

$$\ker P_i \subset \ker P_A$$

which and image $(I - P_i) = \ker P_i$ imply $P_A(I - P_i) = 0$, that is,

$$P_A P_i = P_A \tag{2.17}$$

By the initialization (2.11), one has $P_i x_i(0) = P_i q$. This, along with (2.17) leads to (2.16).

Now we suppose (2.15) is true for all i at t, and show it is true at $t+1$. From (2.4), (2.17) and the induction assumption, one has

$$\begin{aligned} P_A x_i(t+1) &= P_A x_i(t) - P_A P_i \left(x_i(t) - \frac{1}{d_i(t)} \sum_{j \in \mathcal{N}_j(t)} x_j(t) \right) \\ &= P_A x_i(t) - \left(P_A x_i(t) - \frac{1}{d_i(t)} \sum_{j \in \mathcal{N}_j(t)} P_A x_j(t) \right) \\ &= \frac{1}{d_i(t)} \sum_{j \in \mathcal{N}_j(t)} P_A q \\ &= P_A q \end{aligned}$$

Thus (2.15) is true. This completes the proof. ■

Remark 2.3.1 *It is worth pointing out that the initialization* (2.11) *is a special case of the initialization step in the DALE. All the results obtained under the DALE in [12] are directly applicable here.*

2.3.3 Validation

Suppose that the following linear equation is solved by the improved DALE with 3 agents.

$$\begin{bmatrix} A_1 \\ A_2 \\ A_3 \end{bmatrix} = \left[\begin{array}{ccc} 1 & -1 & 1 \\ \hline 1 & -1 & -1 \\ \hline 0 & 0 & 1 \end{array}\right], \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 3 \\ -3 \\ 3 \end{bmatrix} \tag{2.18}$$

$x^q_{\min}$ and the preferred point q are:

$$x^q_{\min} = \begin{bmatrix} 5.5 \\ 5.5 \\ 3 \end{bmatrix}, \quad q = \begin{bmatrix} 5 \\ 6 \\ 5 \end{bmatrix} \tag{2.19}$$

Here, we assume the three agents are all connected with each other with undirected edges. Let $V(t) = \sum_{i=1}^{3} |x_i(t) - x^q_{\min}|^2$. Then $V(t) = 0$ if and only if all $x_i(t) = x^q_{\min}$.

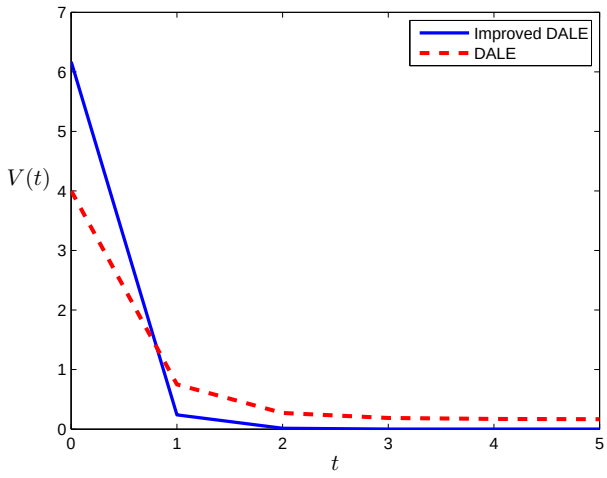

Figure 2.1. The improved DALE with special initializations achieves $x^q_{\min}$

Fig. 2.1 shows that the DALE in [12] does not achieve $x^q_{\min}$ while the modified DALE with the special initialization step (2.11) does.

We also provide Fig. 2 to illustrate the detailed process of the improved DALE to achieve $x^q_{\min}$. By $P_i x_i(0) = P_i q$ and $A_i x_i(0) = b_i$, each $x_i(0)$ is chosen as the closest

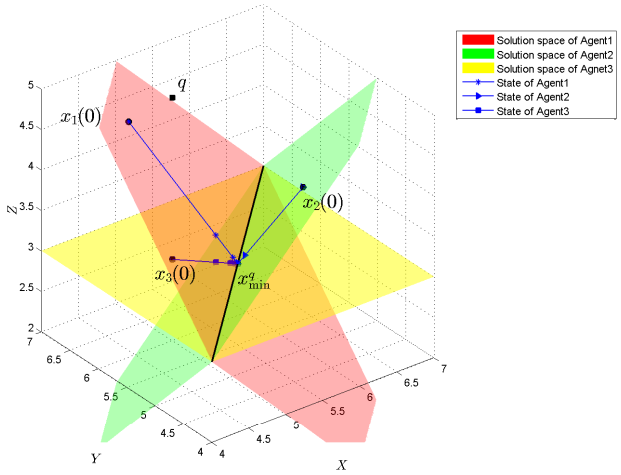

Figure 2.2. The process of each agent achieves $x^q_{\min}$ in 3D space

point to q subject to $A_i x = b_i$. All the $x_i(t)$ then converge to reach a consensus value which is $x^q_{\min}$ as in indicated in Fig. 2.2.

2.4 A Distributed Algorithm for minimum l_1 norm solution

2.4.1 The Problem

Note that if a linear equation $Ax = b$ has multiple solutions, except minimum l_2 norm, its sparse (minimum l_0-norm) solution is also of particular interest in many engineering applications. The examples include earthquake location detection [70], analysis of statistical data [71], solving biomagnetic inverse problems [72], compressive sensing, and so on. Challenged by the fact that the l_0-norm minimization problem is NP-hard [74], researchers usually turn to achieve solutions with minimum l_1-norm instead, for which the function to be minimized is convex; the obtained solution, is almost surely unique and equals the sparse (minimum l_0-norm) solution [75]. Existing results for achieving minimum l_1-norm solutions are usually based on the idea of LASSO [76], and they usually require a centralized coordinator and are not easily generalized to the distributed case. Existing results in distributed optimization are

also not directly applicable since they either assume all agents hold the same constraints [37] or different but compact constraints [38], and they typically require the weighting matrix associated with the network graph to be doubly stochastic [37, 38] or at least weighted balanced [41]. However, when solving under-determined equation sets via multi-agent networks, the local equations known by different agents can not be the same; the solution set to the local equation constraint is an affine subspace which is not compact; and as illustrated in [77], for a directed graph, additional co-operations among agents are usually required to guarantee its weighting matrix is doubly stochastic.

Motivated by these facts, in this section, we consider the modification of DALE [12], that can achieve minimum l_1 norm solution of a linear equation in a distributed manner. That is

$$x^* = \arg\min_{Ax=b} \|x\|_1, \tag{2.20}$$

2.4.2 Main Result

The update

As we know that the DALE developed in [12] is able to find a feasible solution to the linear equation, to further guide its results to be the minimum l_1-norm solution, we add the subgradient of $\|x\|_1$ subject to $A_i x = b_i$, namely, $P_i \text{sgn}\,(x_i(t))$, to (2.4) and have the following

$$x_i(t+1) = x_i(t) - P_i\left(x_i(t) - s_{ij}\sum_{j\in\mathcal{N}_i} x_j(t)\right) - \frac{P_i}{t+1}\text{sgn}\,(x_i(t)) \tag{2.21}$$

with $A_i x_i(0) = b_i$, $i = 1, 2, ..., m$. Please note that s_{ij} is the weight of the edge, for DALE, a particular choice of s_{ij} is $\frac{1}{d_i}$.

Remark 2.4.1 *Because $A_i x_i(0) = b_i$, and* image $P_i = \ker A_i$*, under the distributed update* (2.21)*, one has $A_i x_i(t) = b_i$ for $\forall t > 0$. Note that $\frac{1}{t+1}$ is introduced to adjust impact of the term $P_i\text{sgn}\,(x_i(t))$ to the original update* (2.4)*, a device which is*

commonly used in many distributed optimization algorithms [37]. This takes care of the fact that $P_i\text{sgn}\left(x_i(t)\right)$ *cannot be expected to tend to zero. Without such adjusted term* $\frac{1}{t+1}$*, we could never secure a consensus steady state solution* $x_i(t) = x^*$ *with* $Ax^* = b$*.*

Remark 2.4.2 *The update* (2.21) *is different from the algorithms proposed in [37, 38]. In update* (2.21)*, the gradient term* $\frac{1}{t+1}P_i\text{sgn}\left(x_i(t)\right)$ *is computed with respect to the current state* $x_i(t)$ *of agent* i*, thus, it is independent of the current round of communication. In [37,38], the gradient follows the form of* $\frac{1}{t+1}P_i\text{sgn}\left(s_{ij}\sum_{j\in\mathcal{N}_i}x_j(t)\right)$*, which is computed using a weighted average of agent* i*'s neighbors states.*

The distributed update (2.21) leads to the following result

Theorem 2.4.1 *Suppose the equation* $Ax = b$ *is under-determined with a unique minimum* l_1*-norm solution. Suppose the graph* $\mathbb{G}$ *of an associated* m*-agent network is directed and strongly connected, and its associated weighted adjacency matrix is row stochastic. Let each agent knows* A_i *and* b_i*. Initialize* $x_i(0)$ *such that* $A_ix_i(0) = b_i$*. Then, under the distributed update* (2.21)*, all* $x_i(t)$ *converge asymptotically to a constant given by* x^**, which is the unique minimum* l_1*-norm solution*[2] *to equation* $Ax = b$*.*

For the convenience of establishing Theorem 2.4.1, let $\boldsymbol{x}(t) = \text{col}\left\{x_1(t), \cdots, x_m(t)\right\}$ denote a stack of all $x_i(t)$; let $\bar{P} = \text{diag}\left\{P_1, \cdots, P_m\right\}$ denote a block-diagonal matrix with the ith diagonal block equal to P_i. Further let $\bar{S} = S \otimes I_n$, where $S \in \mathbb{R}^{m\times m}$ is a weighted adjacency matrix of the graph $\mathbb{G}$. Then based on equation (2.21), the evolution of all the states in the network can be rewritten in a compact form as

$$\boldsymbol{x}(t+1) = \bar{Q}\boldsymbol{x}(t) - \frac{1}{t+1}\bar{P}\text{sgn}\left(\boldsymbol{x}(t)\right) \tag{2.22}$$

where

$$\bar{Q} = I - \bar{P} + \bar{P}\bar{S}.$$

[2]Note that if the minimum l_1-norm solution is non-unique, the algorithm will converge to one of the minimum l_1-norm solutions.

To prove Theorem 2.4.1, it is sufficient to show that $\boldsymbol{x}(t) \to \boldsymbol{x}^*$, where $\boldsymbol{x}^* \triangleq \mathbf{1} \otimes x^*$. Towards this end, our proof is divided into three steps, which progressively lead to the fact that $\boldsymbol{x}(t) \to \boldsymbol{x}^*$. Firstly, for the purposes of proving the correctness of the algorithm, but not something computed in the course of executing the algorithm, we introduce a trajectory $\boldsymbol{z}(t)$ linked in a certain way to $\boldsymbol{x}(t)$. Then based on this $\boldsymbol{z}(t)$, we show that $\boldsymbol{x}(t) \to \boldsymbol{z}(t)$. Finally, we show that $\boldsymbol{z}(t) \to \boldsymbol{x}^*$. The details of these steps are provided in the following subsections.

Introducing a trajectory $\boldsymbol{z}(t)$.

For each time step t, define

$$\boldsymbol{z}(t,k) \triangleq \bar{Q}^k \boldsymbol{x}(t). \tag{2.23}$$

Then the following Proposition holds.

Proposition 2.4.1 *For each fixed t, as $k \to \infty$, the following limit of $\boldsymbol{z}(t,k)$ exists,*

$$\boldsymbol{z}(t) \triangleq \lim_{k\to\infty} \boldsymbol{z}(t,k) = \lim_{k\to\infty} \bar{Q}^k \boldsymbol{x}(t). \tag{2.24}$$

Moreover, $\bar{Q}\boldsymbol{z}(t) = \boldsymbol{z}(t)$ and for $\forall t$, there holds $\boldsymbol{z}(t) = \mathbf{1}_m \otimes z(t)$ where $z(t) \in \mathbb{R}^n$ is a solution to $Ax = b$.

To prove Proposition 2.4.1, we propose the following lemma, for which we provide the proof in the Appendix.

Lemma 2.4.1 *The following statements hold:*

(a) All eigenvalues of $\bar{P}\bar{S}$ have magnitude less than or equal to 1.

(b) $\lambda^\star = 1$ is the only eigenvalue of $\bar{P}\bar{S}$ with magnitude 1. It is non-defective and any corresponding eigenvector satisfies $\bar{P}\bar{S}\boldsymbol{u} = \bar{S}\boldsymbol{u} = \boldsymbol{u}$ where $\boldsymbol{u} = \mathbf{1}_m \otimes u$ and $u \in \ker A$.

Proof of Proposition 2.4.1: Let t be arbitrary but fixed. By definition (2.23), one has

$$\boldsymbol{z}(t,k+1) = \bar{Q}\boldsymbol{z}(t,k) = (I - \bar{P} + \bar{P}\bar{S})\boldsymbol{z}(t,k) \tag{2.25}$$

with $\boldsymbol{z}(t,0) = \boldsymbol{x}(t)$. Since $\bar{P} = \text{diag}\,\{P_1, \cdots, P_m\}$, and setting $\boldsymbol{z} = \text{col}\,\{z_1, \cdots, z_m\}$, then update (2.25) can be rewritten as

$$z_i(t,k+1) = z_i(t,k) - P_i\left(z_i(t,k) - s_{ij}\sum_{j\in\mathcal{N}_i} z_j(t,k)\right). \tag{2.26}$$

From update (2.26) and the fact that P_i is a projection matrix to $\ker A_i$, one has $A_i z_i(t,k+1) = A_i z_i(t,k)$. Since also $z_i(t,0) = x_i(t)$ is a solution to $A_i x_i = b_i$, one has for $\forall t, k$, $z_i(t,k)$ is a solution to $A_i z_i = b_i$.

To continue, define $\boldsymbol{z}^* \triangleq \mathbf{1}_m \otimes z^*$, where $z^* \in \mathbb{R}^n$ is an arbitrary solution to $Ax = b$. Since S is row stochastic, for any $z^* \in \mathbb{R}^n$, one has $\bar{S}\boldsymbol{z}^* = (S \otimes I_n)(\mathbf{1}_m \otimes z^*) = \boldsymbol{z}^*$. Further, define $\boldsymbol{\eta}(t,k) \triangleq \boldsymbol{z}(t,k) - \boldsymbol{z}^*$, where $\boldsymbol{\eta}(t,k) = \text{col}\,\{\eta_1(t,k), \cdots, \eta_m(t,k)\}$ and $\eta_i(t,k) = z_i(t,k) - z^*$ for all $i = 1, \cdots, m$. Recall that both $z_i(t,k)$ and z^* are solutions to $A_i z_i = b_i$; then $\eta_i(t,k) \in \ker A_i$. Because P_i is a projection matrix to $\ker A_i$, one has $P_i\eta_i(t,k) = \eta_i(t,k)$, that is $\bar{P}\boldsymbol{\eta}(t,k) = \boldsymbol{\eta}(t,k)$. Then, by subtracting $\boldsymbol{z}^*$ on both sides of (2.25), one has

$$\begin{aligned}
\boldsymbol{\eta}(t,k+1) &= \boldsymbol{\eta}(t,k) - (\bar{P} - \bar{P}\bar{S})\boldsymbol{\eta}(t,k) - (\bar{P} - \bar{P}\bar{S})\boldsymbol{z}^* \\
&= \boldsymbol{\eta}(t,k) - (\bar{P} - \bar{P}\bar{S})\boldsymbol{\eta}(t,k) \\
&= \bar{P}\bar{S}\boldsymbol{\eta}(t,k)
\end{aligned} \tag{2.27}$$

By Lemma 2.4.1, there exists a non-singular matrix T such that

$$\bar{P}\bar{S} = T\begin{bmatrix} I & 0 \\ 0 & R \end{bmatrix} T^{-1} \tag{2.28}$$

where all the eigenvalues of R are the eigenvalues of $\bar{P}\bar{S}$ with magnitude less than 1. Let

$$M = \lim_{k\to\infty} \left(\bar{P}\bar{S}\right)^k = T\begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} T^{-1} \tag{2.29}$$

Define $\boldsymbol{\eta}(t)^* = \lim_{k\to\infty} \boldsymbol{\eta}(t,k)$; then by update (2.27),

$$\boldsymbol{\eta}(t)^* = \lim_{k\to\infty} (\bar{P}\bar{S})^k \boldsymbol{\eta}(t,0) = T \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} T^{-1} \boldsymbol{\eta}(t,0) \tag{2.30}$$

Further, by the definition of $\boldsymbol{\eta}$, one has

$$\lim_{k\to\infty} \boldsymbol{z}(t,k) = \lim_{k\to\infty} \boldsymbol{\eta}(t,k) + \boldsymbol{z}^* = \boldsymbol{\eta}(t)^* + \boldsymbol{z}^* \tag{2.31}$$

Equation (2.31) verifies the existence of $\lim_{k\to\infty} \boldsymbol{z}(t,k)$, namely $\boldsymbol{z}(t)$ as defined in (2.24). As a consequence, $\bar{Q}\boldsymbol{z}(t) = \lim_{k\to\infty} \bar{Q}^{k+1}\boldsymbol{x}(t) = \boldsymbol{z}(t)$. To further show that $\boldsymbol{z}(t) = \mathbf{1}_m \otimes z(t)$ and for $\forall t$, $z(t)$ is a solution to $Ax = b$, recall from equations (2.28) and (2.30) that

$$\bar{P}\bar{S}\boldsymbol{\eta}(t)^* = T \begin{bmatrix} I & 0 \\ 0 & R \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} T^{-1} \boldsymbol{\eta}(t,0) = \boldsymbol{\eta}(t)^* \tag{2.32}$$

Thus, by Lemma 2.4.1 (b), one has $\boldsymbol{\eta}(t)^* = \mathbf{1}_m \otimes \eta(t)$ and for $\forall t$, $\eta(t) \in \ker A$. This, along with the definition of $\boldsymbol{z}^*$ yields

$$\boldsymbol{z}(t) = \boldsymbol{\eta}(t)^* + \boldsymbol{z}^* = \mathbf{1}_m \otimes (\eta(t) + z^*) = \mathbf{1}_m \otimes z(t) \tag{2.33}$$

Because $z^* \in \mathbb{R}^n$ is a solution to $Ax = b$ and $\eta(t) \in \ker A$, it follows that $z(t)$ is also a solution to $Ax = b$. This completes the proof. ■

Proof of $\boldsymbol{x}(t) \to \boldsymbol{z}(t)$.

Proposition 2.4.2 *For update* (2.22)*, given any initial* $\boldsymbol{x}(0)$ *and the* $\boldsymbol{z}(t)$ *defined in* (2.24)*, there always exists a positive constant* β *independent ot* t *such that*

$$\|\boldsymbol{x}(t) - \boldsymbol{z}(t)\|_2 \leq \frac{\beta}{t+1} \tag{2.34}$$

Proof of Proposition 2.4.2: Pre-multiply equation (2.22) on the left by $\bar{Q}^k$, leading to

$$\bar{Q}^k \boldsymbol{x}(t+1) = \bar{Q}^{k+1}\boldsymbol{x}(t) - \frac{1}{t+1}\bar{Q}^k \bar{P}\text{sgn}\left(\boldsymbol{x}(t)\right) \tag{2.35}$$

By taking $k \to \infty$ and recalling from definition (2.24) that $\boldsymbol{z}(t) = \lim_{k\to\infty} \bar{Q}^k \boldsymbol{x}(t) = \lim_{k\to\infty} \bar{Q}^{k+1} \boldsymbol{x}(t)$, one has

$$\boldsymbol{z}(t+1) = \boldsymbol{z}(t) - \frac{1}{t+1} \lim_{k\to\infty} \bar{Q}^k \bar{P} \text{sgn}\left(\boldsymbol{x}(t)\right) \tag{2.36}$$

Recall that $\bar{Q} = I - \bar{P} + \bar{P}\bar{S}$ and $\bar{P}^2 = \bar{P}$ (a property of projection matrices); then

$$\begin{aligned}
\lim_{k\to\infty} \bar{Q}^k \bar{P} &= \lim_{k\to\infty} (I - \bar{P} + \bar{P}\bar{S})^t \bar{P} \\
&= \lim_{k\to\infty} (I - \bar{P} + \bar{P}\bar{S})^{(k-1)} (I - \bar{P} + \bar{P}\bar{S}) \bar{P} \\
&= \lim_{k\to\infty} (I - \bar{P} + \bar{P}\bar{S})^{(k-1)} \bar{P}\bar{S}\bar{P} \\
&= \lim_{k\to\infty} (I - \bar{P} + \bar{P}\bar{S})^{(k-2)} (\bar{P}\bar{S})^2 \bar{P} \\
&= \lim_{k\to\infty} (\bar{P}\bar{S})^k \bar{P} = M\bar{P}
\end{aligned} \tag{2.37}$$

where M is defined in (2.29). Using this, equation (2.36) can be further written as

$$\boldsymbol{z}(t+1) = \bar{Q}\boldsymbol{z}(t) - \frac{1}{t+1} M\bar{P} \text{sgn}\left(\boldsymbol{x}(t)\right) \tag{2.38}$$

Define $\boldsymbol{e}(t) = \boldsymbol{x}(t) - \boldsymbol{z}(t)$. Subtracting (2.38) from (2.22) yields

$$\boldsymbol{e}(t+1) = \bar{Q}\boldsymbol{e}(t) - \frac{1}{t+1} (I - M)\bar{P} \text{sgn}\left(\boldsymbol{x}(t)\right) \tag{2.39}$$

Now, recall from Remark 2.4.1 and Proposition 2.4.1 that $x_i(t)$ and $z(t)$ are solutions to $A_i x_i = b_i$, which implies that $P_i(x_i(t) - z(t)) = x_i(t) - z(t)$, that is, $\bar{P}\boldsymbol{e}(t) = \boldsymbol{e}(t)$. Thus, by using (2.37) in reverse,

$$\begin{aligned}
M\boldsymbol{e}(t) = M\bar{P}\boldsymbol{e}(t) &= \lim_{k\to\infty} \bar{Q}^k \bar{P}\boldsymbol{e}(t) \\
&= \lim_{k\to\infty} \bar{Q}^k \boldsymbol{e}(t) = \lim_{k\to\infty} \bar{Q}^k (\boldsymbol{x}(t) - \boldsymbol{z}(t)) \\
&= \boldsymbol{z}(t) - \boldsymbol{z}(t) = 0
\end{aligned}$$

Bringing this, equations (2.28) and (2.29) into update (2.39), yields

$$\begin{aligned}
\boldsymbol{e}(t+1) &= (I - \bar{P} + \bar{P}\bar{S} - M)\boldsymbol{e}(t) - \frac{I-M}{t+1} \bar{P} \text{sgn}\left(\boldsymbol{x}(t)\right) \\
&= \bar{P}\bar{S}\boldsymbol{e}(t) - \frac{1}{t+1} (I - M)\bar{P} \text{sgn}\left(\boldsymbol{x}(t)\right) \\
&= T \begin{bmatrix} 0 & 0 \\ 0 & R \end{bmatrix} T^{-1} \boldsymbol{e}(t) - \frac{I-M}{t+1} \bar{P} \text{sgn}\left(\boldsymbol{x}(t)\right)
\end{aligned} \tag{2.40}$$

Since all eigenvalues of R have magnitude strictly less than one, there must exist a scalar $0 < \rho < 1$ such that

$$\|\boldsymbol{e}(t+1)\|_2 \leq \rho\|\boldsymbol{e}(t)\|_2 + \frac{c_e}{t+1} \tag{2.41}$$

where $c_e > 0$ is the upper bound of $\|(I-M)\bar{P}\text{sgn}\,(\boldsymbol{x}(t))\|_2$.

Now, given equation (2.41), for $t = 2\tau$, $\tau = 0, 1, 2, \cdots$, one has

$$\begin{aligned}
\|\boldsymbol{e}(2\tau)\|_2 &\leq \rho^{2\tau}\|\boldsymbol{e}(0)\|_2 + c_e \sum_{j=1}^{2\tau} \frac{\rho^{2\tau-j}}{j+1} \\
&= \rho^{2\tau}\|\boldsymbol{e}(0)\|_2 + c_e\rho^{\tau} \sum_{j=1}^{\tau} \frac{\rho^{\tau-j}}{j+1} + c_e \sum_{j=\tau+1}^{2\tau} \frac{\rho^{2\tau-j}}{j+1} \\
&\leq \rho^{2\tau}\|\boldsymbol{e}(0)\|_2 + c_e \frac{\rho^{\tau}}{1-\rho} + c_e \frac{1}{\tau+2}\frac{1}{1-\rho}
\end{aligned} \tag{2.42}$$

For $t = 2\tau + 1$, $\tau = 0, 1, 2, \cdots$, one has

$$\begin{aligned}
&\|\boldsymbol{e}(2\tau+1)\|_2 \leq \rho\|\boldsymbol{e}(2\tau)\|_2 + \frac{c_e}{2\tau+2} \\
&\leq \rho^{2k+1}\|\boldsymbol{e}(0)\|_2 + c_e \frac{\rho^{\tau+1}}{1-\rho} + \rho c_e \frac{1}{\tau+2}\frac{1}{1-\rho} + c_e \frac{1}{2\tau+2}
\end{aligned} \tag{2.43}$$

Since $\boldsymbol{e}(t) = \boldsymbol{x}(t) - \boldsymbol{z}(t)$, by combining equations (2.42) and (2.43), for $\forall t > 0$, it is evident there must exist a positive constant β such that (2.34) is true. This completes the proof. ■

Proof of $\boldsymbol{z}(t) \to \boldsymbol{x}^*$.

Proposition 2.4.3 *Suppose the minimum l_1-norm solution x^* to problem* (2.20) *is unique. Let $\boldsymbol{x}^* \triangleq \mathbf{1} \otimes x^*$. Then for update* (2.22)*, given any initial $\boldsymbol{x}(0)$, and the $\boldsymbol{z}(t)$ defined in* (2.24)*, one has*

$$\|\boldsymbol{z}(t) - \boldsymbol{x}^*\|_2 \to 0 \quad as \quad t \to \infty \tag{2.44}$$

Before proving Proposition 2.4.3, we define $\boldsymbol{x}^* = \mathbf{1}_m \otimes x^*$ where x^* is the unique minimum l_1-norm solution defined in (2.20). Let $\boldsymbol{\epsilon}(t) = \boldsymbol{z}(t) - \boldsymbol{x}^*$ and define a

function $V(\boldsymbol{\epsilon}(t)) \triangleq \boldsymbol{\epsilon}(t)^\top \boldsymbol{\epsilon}(t)$. Let $\bar{\Pi} \triangleq \Pi \otimes I_n \in \mathbb{R}^{mn \times mn}$ be a diagonal matrix, where $\Pi = \text{diag}\,\{\pi_1, \cdots, \pi_m\}$ and $[\pi_1 \ \pi_2 \ \cdots \ \pi_m] = \pi^\top = \lim_{k\to\infty} \left(\mathbf{1}_m^\top S^k\right) \in \mathbb{R}^{1\times m}$. Then, we introduce the following two lemmas to summarize some useful results with proofs provided in the Appendix.

Lemma 2.4.2 *The following statements hold:*

(a) The row vector $\pi^\top$ is a left eigenvector of S corresponding to eigenvalue 1, $\bar{\Pi}$ is positive definite and

$$\boldsymbol{\epsilon}(t)^\top M\bar{P}\text{sgn}\,(\boldsymbol{x}(t)) = \boldsymbol{\epsilon}(t)^\top \bar{\Pi}\ \text{sgn}\,(\boldsymbol{x}(t))$$

(b)
$$[\text{sgn}\,(\boldsymbol{x}(t))]^\top\, \bar{\Pi}(\boldsymbol{x}(t) - \boldsymbol{x}^*) \geq \|\bar{\Pi}\boldsymbol{x}(t)\|_1 - \|\bar{\Pi}\boldsymbol{x}^*\|_1.$$

(c)
$$\|\bar{\Pi}\boldsymbol{z}(t)\|_1 = \frac{\mathbf{1}_m^\top \pi}{m}\|\boldsymbol{z}(t)\|_1 \quad \textit{and} \quad \|\bar{\Pi}\boldsymbol{x}^*\|_1 = \frac{\mathbf{1}_m^\top \pi}{m}\|\boldsymbol{x}^*\|_1.$$

(d)
$$|\|\bar{\Pi}\boldsymbol{x}(t)\|_1 - \|\bar{\Pi}\boldsymbol{z}(t)\|_1| \leq \max\{\pi_i\}\frac{\sqrt{mn}\beta}{t+1},$$

where mn is the dimension of $\boldsymbol{x}(t)$ and $\boldsymbol{z}(t)$, with m the number of agents in the network, n the dimension of each $x_i(t)$, $z_i(t)$.

Lemma 2.4.3 *Suppose the minimum l_1-norm solution x^* of the linear equation $Ax = b$ is unique. Further, suppose $\|\boldsymbol{\epsilon}(t)\|_2 \leq \Delta$ is bounded, where $\boldsymbol{\epsilon}(t) = \boldsymbol{z}(t) - \boldsymbol{x}^*$, then there exists a positive constant α such that $\forall t > 0$,*

$$\|\boldsymbol{z}(t)\|_1 - \|\boldsymbol{x}^*\|_1 \geq \alpha\boldsymbol{\epsilon}(t)^\top\boldsymbol{\epsilon}(t) = \alpha V(\boldsymbol{\epsilon}(t)). \tag{2.45}$$

Proof of Proposition 2.4.3: By (2.37), subtracting $\boldsymbol{x}^*$ from both sides of equation (2.36) yields

$$\boldsymbol{\epsilon}(t+1) = \boldsymbol{\epsilon}(t) - \frac{1}{t+1}M\bar{P}\text{sgn}\,(\boldsymbol{x}(t)). \tag{2.46}$$

Based on (a) of Lemma 2.4.2, one has

$$\begin{aligned}
&V(\boldsymbol{\epsilon}(t+1))\\
=&\left[\boldsymbol{\epsilon}(t)-\frac{1}{t+1}M\bar{P}\text{sgn}\left(\boldsymbol{x}(t)\right)\right]^{\top}\left[\boldsymbol{\epsilon}(t)-\frac{1}{t+1}M\bar{P}\text{sgn}\left(\boldsymbol{x}(t)\right)\right]\\
=&V(\boldsymbol{\epsilon}(t))-\frac{2}{t+1}\boldsymbol{\epsilon}(t)^{\top}M\bar{P}\text{sgn}\left(\boldsymbol{x}(t)\right)+\frac{1}{(t+1)^2}\|M\bar{P}\text{sgn}\left(\boldsymbol{x}(t)\right)\|_2^2\\
\leq&V(\boldsymbol{\epsilon}(t))-\frac{2}{t+1}\boldsymbol{\epsilon}(t)^{\top}\bar{\Pi}\text{sgn}\left(\boldsymbol{x}(t)\right)+\frac{\psi}{(t+1)^2}\\
=&V(\boldsymbol{\epsilon}(t))-\frac{2}{t+1}\text{sgn}\left(\boldsymbol{x}(t)\right)^{\top}\bar{\Pi}(\boldsymbol{z}(t)-\boldsymbol{x}^*)+\frac{\psi}{(t+1)^2}\\
=&V(\boldsymbol{\epsilon}(t))-\frac{2}{t+1}\text{sgn}\left(\boldsymbol{x}(t)\right)^{\top}\bar{\Pi}(\boldsymbol{x}(t)-\boldsymbol{x}^*)\\
&\qquad-\frac{2}{t+1}\text{sgn}\left(\boldsymbol{x}(t)\right)^{\top}\bar{\Pi}(\boldsymbol{z}(t)-\boldsymbol{x}(t))+\frac{\psi}{(t+1)^2}\\
\leq&V(\boldsymbol{\epsilon}(t))-\frac{2}{t+1}\text{sgn}\left(\boldsymbol{x}(t)\right)^{\top}\bar{\Pi}(\boldsymbol{x}(t)-\boldsymbol{x}^*)+\frac{2\bar{\gamma}\beta}{(t+1)^2}+\frac{\psi}{(t+1)^2}
\end{aligned}\tag{2.47}$$

where ψ and $\bar{\gamma}$ are positive constants such that for all $\forall\boldsymbol{x}(t)$, $\|M\bar{P}\text{sgn}\left(\boldsymbol{x}(t)\right)\|_2^2\leq\psi$ and $\|\bar{\Pi}\text{sgn}\left(\boldsymbol{x}(t)\right)\|_2\leq\bar{\gamma}$ (the last inequality in (2.47) results from (2.34) of Proposition 2.4.2). Then, by bringing (b), (c) and (d) of Lemma 2.4.2 into equation (2.47), one has

$$\begin{aligned}
&V(\boldsymbol{\epsilon}(t+1))-V(\boldsymbol{\epsilon}(t))\\
\leq&-\frac{2}{t+1}\left(\|\bar{\Pi}\boldsymbol{x}(t)\|_1-\|\bar{\Pi}\boldsymbol{x}^*\|_1\right)+\frac{2\bar{\gamma}\beta}{(t+1)^2}+\frac{\psi}{(t+1)^2}\\
=&-\frac{2}{t+1}\big(\|\bar{\Pi}\boldsymbol{z}(t)\|_1-\|\bar{\Pi}\boldsymbol{x}^*\|_1-\big(\|\bar{\Pi}\boldsymbol{z}(t)\|_1-\|\bar{\Pi}\boldsymbol{x}(t)\|_1\big)\big)+\frac{2\bar{\gamma}\beta+\psi}{(t+1)^2}\\
\leq&-\frac{2}{t+1}\left(\|\bar{\Pi}\boldsymbol{z}(t)\|_1-\|\bar{\Pi}\boldsymbol{x}^*\|_1-\max\{\pi_i\}\frac{\sqrt{mn}\beta}{t}\right)+\frac{2\bar{\gamma}\beta+\psi}{t^2}\\
\leq&-\frac{2}{t+1}\frac{\mathbf{1}_m^{\top}\pi}{m}\left(\|\boldsymbol{z}(t)\|_1-\|\boldsymbol{x}^*\|_1\right)+\max\{\pi_i\}\frac{2\sqrt{mn}\beta}{(t+1)^2}+\frac{2\bar{\gamma}\beta+\psi}{(t+1)^2}\\
=&-\frac{2\gamma}{t+1}\left(\|\boldsymbol{z}(t)\|_1-\|\boldsymbol{x}^*\|_1\right)+\frac{\bar{\psi}}{(t+1)^2}
\end{aligned}\tag{2.48}$$

where $\gamma=\frac{\mathbf{1}_m^{\top}\pi}{m}$ and $\bar{\psi}=2\max\{\pi_i\}\sqrt{mn}\beta+2\bar{\gamma}\beta+\psi$.

To continue, since $\boldsymbol{z}(t) = \mathbf{1}_m \otimes z(t)$ and $\boldsymbol{x}^* = \mathbf{1}_m \otimes x^*$, where $\boldsymbol{z}(t)$ is a solution to $Ax = b$ and x^* is the unique minimum l_1-norm solution to $Ax = b$, one has $\|\boldsymbol{z}(t)\|_1 - \|\boldsymbol{x}^*\|_1 \geq 0$. This taken with (2.48) implies

$$V(\boldsymbol{\epsilon}(t+1)) \leq V(\boldsymbol{\epsilon}(t)) + \frac{\bar{\psi}}{(t+1)^2} \tag{2.49}$$

Since $\bar{\psi}$ is a constant, and $\sum_{t=1}^{\infty} \frac{1}{t^2} < \infty$, then $V(\boldsymbol{\epsilon}(t))$ must be bounded. Therefore, there exists a constant Δ such that $\|\boldsymbol{\epsilon}(t)\|_2 \leq \Delta$. Then, based on Lemma 2.4.3, by introducing (2.45) to (2.48), one has :

$$V(\boldsymbol{\epsilon}(t+1)) \leq (1 - \frac{2\gamma\alpha}{t+1})V(\boldsymbol{\epsilon}(t)) + \frac{\bar{\psi}}{(t+1)^2} \tag{2.50}$$

The inequality (2.50) can be 'solved' by writing it in summation form:

$$V(\boldsymbol{\epsilon}(t+1)) \leq \frac{\bar{\psi}}{(t+1)^2} + \sum_{\tau=2}^{t} \left[\frac{\bar{\psi}}{(\tau-1)^2} \prod_{k=\tau}^{t} (1 - \frac{2\gamma\alpha}{k}) \right] \tag{2.51}$$

Define $F(\tau_0, t) = \prod_{k=\tau_0}^{t} (1 - \frac{2\gamma\alpha}{k})$, where τ_0 is sufficiently large such that $0 < 1 - \frac{2\gamma\alpha}{k} < 1$ for $\forall k \geq \tau_0$. Then

$$\log F(\tau_0, t) = \sum_{k=\tau_0}^{t} \log(1 - \frac{2\gamma\alpha}{k})$$

Since $0 < 1 - \frac{2\gamma\alpha}{k} < 1$ for $\forall k \geq \tau_0$, it follows that $\log(1 - \frac{2\gamma\alpha}{k}) \leq -\frac{2\gamma\alpha}{k}$. Thus,

$$\begin{aligned} \log F(\tau_0, t) &\leq -2\gamma\alpha \sum_{k=\tau_0}^{t} \frac{1}{k} < -2\gamma\alpha \int_{k=\tau_0}^{t} \frac{1}{k+1} \\ &= -2\gamma\alpha \log\left(\frac{t+1}{\tau_0+1}\right). \end{aligned}$$

In addition, since $\tau \geq 2$ and $\prod_{k=\tau}^{\tau_0-1} (1 - \frac{2\gamma\alpha}{k})$ is a product of finite terms, with each term being bounded by $[1 - \gamma\alpha,\ 1]$, this product must be also bounded by a certain $\phi > 0$; then,

$$\begin{aligned} \prod_{k=\tau}^{t} (1 - \frac{2\gamma\alpha}{k}) &= \prod_{k=\tau}^{\tau_0-1} (1 - \frac{2\gamma\alpha}{k}) F(\tau_0, t) \\ &< \phi e^{-2\gamma\alpha \log(\frac{t+1}{\tau_0+2})} = \phi \left(\frac{\tau_0+2}{t+1}\right)^{2\gamma\alpha} \end{aligned} \tag{2.52}$$

Using equation (2.52) in (2.51), one has

$$V(\boldsymbol{\epsilon}(t+1)) < \frac{\bar{\psi}}{t^2} + \sum_{\tau=2}^{t} \left[\frac{\bar{\psi}}{(\tau-1)^2} \phi \left(\frac{\tau_0+2}{t+1} \right)^{2\gamma\alpha} \right]$$
$$= \frac{\bar{\psi}}{t^2} + \left(\frac{\tau_0+2}{t+1} \right)^{2\gamma\alpha} \cdot \phi \sum_{\tau=2}^{t} \frac{\psi}{(\tau-1)^2} \tag{2.53}$$

Since α, γ, ϕ and $\bar{\psi}$ are positive constants, as $t \to \infty$, one has $\frac{\bar{\psi}}{t^2} \to 0$, $\left(\frac{\tau_0+2}{t+1}\right)^{2\alpha} \to 0$ and $\phi \sum_{\tau=2}^{t} \frac{\bar{\psi}}{(\tau-1)^2}$ is bounded. Thus,

$$V(\boldsymbol{\epsilon}(t+1)) \to 0 \quad \text{as} \quad t \to \infty.$$

That is, $\boldsymbol{\epsilon}(t) = (\boldsymbol{z}(t) - \boldsymbol{x}^*) \to 0$ as $t \to \infty$. This completes the proof. ■

Proof of Theorem 2.4.1.

Based on the Proportion 2.4.2 and 2.4.3, one has for any initial $\boldsymbol{x}(0)$, update (2.22) will drive $\boldsymbol{x}(t) \to \boldsymbol{x}^*$ as $t \to \infty$. Equivalently, by update (2.21), the states $x_i(t), i = 1, \cdots, m$ in all agents will converge to x^*. This completes the proof. ■

2.4.3 Validation

Here, we describe the numerical simulations in MATLAB to validate the main result, noting a particular representative example. The simulations were conducted for a number of randomly generated networks with randomly generated linear equations. More precisely, we employ a directed, strongly connected network of $m = 16$ agents, where any two agents are connected with a probability of 0.4. Let $s_{ij} = \frac{1}{d_i}$. Let each agent knows a $A_i \in \mathbb{R}^{2\times 33}$ and $b_i \in \mathbb{R}^2$ with entries randomly selected from the interval $[0,\ 1]$. The equation set $A_i x = b_i$, $i = 1, \cdots, 16$ has a unique minimum l_1 norm solution x^* with probability 1. Define $V(t) \triangleq \|\boldsymbol{x}(t) - \boldsymbol{x}^*\|_2^2$, where $\boldsymbol{x}(t) = \text{col}\ \{x_1(t), \cdots, x_m(t)\}$ and $\boldsymbol{x}^* = \mathbf{1} \otimes x^*$.

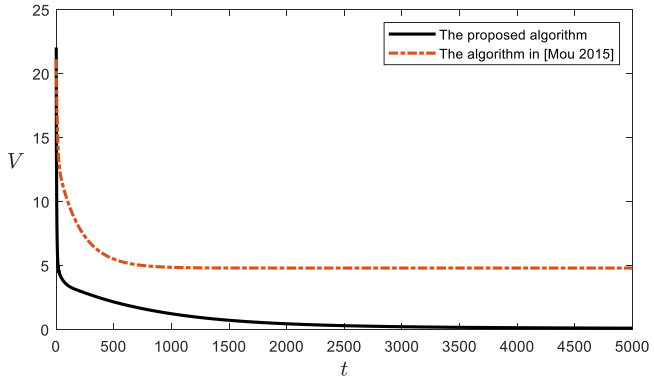

Figure 2.3. Convergence of the distributed update (2.21) under a directed, strongly connected network of 16 agents.

The curves shown in Figure 2.3 are generally representative of the convergence behavior in the vast majority of simulations we undertook but depict just one example. Comparisons are given for the algorithm proposed in this section and the one proposed in [12], as revealed by the results, the distributed update (2.21) is able to drive the states $x_i(t)$ in all agents to the unique minimum l_1 norm solution x^* of the equation set, which validates Theorem 2.4.1. On the contrary, the algorithm introduced in [12] for solving linear equations is not guaranteed to achieve x^*.

Appendix

Proof of Lemma 2.4.1

(**a**) Since S is a row stochastic matrix corresponding to the weighted adjacency matrix of a strongly connected graph, by the Perron-Frobenius theorem, S has a simple eigenvalue equal to 1 and all the other eigenvalues of S have magnitude strictly less than 1. Now let $\omega^\top = \begin{bmatrix}\omega_1, \cdots, \omega_m\end{bmatrix} \neq 0$ be a normalized left Perron-Frobenius eigenvector[3] of S. Let $\Omega = \text{diag}\ \{\omega_1, \cdots, \omega_m\}$ and $H = \Omega - S^\top \Omega S$. Since Ω is diagonal

[3]A Perron-Frobenius eigenvector is an eigenvector of S corresponding to the 1 eigenvalue, furthermore it is positive with entries summing to 1.

and positive; and all elements of S are non-negative, then all the off-diagonal elements of H are non-positive. Further note that the row sums of H have the property:

$$\begin{aligned} H \cdot \mathbf{1}_m &= (\Omega - S^\top \Omega S)\mathbf{1}_m \\ &= \pi - S^\top \Omega \mathbf{1}_m \\ &= \pi - S^\top \pi = 0 \end{aligned}$$

Thus, H must be a Laplacian matrix of a certain graph so that $H = \Omega - S^\top \Omega S \geq 0$. Recall that the graph $\mathbb{G}$ for our problem is strongly connected, which means $\pi_i > 0$, thus, Ω is positive. By left/right multiplying by $\Omega^{-1/2}$ in the expression for H, one has

$$\Omega^{-1/2} S^\top \Omega S \Omega^{-1/2} \leq I_m. \tag{2.54}$$

It follows that

$$\sigma_{\max}(\bar{\Omega}^{-1/2} \bar{S}^\top \bar{\Omega} \bar{S} \bar{\Omega}^{-1/2}) \leq 1 \tag{2.55}$$

where $\bar{\Omega} = \Omega \otimes I_n$, $\bar{S} = S \otimes I_n$ and $\sigma_{\max}(\cdot)$ denotes the largest singular value of a matrix. Note that P_i is the projection to $\ker A_i$ and $\bar{P} = \text{diag}\,\{P_1, \cdots, P_m\}$, then one must have $\sigma_{\max}(P) \leq 1$. This, along with (2.55), leads to

$$\sigma_{\max}(\bar{P} \bar{\Omega}^{-1/2} \bar{S}^\top \bar{\Omega} \bar{S} \bar{\Omega}^{-1/2} \bar{P}) \leq 1$$

Furthermore, since $\bar{\Omega} = \Omega \otimes I_n$ where Ω is a diagonal matrix and $\bar{P}$ is a block-diagonal matrix with each block $P_i \in \mathbb{R}^{n \times n}$, then one has $\bar{P}\bar{\Omega}^{-1/2} = \bar{\Omega}^{-1/2}\bar{P}$. That is

$$\sigma_{\max}\left((\bar{\Omega}^{-1/2} \bar{P} \bar{S}^\top \bar{\Omega}^{1/2})(\bar{\Omega}^{1/2} \bar{S} \bar{P} \bar{\Omega}^{-1/2})\right) \leq 1 \tag{2.56}$$

This indicates that all the eigenvalues of $\bar{\Omega}^{1/2} \bar{S} \bar{P} \bar{\Omega}^{-1/2}$ has magnitude less than or equal to one. Since for all the eigenvalues, we have $\lambda(\bar{P}\bar{S}) = \lambda(\bar{S}\bar{P}) = \lambda(\bar{\Omega}^{1/2} \bar{S} \bar{P} \bar{\Omega}^{-1/2})$, this completes the proof of statement (a).

(**b**) Note that the equality $|\lambda(\bar{P}\bar{S})| = 1$ holds if and only if there exist a vector $\boldsymbol{u} \neq 0$ such that

$$\bar{P}\bar{S}\boldsymbol{u} = \lambda^\star \boldsymbol{u} \quad \text{with} \quad |\lambda^\star| = 1 \tag{2.57}$$

Thus, $\bar{S}\boldsymbol{u} \neq 0$ and

$$\bar{\Omega}^{1/2}\bar{S}\bar{P}\bar{\Omega}^{-1/2}\bar{\Omega}^{1/2}\bar{S}\boldsymbol{u} = \lambda^{\star}\bar{\Omega}^{1/2}\bar{S}\boldsymbol{u}. \tag{2.58}$$

Let $\boldsymbol{q} = \bar{\Omega}^{1/2}\bar{S}\boldsymbol{u}$, since $\bar{P}\bar{\Omega}^{-1/2} = \bar{\Omega}^{-1/2}\bar{P}$, then (2.58) can be rewritten as

$$\bar{\Omega}^{1/2}\bar{S}\bar{\Omega}^{-1/2}\bar{P}\boldsymbol{q} = \lambda^{\star}\boldsymbol{q} \tag{2.59}$$

The equality of (2.59) holds only if

$$\|\bar{\Omega}^{1/2}\bar{S}\bar{\Omega}^{-1/2}\bar{P}\boldsymbol{q}\| = \|\lambda^{\star}\boldsymbol{q}\| = |\lambda^{\star}| \cdot \|\boldsymbol{q}\| = \|\boldsymbol{q}\| \tag{2.60}$$

Recall that $\sigma_{\max}(\bar{P}) \leq 1$ and $\sigma_{\max}(\bar{\Omega}^{1/2}\bar{S}\bar{\Omega}^{-1/2}) \leq 1$, thus,

$$\|\bar{\Omega}^{1/2}\bar{S}\bar{\Omega}^{-1/2}\bar{P}\boldsymbol{q}\| \leq \|\bar{P}\boldsymbol{q}\| \leq \|\boldsymbol{q}\|. \tag{2.61}$$

Then, (2.60) holds if and only if $\|\bar{P}\boldsymbol{q}\| = \|\boldsymbol{q}\|$. Further, recall that $\bar{P}$ is a projection matrix; then additionally

$$\bar{P}\boldsymbol{q} = \boldsymbol{q} \tag{2.62}$$

Bringing (2.62) into (2.59) leads to

$$\bar{\Omega}^{1/2}\bar{S}\bar{\Omega}^{-1/2}\boldsymbol{q} = \lambda^{\star}\boldsymbol{q}, \quad \text{with} \quad |\lambda^{\star}| = 1. \tag{2.63}$$

Note that $\bar{\Omega}^{1/2}\bar{S}\bar{\Omega}^{-1/2}$ and $\bar{S}$ are similar matrices with identical eigenvalues. Further, recall the definition of $\bar{S} = S \otimes I_n$ and the fact that $|\lambda(S)| = 1$ if and only if $\lambda(S) = 1$, thus, one has $\lambda^{\star} = 1$. Bringing this into equation (2.57) leads to

$$\bar{P}\bar{S}\boldsymbol{u} = \boldsymbol{u} \tag{2.64}$$

To continue, recall that $\boldsymbol{q} = \bar{\Omega}^{1/2}\bar{S}\boldsymbol{u}$ and $\bar{P}\bar{\Omega}^{1/2} = \bar{\Omega}^{1/2}\bar{P}$. Then equation (2.62) implies

$$\bar{\Omega}^{1/2}\bar{P}\bar{S}\boldsymbol{u} = \bar{P}\bar{\Omega}^{1/2}\bar{S}\boldsymbol{u} = \bar{\Omega}^{1/2}\bar{S}\boldsymbol{u} \tag{2.65}$$

Since $\bar{\Omega}^{1/2}$ is positive definite, one has

$$\bar{P}\bar{S}\boldsymbol{u} = \bar{S}\boldsymbol{u} \tag{2.66}$$

Equations (2.64) and (2.66) implies

$$\bar{S}\boldsymbol{u} = \boldsymbol{u}. \tag{2.67}$$

Thus, $\boldsymbol{u} = \mathbf{1}_m \otimes u$, $u \in \mathbb{R}^n$. Bringing (2.67) back to (2.64) yields $\bar{P}\boldsymbol{u} = \boldsymbol{u}$. This, along with $\boldsymbol{u} = \mathbf{1}_m \otimes u$ implies that $u \in \bigcap_{i=1}^{m} \ker A_i = \ker A$. Since $Ax = b$ is under-determined, $\ker A \neq \emptyset$ and such u exists. Till now, we have validated that $|\lambda(\bar{P}\bar{S})| = 1$, which happens if and only if $\bar{P}\bar{S}\boldsymbol{u} = \lambda^\star \boldsymbol{u}$ with $\boldsymbol{u} = \mathbf{1}_m \otimes u,\ u \in \ker A$, and $\lambda^\star = 1$.

Now, we prove that the eigenvalues of $\bar{P}\bar{S}$ equal to 1 must be non-defective by contradiction. Suppose the matrix $\bar{P}\bar{S}$ has defective eigenvalues equal to 1. Then so must the matrix $\bar{\Omega}^{1/2}\bar{S}\bar{P}\bar{\Omega}^{-1/2}$ since they are similar. From the definition of defective eigenvalues, there always exist vectors $\boldsymbol{v}_1 \neq \boldsymbol{v}_2$, with $\|\boldsymbol{v}_1\| = \|\boldsymbol{v}_2\| = 1$, and $\boldsymbol{v}_1^\top \boldsymbol{v}_2 \geq 0$ such that

$$(\bar{\Omega}^{1/2}\bar{S}\bar{P}\bar{\Omega}^{-1/2} - I)\boldsymbol{v}_1 = 0 \tag{2.68}$$

$$(\bar{\Omega}^{1/2}\bar{S}\bar{P}\bar{\Omega}^{-1/2} - I)\boldsymbol{v}_2 = \boldsymbol{v}_1 \tag{2.69}$$

Since $\|\boldsymbol{v}_1\| = \|\boldsymbol{v}_2\| = 1$ and $\boldsymbol{v}_1 \neq \boldsymbol{v}_2$, one can always find a vector $\|\hat{\boldsymbol{v}}\| = 1$ such that $\hat{\boldsymbol{v}} = r_1\boldsymbol{v}_1 + r_2\boldsymbol{v}_2$ with $r_1 + r_2 > 1$. Then one has

$$\begin{aligned} \frac{\|\bar{\Omega}^{1/2}\bar{S}\bar{P}\bar{\Omega}^{-1/2}\hat{\boldsymbol{v}}\|}{\|\hat{\boldsymbol{v}}\|} &= \|\bar{\Omega}^{1/2}\bar{S}\bar{P}\bar{\Omega}^{-1/2}(r_1\boldsymbol{v}_1 + r_2\boldsymbol{v}_2)\| \\ &= \|(r_1 + r_2)\boldsymbol{v}_1 + r_2\boldsymbol{v}_2\| \end{aligned} \tag{2.70}$$

Recall that $r_1 > 0$, $r_2 > 0$ and $\boldsymbol{v}_1^\top \boldsymbol{v}_2 \geq 0$, then

$$\|(r_1 + r_2)\boldsymbol{v}_1 + r_2\boldsymbol{v}_2\| \geq \|(r_1 + r_2)\boldsymbol{v}_1\| = r_1 + r_2 > 1.$$

This indicates $\sigma_{\max}\left(\bar{\Omega}^{1/2}\bar{S}\bar{P}\bar{\Omega}^{-1/2}\right) > 1$, which contradicts with equation (2.56). Thus, the eigenvalues of $\bar{P}\bar{S}$ equal to 1 must be non-defective. This completes the proof. ■

Proof of Lemma 2.4.2

(**a**) Since the matrix S is row stochastic, it has an eigenvalue equal to 1. Let $\pi^\top =$

$\lim_{k\to\infty}\left(\mathbf{1}_m^\top S^k\right)$; then one has $\pi^\top S = \lim_{k\to\infty}\left(\mathbf{1}_m^\top S^{k+1}\right) = \pi^\top$. Thus, $\pi^\top$ is a left eigenvector of S corresponding to eigenvalue 1. Further since S is primitive (due to the strong connectedness of the graph), then by the Perron-Frobenius theorem [78] , all entries of π are positive and the corresponding eigenvalue at 1 is simple.

Now, recall that $\boldsymbol{z}(t) = \mathbf{1}_m \otimes z(t)$, $\boldsymbol{x}^* = \mathbf{1}_m \otimes x^*$, thus, $\boldsymbol{\epsilon}(t) = \boldsymbol{z}(t) - \boldsymbol{x}^* = \mathbf{1}_m \otimes (z(t) - x^*) = \mathbf{1}_m \otimes \epsilon(t)$. Furthermore, since both $z(t)$ and x^* are solutions to $Ax = b$, one has $P_i\epsilon(t) = \epsilon(t)$, that is, $\left(v^\top \otimes \epsilon(t)^\top\right)\bar{P} = \left(v^\top \otimes \epsilon(t)^\top\right)$, for any $v \in \mathbb{R}^m$. With this in mind, by the definition of M in (2.29), one has,

$$\begin{aligned}
&\boldsymbol{\epsilon}(t)^\top M\bar{P}\text{sgn}\left(\boldsymbol{x}(t)\right)\\
&= \lim_{k\to\infty}\left(\mathbf{1}_m^\top \otimes \epsilon(t)^\top\right)(\bar{P}\bar{S})^k\bar{P}\text{sgn}\left(\boldsymbol{x}(t)\right)\\
&= \lim_{k\to\infty}\left(\mathbf{1}_m^\top \otimes \epsilon(t)^\top\right)(S\otimes I_n)(\bar{P}\bar{S})^{(k-1)}\bar{P}\text{sgn}\left(\boldsymbol{x}(t)\right)\\
&= \lim_{k\to\infty}\left(\mathbf{1}_m^\top S \otimes \epsilon(t)^\top\right)(\bar{P}\bar{S})^{(k-1)}\bar{P}\text{sgn}\left(\boldsymbol{x}(t)^\top\right)\\
&= \lim_{k\to\infty}\left(\mathbf{1}_m^\top S^k \otimes \epsilon(t)^\top\right)\bar{P}\text{sgn}\left(\boldsymbol{x}(t)\right)\\
&= \left(\pi^\top \otimes \epsilon(t)^\top\right)\text{sgn}\left(\boldsymbol{x}(t)\right)\\
&= \boldsymbol{\epsilon}(t)^\top\bar{\Pi}\ \text{sgn}\left(\boldsymbol{x}(t)\right)
\end{aligned} \tag{2.71}$$

This completes the proof.

(**b**) Consider a function $G(\boldsymbol{\xi}) = \|\bar{\Pi}\boldsymbol{\xi}\|_1$, where $\boldsymbol{\xi} \in \mathbb{R}^{mn}$. $G(\boldsymbol{\xi})$ is convex because the l_1-norm is convex. It follows that

$$\begin{aligned}
&\left[\text{sgn}\left(\bar{\Pi}\boldsymbol{\xi}_1\right)\right]^\top\bar{\Pi}(\boldsymbol{\xi}_1 - \boldsymbol{\xi}_2) = \left[\left.\frac{\partial G(\boldsymbol{\xi})}{\partial\boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=\boldsymbol{\xi}_1}\right]^\top(\boldsymbol{\xi}_1 - \boldsymbol{\xi}_2)\\
&\geq G(\boldsymbol{\xi}_1) - G(\boldsymbol{\xi}_2) = \|\bar{\Pi}\boldsymbol{\xi}_1\|_1 - \|\bar{\Pi}\boldsymbol{\xi}_2\|_1
\end{aligned}$$

Recall that $\bar{\Pi}$ is a diagonal matrix with all entries being positive, which means $\text{sgn}\left(\bar{\Pi}\boldsymbol{\xi}_1\right) = \text{sgn}\left(\boldsymbol{\xi}_1\right)$; then by letting $\boldsymbol{x}(t) = \boldsymbol{\xi}_1$ and $\boldsymbol{x}^* = \boldsymbol{\xi}_2$ one has

$$\left[\text{sgn}\left(\boldsymbol{x}(t)\right)\right]^\top\bar{\Pi}(\boldsymbol{x}(t) - \boldsymbol{x}^*) \geq \|\bar{\Pi}\boldsymbol{x}(t)\|_1 - \|\bar{\Pi}\boldsymbol{x}^*\|_1 \tag{2.72}$$

This completes the proof.

(**c**) Since $\boldsymbol{z}(t) = \mathbf{1}_m \otimes z(t)$, $\boldsymbol{x}^* = \mathbf{1}_m \otimes x^*$, by the definition of $\bar{\Pi}$, one has

$$\begin{aligned}\|\bar{\Pi}\boldsymbol{z}(t)\|_1 &= \| \left(\text{diag}\left\{\pi_1,\ \pi_2,\ \cdots,\ \pi_m\right\} \otimes I_n\right)\left(\mathbf{1}_m \otimes z(t)\right) \|_1 \\ &= \|\text{col}\left\{\pi_1,\ \pi_2,\ \cdots,\ \pi_m\right\} \otimes z(t)\|_1 = \|\pi \otimes z(t)\|_1 \\ &= \mathbf{1}_m^\top \pi \|z(t)\|_1 = \frac{\mathbf{1}_m^\top \pi}{m} \|\boldsymbol{z}(t)\|_1 \end{aligned} \tag{2.73}$$

Similarly,

$$\|\bar{\Pi}\boldsymbol{x}^*\|_1 = \frac{\mathbf{1}_m^\top \pi}{m} \|\boldsymbol{x}^*\|_1 \tag{2.74}$$

This completes the proof.

(**d**) Let $[\ \cdot\ ]_j$ denote the jth entry of a vector. Using the Cauchy-Schwarz inequality, one has

$$\begin{aligned} &\left|\|\bar{\Pi}\boldsymbol{x}(t)\|_1 - \|\bar{\Pi}\boldsymbol{z}(t)\|_1\right| = \left|\sum_{j=1}^{mn} |[\bar{\Pi}\boldsymbol{x}(t)]_j| - \sum_{j=1}^{mn} |[\bar{\Pi}\boldsymbol{z}(t)]_j|\right| \\ =& \sum_{j=1}^{mn} \left||[\bar{\Pi}\boldsymbol{x}(t)]_j| - |[\bar{\Pi}\boldsymbol{z}(t)]_j|\right| \le \sum_{j=1}^{mn} \left|[\bar{\Pi}\boldsymbol{x}(t)]_j - [\bar{\Pi}\boldsymbol{z}(t)]_j\right| \\ =& \sum_{j=1}^{mn} \left|[\bar{\Pi}\boldsymbol{e}(t)]_j\right| \le \max\{\pi_i\} \sum_{j=1}^{mn} \left|[\boldsymbol{e}(t)]_j\right| \cdot 1 \\ \le& \max\{\pi_i\} \left(\sum_{j=1}^{mn} \left|[\boldsymbol{e}(t)]_j\right|^2\right)^{\frac{1}{2}} \left(\sum_{j=1}^{mn} 1^2\right)^{\frac{1}{2}} \\ =& \max\{\pi_i\}\sqrt{mn}\|\boldsymbol{e}(t)\|_2 = \max\{\pi_i\}\frac{\sqrt{mn}\beta}{t+1} \end{aligned}$$

This completes the proof. ■

Proof of Lemma 2.4.3

If $\boldsymbol{z}(t) - \boldsymbol{x}^* = \boldsymbol{\epsilon}(t) = 0$, then $V(\boldsymbol{\epsilon}(t)) = 0$. Thus, (2.45) holds. If $\boldsymbol{\epsilon}(t) \neq 0$, recall that $\boldsymbol{z}(t) = \mathbf{1}_m \otimes z(t)$, $\boldsymbol{x}^* = \mathbf{1}_m \otimes x^*$, then, $\boldsymbol{\epsilon}(t) = \boldsymbol{z}(t) - \boldsymbol{x}^* = \mathbf{1}_m \otimes (z(t) - x^*) = \mathbf{1}_m \otimes \epsilon(t)$. Since both $z(t)$ and x^* are solutions to $Ax = b$, one has $\epsilon(t) \in \ker A$. Let κ be a positive scalar whose value can be made arbitrarily small. Let

$$h \triangleq \kappa \cdot \frac{\epsilon(t)}{\|\epsilon(t)\|_1}, \tag{2.75}$$

obviously, $\|h\|_1 = \kappa$ and $h \in \ker A$. Since $\|\cdot\|_1$ is piece-wise linear, and given that x^* is fixed, then κ can always be chosen small enough such that for any h in the form of (2.75), the function value of $\|x\|_1$ varies linearly along the line segment $\{x^* + \mu h \mid \mu \in (0, 1]\}$. That is, for $y = \mu h$ with $\mu \in (0, 1]$, one has sgn $(x^* + y) =$ sgn $(x^* + h)$ and

$$\|x^* + y\|_1 - \|x^*\|_1 = \text{sgn } (x^* + h)^\top y. \tag{2.76}$$

Recall that $\frac{y}{\|y\|_1} = \frac{h}{\|h\|_1}$, then equation (2.76) can be further written as

$$\|x^* + y\|_1 - \|x^*\|_1 = \eta(x^*, h)\|y\|_1. \tag{2.77}$$

where

$$\eta(x^*, h) = \frac{\text{sgn } (x^* + h)^\top h}{\|h\|_1} \in \mathbb{R}. \tag{2.78}$$

Since $y \in \ker A$ and x^* is the unique minimum l_1-norm solution to $Ax = b$, from (2.77), one has $\eta(x^*, h) > 0$. Further since the h in (2.78) is chosen from a compact set such that $h \in \ker A$ and $\|h\|_1 = \kappa$, the values of $\eta(x^*, h)$ must have a lower bound $\hat{\eta}$, which is strictly positive. Thus, from (2.77), one has

$$\|x^* + y\|_1 - \|x^*\|_1 \geq \hat{\eta}\|y\|_1. \tag{2.79}$$

Based on inequality (2.79), we will consider two possibilities, depending on the magnitude of $\|\epsilon(t)\|_1$. First, if $\|\epsilon(t)\|_1 \leq \kappa$, that is $\frac{\|\epsilon(t)\|_1}{\kappa} \leq 1$, we let $\mu = \frac{\|\epsilon(t)\|_1}{\kappa}$, which leads to $y = \mu h = \frac{\|\epsilon(t)\|_1}{\kappa} \cdot \kappa \frac{\epsilon(t)}{\|\epsilon(t)\|_1} = \epsilon(t)$. Then from (2.79), one has

$$\|z(t)\|_1 - \|x^*\|_1 = \|x^* + \epsilon(t)\|_1 - \|x^*\|_1 \geq \hat{\eta}\|\epsilon(t)\|_1. \tag{2.80}$$

Second, if $\|\epsilon(t)\|_1 > \kappa$, it follows (because of (2.75) and $y = \mu h$) that $\epsilon(t) = \frac{\|\epsilon(t)\|_1}{\mu\kappa} y$, where $\frac{\|\epsilon(t)\|_1}{\mu\kappa} > 1$. Then due to the convexity of $\|\cdot\|_1$, one has

$$\begin{aligned}
\|z(t)\|_1 - \|x^*\|_1 &= \left\| x^* + \frac{\|\epsilon(t)\|_1}{\mu\kappa} y \right\|_1 - \|x^*\|_1 \\
&\geq \frac{\|\epsilon(t)\|_1}{\mu\kappa} \left(\|x^* + y\|_1 - \|x^*\|_1 \right) \geq \frac{\|\epsilon(t)\|_1}{\mu\kappa} \hat{\eta}\|y\|_1 \\
&= \hat{\eta}\|\epsilon(t)\|_1.
\end{aligned} \tag{2.81}$$

where the last equality follows because $y = \mu h$ and $\|h\|_1 = \kappa$. Then equations (2.80) and (2.81), along with the fact that $\boldsymbol{z}(t) = \mathbf{1}_m \otimes z(t)$, $\boldsymbol{x}^* = \mathbf{1}_m \otimes x^*$, lead to

$$\|\boldsymbol{z}(t)\|_1 - \|\boldsymbol{x}^*\|_1 \geq \hat{\eta}\|\boldsymbol{\epsilon}(t)\|_1.$$

Recall that $\|\boldsymbol{\epsilon}(t)\|_1 \geq \|\boldsymbol{\epsilon}(t)\|_2$ [79] and $\|\boldsymbol{\epsilon}(t)\|_2 \leq \Delta$, then

$$\|\boldsymbol{z}(t)\|_1 - \|\boldsymbol{x}^*\|_1 \geq \hat{\eta}\frac{\|\boldsymbol{\epsilon}(t)\|_2^2}{\|\boldsymbol{\epsilon}(t)\|_2} \geq \frac{\hat{\eta}}{\Delta}V(t)$$

Let $\alpha = \dfrac{\hat{\eta}}{\Delta}$. This completes the proof. ■

3. SCALABLE, DISTRIBUTED ALGORITHMS FOR SOLVING LINEAR EQUATIONS VIA DOUBLE-LAYERED NETWORKS[1]

3.1 Introduction

In Chapter 2, we have introduced the consensus-based distributed algorithms for solving linear equations. In this chapter, we introduce another type of coordination that can further improve the scalability of distributed algorithms for solving linear equations. Actually, elegant and powerful as the idea of consensus is, there has been rather limited application of consensus-based algorithms into situations when coordination among agents requires more than reaching consensus, especially when conservation requirements are involved. Different from consensus, *conservation* is a constraint that the sum of certain functions of agents' states needs to be constant [31]. Various types of conservation arise in many engineering applications, including conservation of resources in distributed allocation [80], conservation of total energy in controlling hybrid vehicles [81], conservation of flows in traffic control [82], conservation of linear and angular momentum in formation control, and so on. Recognition of the potential of conservation in complementing consensus has motivated us to integrate both consensus and conservation together in one framework with the goal of combining their advantages together for achieving efficiency and scalability of distributed coordination.

One natural way to achieve such integration is by layered coordination [83, 84], which has been proven to be a powerful tool in many similar situations. For example, practical tasks involving a large number of robots can be achieved by coordination in the planning layer, executive layer and/or behavior layer [85]. Complicated optimiza-

[1]Resarch in this section has been published in the papers [6] with me as the leading author.

tion problems can be solved by coordination through layers, each of which iterates on its own subsets of decision variables using local information to achieve individual optimality [86]. Deep learning algorithms can be established by grouping neural nodes into multiple layers to achieve different functions including feature extraction, collection, comparison, and fusion [87]. Compared with single-layered networks, double-layered networks provide a natural description for quantifying the interconnectivity between different categories of connections [88], improve accuracy [89] and efficiency [90]. This has motivated us to employ double-layered frameworks for the integration of consensus and conservation.

In this chapter, we consider a double-layered multi-agent network composed of clusters where each cluster consists of one aggregator and a network of agents (as will be detailed later in Fig. 3.1). Double-layered structures have played a significant role in distributed consensus [91–95] as well as distributed synchronization under communication delays or disturbances [96–98]. By selecting one or more agents as leaders from each cluster and sparsely connecting these leaders, clusters are able to coordinate for achieving unconstrained consensus [93–95]. This approach is, however, not directly applicable to developing distributed linear equation solvers, the goal of which is not only to achieve a consensus but a consensus subject to a group of linear constraints. Thus, we introduce one aggregator in each cluster, which has no computational burden but is able to collect and distribute information within the cluster. Information exchange between clusters is achieved through the upper-level network of aggregators. Such a technique of mixed use of heterogeneous agents has proved its efficiency in mobile communication networks [99]. The double-layered framework enables achieving two different types of coordination simultaneously, namely, one layer taking care of consensus while the other for conservation. This allows us to develop distributed algorithms for solving linear equations, which outperform existing distributed linear equation solvers [12, 13, 16, 62], in a number of aspects, including but not limited to the following:

- each agent does not have to know as much as a complete row or column of the overall equation;
- each agent only needs to control as few as two scalar states when the number of clusters and the number of agents in each cluster (rather than in the whole network), are equal to the number of rows and columns of the overall equation, respectively;
- the dimensions of agents' state vectors do not have to be the same, which is in contrast to algorithms based on the idea of standard consensus. These consensus-based algorithms inherently require all agents' states to be of the same dimension [12, 13, 16, 62];

3.2 Problem Formulation

Consider a double-layered multi-agent network consisting of a number of c clusters. Each cluster i is composed of one aggregator denoted by i and a number of c_i agents denoted by $i_1, i_2, \cdots, i_{c_i}$. An aggregator is able to collect and distribute information with all agents in the same cluster. Further, we suppose each aggregator i is also able to receive information from certain other aggregators which are called i's *aggregator-neighbors* and denoted by $\mathcal{N}_i$; also, we assume $i \in \mathcal{N}_i$. The neighbor relations of aggregators can be characterized by a c-node graph $\mathbb{G}$, in which there is an arc from $\bar{i}$ to i if and only if $\bar{i} \in \mathcal{N}_i$. Here, aggregators, which are not required to perform any computations, are introduced only for the purpose of information exchange among clusters including collecting information and distributing information within the cluster where the aggregator is located. In order to minimize the information overheads and the work load of aggregators, we do **not** require all information in a cluster to be shared with other clusters through communications between aggregators, as will be seen later, in Remark 3.3.1 and 3.4.1. Within each cluster i, each agent i_j, $j = 1, \cdots, c_i$, is able to receive information from certain agents, which are called agent i_j's *agent-neighbors* denoted by $\mathcal{N}_{ij}$, specially $i_j \in \mathcal{N}_{ij}$. The neighbor

relations within cluster i can be characterized by a c_i-node graph $\mathbb{G}_i$, in which there is an arc from $i_{\bar{j}}$ to i_j if and only if $i_{\bar{j}} \in \mathcal{N}_{ij}$. Suppose all $\mathbb{G}_i$, $i = 1, 2, \cdots, c$ and $\mathbb{G}$ are connected and bidirectional[2]. One example of such a double-layer multi-agent network is shown in Fig. 4.1. Consistently with the description above, we emphasize that no aggregator is a node of $\mathbb{G}_i$ and no agent is a node of $\mathbb{G}$.

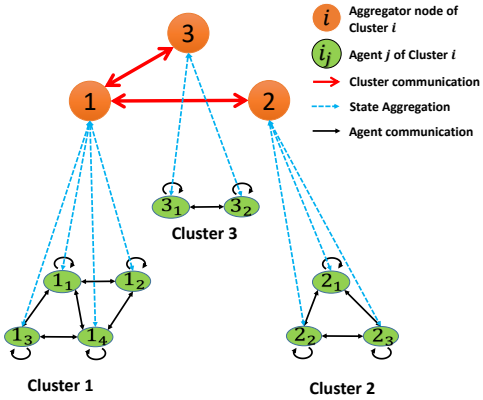

Figure 3.1. An Example of a Double-layered Multi-Agent Network

Consider an overall linear equation

$$A\boldsymbol{x} = b$$

where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$. Suppose $Ax = b$ has at least one solution. Suppose each agent i_j knows part of the overall linear equation (in accord with rules given in the next section), which might not be as much as a complete row or column of A. Let each agent i_j control a state vector $x_{ij}(t)$, while the cluster aggregators do not control any state and only play the role of providing communications. The **problem** of interest is to develop a distributed update for each agent such that all $x_{ij}(t)$ converge to constant vectors x^*_{ij}, $j = 1, 2, \cdots, c_i$ and $i = 1, 2, \cdots, c$, which jointly, via an arrangement set out in the next section, form a solution to $Ax = b$.

[2]Here we use the term bidirectional instead of undirected to emphasize the two-way nature of information flows.

3.3 Global-Consensus and Local-Conservation

Suppose each agent i_j in a cluster i knows $A_{ij} \in \mathbb{R}^{m_i \times n_{ij}}$ and $b_{ij} \in \mathbb{R}^{m_i}$ such that the collection of them

$$\begin{bmatrix} A_{i1} & A_{i2} & \cdots & A_{ic_i} \end{bmatrix} = A_i \in \mathbb{R}^{m_i \times n}, \quad \sum_{j=1}^{c_i} b_{ij} = b_i \in \mathbb{R}^{m_i} \tag{3.1}$$

are a block row of the overall equation, where

$$A = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_c \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_c \end{bmatrix}. \tag{3.2}$$

Here, b_i result from a partition of b. The b_{ij} could be any vectors that satisfy the equation (3.1). One simple choice of such b_{ij} is obtained by setting one b_{ij} equal to b_i and the others equal to zero. In addition, one has

$$\sum_{j=1}^{c_i} n_{ij} = n, \ i = 1, 2, \cdots, c, \quad \sum_{i=1}^{c} m_i = m. \tag{3.3}$$

Note that $n_{ij} = 1$ and $m_i = 1$ are permitted, but are of course not required. Consistent with the set-up of Fig. 3.1, an example of how each agent's locally available information A_{ij}, b_{ij} is related to the overall equation $Ax = b$ is shown in Fig. 3.2.

Suppose each agent i_j controls a state vector $x_{ij}(t) \in \mathbb{R}^{n_{ij}}$. In this section, we aim to devise a distributed update for each agent i_j's state $x_{ij}(t)$ to converge exponentially fast to a constant vector x_{ij}^* such that:

- All x_{ij}^*, $j = 1, 2, \cdots, c_i$ within each cluster i, $i = 1, 2, \cdots, c$, satisfy the following

$$\textbf{Local Conservation:} \quad \sum_{j=1}^{c_i} (A_{ij} x_{ij}^* - b_{ij}) = 0. \tag{3.4}$$

- All $\boldsymbol{x}_i^* = \text{col}\,\{x_{i1}^*, \cdots, x_{ic_i}^*\}$, $i = 1, 2, \cdots, c$, among all clusters in the network reach a consensus $\boldsymbol{x}^*$, that is,

$$\textbf{Global Consensus:} \quad \boldsymbol{x}_1^* = \boldsymbol{x}_2^* = \cdots = \boldsymbol{x}_c^* = \boldsymbol{x}^* \tag{3.5}$$

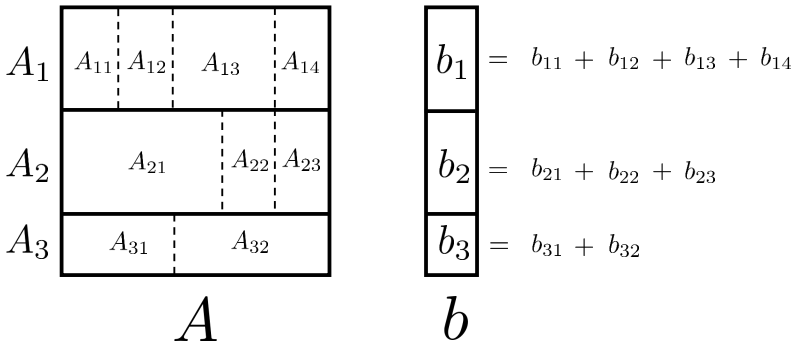

Figure 3.2. An example of the relation between agents' locally available information and the overall equation for the network of Fig. 3.1. The various sub-matrices and sub-vectors do not have to be scalar.

From (3.1) and (3.4) one has $A_i \boldsymbol{x}_i^* = b_i$. This and the global consensus in (3.5) imply $A\boldsymbol{x}^* = b$. All x_{ij}^* satisfying the local conservation (3.4) and the global consensus (3.5) are said to form a *consensus-conservation solution* $\boldsymbol{x}^*$ to $Ax = b$. Note that however large the matrix A may be, it is always possible to conceive of a network structure in which the state vector size and the degree of every agent in each subnetwork are bounded independently of the size of A. This is a crucial scalability property.

3.3.1 The Update

Let $\boldsymbol{x}_i(t) \in \mathbb{R}^n$ denote a column collection of agent states in cluster i, $i = 1, 2, \cdots, c$, that is,

$$\boldsymbol{x}_i(t) = \text{col}\,\{x_{i1}(t), \cdots, x_{ic_i}(t)\}. \tag{3.6}$$

Aggregator i, through communication with the agents i_j maintains a copy of $\boldsymbol{x}_i(t)$. This vector of course does depend on the size of A; it is inevitable that if one is solving $Ax = b$ some part of the network must be designed to hold the solution x. In our case, all aggregators will hold the solution in the limit as t goes to infinity. To achieve a consensus-conservation solution $\boldsymbol{x}^*$, it is sufficient to achieve $A_i \boldsymbol{x}_i(t) = b_i$ while all $\boldsymbol{x}_i(t)$, $i = 1, 2, \cdots, c$, reach a consensus. In order to allow the aggregator to decompose

the global consensus of $\boldsymbol{x}_i(t)$ into relations involving agents' states, we let $E_{ij} \in \mathbb{R}^{n_{ij} \times n}$ denote a matrix consisting of rows from I_n such that col $\{E_{ij}, j = 1, 2, \cdots, c_i\} = I_n$ and

$$A_{ij} = A_i E'_{ij}.$$

Then one has

$$x_{ij}(t) = E_{ij}\boldsymbol{x}_i(t).$$

It follows that all $\boldsymbol{x}_i(t)$ reaching a consensus is equivalent to requiring that $\forall i = 1, \cdots, c_i$ and $\forall k \in \mathcal{N}_i$,

$$x_{ij}(t) \to E_{ij}\boldsymbol{x}_k(t). \tag{3.7}$$

To achieve the local conservation (3.4), one also introduces an additional *coordination state* $z_{ij}(t) \in \mathbb{R}^{m_i}$ associated with and stored by each agent i_j. As noted earlier, m_i may even be 1, and in any case, the inclusion of $z_{ij}(t)$ as a quantity managed by agent i_j does not destroy the scalability property associated with the arrangement, either by virtue of its dimension or, as will be seen below, by virtue of its calculation. Then we propose the following update for each agent i_j, $i = 1, 2, \cdots, c$ and $j = 1, 2, \cdots, c_i$:

$$\begin{aligned} \dot{x}_{ij} =& - A'_{ij}\left(A_{ij}x_{ij} - b_{ij} - \sum_{i_k \in \mathcal{N}_{ij}} (z_{ij} - z_{ik})\right) \\ & - \sum_{k \in \mathcal{N}_i} (x_{ij} - E_{ij}\boldsymbol{x}_k) \end{aligned} \tag{3.8}$$

$$\dot{z}_{ij} = A_{ij}x_{ij} - b_{ij} - \sum_{i_k \in \mathcal{N}_{ij}} (z_{ij} - z_{ik}) \tag{3.9}$$

where the first line of update (3.8) and (3.9) aim to achieve the local conservation (3.4) while the second line of update (3.8) aims to achieve the global-consensus (3.5). One natural generalization to the proposed updates (3.8)-(3.9) is achievable by assigning different weights to controls for the local conservation and the global consensus, respectively, with the aim of achieving faster convergence. Achieving optimal choice of such weights might however require more than locally available information, which will not be discussed in this section.

Remark 3.3.1 *Note immediately that in implementation of* (3.8)-(3.9)*, information about x_{ij} is only shared between clusters while information about z_{ij} is only shared among agents within the same cluster. That is, each aggregator i is able to access $\boldsymbol{x}_k(t)$ for some k, depending on the graph $\mathbb{G}$, through aggregator communications. Of course, $\boldsymbol{x}_k(t)$ is collected by its aggregator neighbor $k \in \mathcal{N}_i$; aggregator i distributes $E_{ij}\boldsymbol{x}_k(t)$ to each agent i_j of cluster i. Note that $|\mathcal{N}_i|$ can be bounded independently of the size of A through the design of the aggregator network with graph $\mathbb{G}$. Within each cluster i, each agent i_j only needs to access its neighbors' coordination state $z_{ik}, i_k \in \mathcal{N}_{ij}$, through agent-agent communication in cluster i. Thus, the proposed updates* (3.8)-(3.9) *are obviously* ***distributed*** *in the sense that only communications among aggregator-neighbors and agent-neighbors are involved. Compared with existing consensus-based distributed linear equation solvers [12, 13, 16], distributed updates* (3.8)-(3.9)

- **individual agents** *in the distributed solvers require* ***much less knowledge*** *of the overall equation and control states of* ***much smaller dimension****. For a given $A \in \mathbb{R}^{m\times n}$ of fixed size, each agent i_j knows $A_{ij} \in \mathbb{R}^{m_i\times n_{ij}}$ and $b_{ij} \in \mathbb{R}^{m_i}$, and controls states $x_{ij}(t) \in \mathbb{R}^{n_{ij}}$, and $z_{ij}(t) \in \mathbb{R}^{m_i}$. Sizes of all these locally available matrices and state vectors might change with respect to the number of clusters and the number of agents in each cluster (but as already noted, can be bounded independently of the size of A). To see why this is so, we note from* (3.3) *and partitions in Fig. 3.2 that increasing c and c_i leads to the decreases of m_i and n_{ij}, respectively. Specially, when the number of clusters is m and the number of agents within each cluster is n, that is, $c = m$ and $c_i = n$, each agent only needs to know* ***two scalar entries*** *$A_{ij} \in \mathbb{R}$, $b_{ij} \in \mathbb{R}$ and updates* ***two scalar states****, namely $x_{ij}(t) \in \mathbb{R}$, $z_{ij}(t) \in \mathbb{R}$.*

- *allow all agents' state vectors to be of* ***different dimensions*** *while in contrast consensus-based distributed linear equation solvers require all agents to control*

states of the same size. Thus the proposed updates might be applied in networks of heterogeneous agents with different capabilities of storage.

3.3.2 Main result

Before proceeding, we first derive a compact form of (3.8)-(3.9). Towards this end, we let $\boldsymbol{z}_i(t) \in \mathbb{R}^{c_i m_i}$ denote the column collection of all agents' coordination states in cluster i, that is,

$$\boldsymbol{z}_i(t) = \text{col}\left\{z_{i1}(t), \cdots, z_{ic_i}(t)\right\}, \quad i = 1, 2, \cdots, c. \tag{3.10}$$

Let

$$\bar{A}_i = \text{diag}\left\{A_{i1}, \cdots, A_{ic_i}\right\}, \quad \bar{b}_i = \text{col}\left\{b_{i1}, \cdots, b_{ic_i}\right\} \tag{3.11}$$

and

$$\bar{L}_{\mathbb{G}_i} = L_{\mathbb{G}_i} \otimes I_{m_i}, \quad i = 1, 2, \cdots, c \tag{3.12}$$

with $L_{\mathbb{G}_i}$ the Laplacian matrix of the c_i-node connected and bidirectional graph $\mathbb{G}_i$. Recalling $\text{col}\left\{E_{ij}, j = 1, 2, \cdots, c_i\right\} = I_n$ and (3.6), one can write equations (3.8) and (3.9) as:

$$\dot{\boldsymbol{x}}_i = -\bar{A}_i'\left(\bar{A}_i\boldsymbol{x}_i - \bar{b}_i - \bar{L}_{\mathbb{G}_i}\boldsymbol{z}_i\right) - \sum_{k\in\mathcal{N}_i}\left(\boldsymbol{x}_i - \boldsymbol{x}_k\right) \tag{3.13}$$

$$\dot{\boldsymbol{z}}_i = \bar{A}_i\boldsymbol{x}_i - \bar{b}_i - \bar{L}_{\mathbb{G}_i}\boldsymbol{z}_i \tag{3.14}$$

for $i = 1, 2, \cdots, c$. Each equation pair in the above describes what is occurring at a particular cluster. Now let $\boldsymbol{x} = \text{col}\left\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_c\right\}$, $\boldsymbol{z} = \text{col}\left\{\boldsymbol{z}_1, \cdots, \boldsymbol{z}_c\right\}$,

$$\hat{A} = \text{diag}\left\{\bar{A}_1, \cdots, \bar{A}_c\right\}, \quad \hat{b} = \text{col}\left\{\bar{b}_1, \cdots, \bar{b}_c\right\}, \tag{3.15}$$

$$\hat{L} = \text{diag}\left\{\bar{L}_{\mathbb{G}_1}, \cdots, \bar{L}_{\mathbb{G}_c}\right\}, \quad \hat{L}_{\mathbb{G}} = L_{\mathbb{G}} \otimes I_n \tag{3.16}$$

with $L_{\mathbb{G}}$ Laplacian matrix of the c-node connected graph $\mathbb{G}$. Equations (3.13)-(3.14) can be further rewritten in the following compact form:

$$\dot{\boldsymbol{x}} = -\hat{A}'\left(\hat{A}\boldsymbol{x} - \hat{b} - \hat{L}\boldsymbol{z}\right) - \hat{L}_{\mathbb{G}}\boldsymbol{x} \tag{3.17}$$

$$\dot{\boldsymbol{z}} = \hat{A}\boldsymbol{x} - \hat{b} - \hat{L}\boldsymbol{z} \tag{3.18}$$

Or equivalently,

$$\begin{bmatrix} \dot{\boldsymbol{x}} \\ \dot{\boldsymbol{z}} \end{bmatrix} = Q \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{z} \end{bmatrix} + \begin{bmatrix} \hat{A}'\hat{b} \\ -\hat{b} \end{bmatrix} \tag{3.19}$$

with

$$Q = \begin{bmatrix} -\hat{A}'\hat{A} - \hat{L}_{\mathbb{G}} & \hat{A}'\hat{L} \\ \\ \hat{A} & -\hat{L} \end{bmatrix}. \tag{3.20}$$

To analyze the convergence of (3.19) we need the following lemma to characterize eigenvalues of Q.

Lemma 3.3.1 *Let*

$$M = \begin{bmatrix} -M_1'M_1 - M_2 & M_1'M_3 \\ \\ M_1 & -M_3 \end{bmatrix}$$

where the M_i are real, $i = 1, 2, 3$, and M_2 and M_3 are positive semi-definite. Then all eigenvalues of M are real negative or 0. Moreover, if 0 is an eigenvalue of M, it must be non-defective[3]*.*

The proof of Lemma 3.3.1 will be given in the Appendix. By this lemma and by establishing the convergence of the linear time-invariant system (3.19) to a constant steady state, one has the following main result.

Theorem 3.3.1 *Suppose $Ax = b$ has at least one solution, and the graphs $\mathbb{G}_i$, $i = 1, 2, \cdots, c$, and $\mathbb{G}$ are connected and bidirectional. Then under the distributed updates (3.8)-(3.9), all $x_{ij}(t)$ with $i = 1, 2, \cdots, c$ and $j = 1, 2, \cdots, c_i$ converge exponentially fast to constant vectors x_{ij}^* satisfying (3.4)-(3.5), which jointly form a consensus-conservation solution $\boldsymbol{x}^*$ to $Ax = b$.*

[3]An eigenvalue is non-defective if any only if its algebraic multiplicity equals its geometric multiplicity. In other words, the Jordan block corresponding to a non-defective eigenvalue is diagonal.

Proof of Theorem 3.3.1: We first prove that there exists a constant vector col $\{\hat{\boldsymbol{x}}, \hat{\boldsymbol{z}}\}$ which is an equilibrium of (3.19). Recall there exists a constant vector $y \in \mathbb{R}^n$ such that $Ay = b$. From the definitions of A_{ij}, b_{ij} in (3.1)-(3.2) and E_{ij}, one has

$$\sum_{j=1}^{c_i}(A_{ij}E_{ij}y - b_{ij}) = 0, \quad i = 1, 2, \cdots, c,$$

This equation and definitions of $\bar{A}_i, \bar{b}_i$ in (3.11) lead to

$$\left(\mathbf{1}'_{c_i} \otimes I_{m_i}\right)\left(\bar{A}_i y - \bar{b}_i\right) = 0, \quad i = 1, 2, \cdots, c. \tag{3.21}$$

Note that $\bar{L}_{\mathbb{G}_i} = L_{\mathbb{G}_i} \otimes I_{m_i}$, where $L_{\mathbb{G}_i}$ is the Laplacian matrix of a c_i-node connected and bidirectional graph $\mathbb{G}_i$. Then

$$\text{image } \bar{L}_{\mathbb{G}_i} = \ker\left(\mathbf{1}'_{c_i} \otimes I_{m_i}\right), \quad i = 1, 2, \cdots, c \tag{3.22}$$

From (3.21) and (3.22), one has

$$\left(\bar{A}_i y - \bar{b}_i\right) \in \text{image } \bar{L}_{\mathbb{G}_i}, \quad i = 1, 2, \cdots, c. \tag{3.23}$$

Then there exists a constant vector $\hat{z}_i \in \mathbb{R}^{c_i m_i}$ such that

$$\bar{A}_i y - \bar{b}_i - \bar{L}_{\mathbb{G}_i}\hat{z}_i = 0, \quad i = 1, 2, \cdots, c. \tag{3.24}$$

Let $\hat{\boldsymbol{x}} = \mathbf{1}_c \otimes y$. Note that $\hat{L}_{\mathbb{G}} = L_{\mathbb{G}} \otimes I_n$ with $L_{\mathbb{G}}$ the Laplacian matrix of the c-node connected and bidirectional graph $\mathbb{G}$. Then

$$\hat{L}_{\mathbb{G}}\hat{\boldsymbol{x}} = 0 \tag{3.25}$$

Let $\hat{\boldsymbol{z}} = \{\hat{z}_1, \hat{z}_2, \cdots, \hat{z}_c\}$. From (3.24) and definitions of $\hat{A}, \hat{b}$ in (3.15), one has

$$\hat{A}\hat{\boldsymbol{x}} - \hat{b} - \hat{L}\hat{\boldsymbol{z}} = 0, \tag{3.26}$$

This equation and (3.25) imply that col $\{\hat{\boldsymbol{x}}, \hat{\boldsymbol{z}}\}$ is an equilibrium of (3.19).

Second, we analyze the convergence of the error

$$\boldsymbol{e}(t) = \begin{bmatrix} \boldsymbol{x}(t) \\ \boldsymbol{z}(t) \end{bmatrix} - \begin{bmatrix} \hat{\boldsymbol{x}} \\ \hat{\boldsymbol{z}} \end{bmatrix}. \tag{3.27}$$

From (3.19) and the fact that col $\{\hat{\boldsymbol{x}}, \hat{\boldsymbol{z}}\}$ is an equilibrium of (3.19), one has

$$\dot{\boldsymbol{e}} = Q\boldsymbol{e} \tag{3.28}$$

From Lemma 3.3.1, the structure of Q in (3.20) and the fact that the Laplacian matrices $\hat{L}$ and $\hat{L}_{\mathbb{G}}$ are symmetric and positive semi-definite, one concludes that all eigenvalues of Q are real negative or 0. Moreover, if 0 is an eigenvalue of Q, it must be non-defective. Thus there exists a constant vector $q \in \ker Q$ such that $\boldsymbol{e}(t)$ of the linear time-invariant system (3.28) converges to q exponentially fast [100]. Thus col $\{\boldsymbol{x}(t), \boldsymbol{z}(t)\}$ converges exponentially fast to a constant vector col $\{\hat{\boldsymbol{x}}^*, \hat{\boldsymbol{z}}^*\}$, where

$$\begin{bmatrix} \hat{\boldsymbol{x}}^* \\ \hat{\boldsymbol{z}}^* \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{x}} \\ \hat{\boldsymbol{z}} \end{bmatrix} + q, \quad q \in \ker Q. \tag{3.29}$$

Partition the constant vector $\hat{\boldsymbol{x}}^*$ such that

$$\hat{\boldsymbol{x}}^* = \text{col}\,\{\boldsymbol{x}_1^*, \cdots, \boldsymbol{x}_c^*\} \tag{3.30}$$

where $\boldsymbol{x}_i^* \in \mathbb{R}^n$ is further partitioned as

$$\boldsymbol{x}_i^* = \text{col}\,\{x_{i1}^*, x_{i2}^*, \cdots, x_{ic_i}^*\} \tag{3.31}$$

with $x_{ij}^* \in \mathbb{R}^{n_{ij}}$. Evidently, we have that $x_{ij}(t)$ converges to x_{ij}^* exponentially fast. In the following, one only needs to show that all these x_{ij}^* satisfy the local conservation in (3.4) and the global consensus in (3.5).

From (3.29) and the property that col $\{\hat{\boldsymbol{x}}, \hat{\boldsymbol{z}}\}$ is an equilibrium of (3.19), one concludes that col $\{\hat{\boldsymbol{x}}^*, \hat{\boldsymbol{z}}^*\}$ is also an equilibrium of (3.19). It follows that

$$0 = -\hat{A}'\left(\hat{A}\hat{\boldsymbol{x}}^* - \hat{b} - \hat{L}\hat{\boldsymbol{z}}^*\right) - \hat{L}_{\mathbb{G}}\hat{\boldsymbol{x}}^* \tag{3.32}$$

$$0 = \hat{A}\hat{\boldsymbol{x}}^* - \hat{b} - \hat{L}\hat{\boldsymbol{z}}^* \tag{3.33}$$

Partition $\hat{\boldsymbol{z}}^* = \text{col}\,\{\boldsymbol{z}_1^*, \boldsymbol{z}_2^*, \cdots, \boldsymbol{z}_c^*\}$ with $\boldsymbol{z}_i^* \in \mathbb{R}^{c_i m_i}$. From (3.33) and the definitions of $\hat{A}, \hat{b}, \hat{L}$ in (3.15)-(3.16), one has

$$\bar{A}_i\boldsymbol{x}_i^* - \bar{b}_i - \bar{L}_{\mathbb{G}_i}\boldsymbol{z}_i^* = 0, \quad i = 1, 2, \cdots, c. \tag{3.34}$$

From the definitions of $\bar{A}_i, \bar{b}_i, \bar{L}_{\mathbb{G}_i}$ in (3.11)-(3.12), one can rewrite (3.34) as

$$\begin{bmatrix} A_{i1}x_{i1}^* \\ A_{i2}x_{i2}^* \\ \vdots \\ A_{ic_i}x_{ic_i}^* \end{bmatrix} - \begin{bmatrix} b_{i1} \\ b_{i2} \\ \vdots \\ b_{ic_i} \end{bmatrix} - (L_{\mathbb{G}_i} \otimes I_{m_i})\boldsymbol{z}_i^* = 0. \tag{3.35}$$

Premultiplying by $\mathbf{1}'_{c_i} \otimes I_{m_i}$ on both sides of (3.35), one has

$$\sum_{j=1}^{c_i}(A_{ij}x_{ij}^* - b_{ij}) - [(\mathbf{1}'_{c_i}L_{\mathbb{G}_i}) \otimes I_{m_i}]\boldsymbol{z}_i^* = 0.$$

Since $L_{\mathbb{G}_i}$ is the Laplacian of c_i-node connected bidirectional graph $\mathbb{G}_i$, one has $\mathbf{1}'_{c_i}L_{\mathbb{G}_i} = 0$. Thus

$$\sum_{j=1}^{c_i}(A_{ij}x_{ij}^* - b_{ij}) = 0. \tag{3.36}$$

In addition, from (3.32)-(3.33), one has

$$\hat{L}_{\mathbb{G}}\hat{\boldsymbol{x}}^* = 0$$

where $\hat{L}_{\mathbb{G}} = L_{\mathbb{G}} \otimes I_m$ with $L_{\mathbb{G}}$ the Laplacian matrix of the c-node connected and bidirectional graph $\mathbb{G}$. Thus there exists a constant vector $\boldsymbol{x}^*$ such that

$$\hat{\boldsymbol{x}}^* = \mathbf{1}_c \otimes \boldsymbol{x}^* \tag{3.37}$$

Together with (3.30), this implies

$$\boldsymbol{x}_1^* = \boldsymbol{x}_2^* = \cdots = \boldsymbol{x}_c^* = \boldsymbol{x}^* \tag{3.38}$$

with $\boldsymbol{x}_i^*$ a collection of x_{ij}^* as defined in (3.31). From (3.36) and (3.38) one concludes that all x_{ij}^* satisfy the local conservation in (3.4) and the global consensus in (3.5).

Therefore, the $x_{ij}(t)$ converge exponentially fast to constant vectors x_{ij}^* which form a consensus-conservation solution $\boldsymbol{x}^*$ to $Ax = b$. This completes the proof. ■

3.3.3 Validation

We utilize the double-layer network as in Fig. 3.1 to solve the following linear equation $A\boldsymbol{x} = b$, which is partitioned according to the structure as in Fig. 3.2 with details as follows: Suppose each agent i_j knows A_{ij} and b_{ij}, and we employ the

$$A=\left[\begin{array}{c|cc|cc} 1 & 2 & 1 & 1 & 2 \\ 2 & -1 & 3 & -3 & 0 \\ \hline -6 & 2 & 6 & 1 & -2 \\ 2 & -6 & 3 & 7 & 0 \\ \hline 4 & -4 & 4 & 8 & 2 \end{array}\right], b=\left[\begin{array}{c} 8 \\ 8 \\ \hline 11 \\ -17 \\ \hline -9 \end{array}\right]=\begin{array}{l} \begin{bmatrix}4\\2\end{bmatrix}+\begin{bmatrix}3\\1\end{bmatrix}+\begin{bmatrix}0\\0\end{bmatrix}+\begin{bmatrix}1\\5\end{bmatrix} \\ =\begin{bmatrix}5\\-8\end{bmatrix}+\begin{bmatrix}0\\0\end{bmatrix}+\begin{bmatrix}6\\-9\end{bmatrix} \\ =\begin{bmatrix}-5\end{bmatrix}+\begin{bmatrix}-4\end{bmatrix} \end{array}$$

updates (3.8) and (3.9) with arbitrary initializations. Let

$$V(t)=\frac{1}{2}\sum_{i=1}^{c}\left\|\begin{bmatrix}x_{i1}(t)\\ \vdots \\ x_{ic_i}(t)\end{bmatrix}-\boldsymbol{x}^*\right\|_2^2$$

where $\boldsymbol{x}^* = \begin{bmatrix}0.77 & 2.79 & 1.98 & -1.10 & 0.38\end{bmatrix}'$ is a solution to $Ax = b$. Thus $V(t)$ measures the closeness of all agent states to forming a consensus-conservation solution. We then utilize the double-layer network as in Fig. 3.3, which results from adding one additional cluster to Fig. 3.1, to solve the same linear equation with partitions as follows (for clarity, the details of agent communications are depicted for only one cluster.):

Simulations in Fig. 3.4 suggests that $V(t)$ converges exponentially fast to 0 and thus all $x_{ij}(t)$ converge exponentially fast to constant vectors that form a consensus-conservation solution $\boldsymbol{x}^*$ in both cases. This is in accord with Theorem 3.3.1. Moreover, the convergence rate can also be affected by the number of partitions and the network topology. Intuitively, more partitions of the overall linear equations require more clusters for implementing the proposed updates, which leads to slower convergence as suggested by Fig. 3.4.

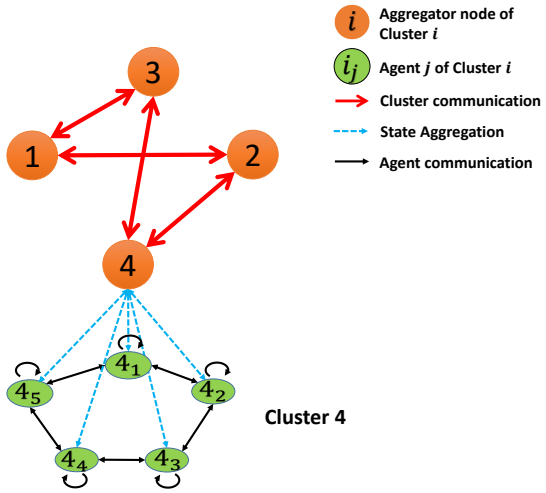

Figure 3.3. Another Double-layered Multi-Agent Network. For clarity, the details of agent communications are depicted for only one cluster.

$$A=\left[\begin{array}{c|cc|cc} 1 & 2 & 1 & 1 & 2 \\ 2 & -1 & 3 & -3 & 0 \\ \hline -6 & 2 & 6 & 1 & -2 \\ \hline 2 & -6 & 3 & 7 & 0 \\ \hline -3 & -4 & 0 & -2 & 6 \end{array}\right], b=\left[\begin{array}{c} 8 \\ 8 \\ \hline 11 \\ \hline -17 \\ \hline -9 \end{array}\right] \begin{array}{l} =\begin{bmatrix}4\\2\end{bmatrix}+\begin{bmatrix}3\\1\end{bmatrix}+\begin{bmatrix}0\\0\end{bmatrix}+\begin{bmatrix}1\\5\end{bmatrix} \\ =[5]+[0]+[6] \\ =[-9]+[-8] \\ =[-4]+[-5]+[0]+[0]+[0] \end{array}$$

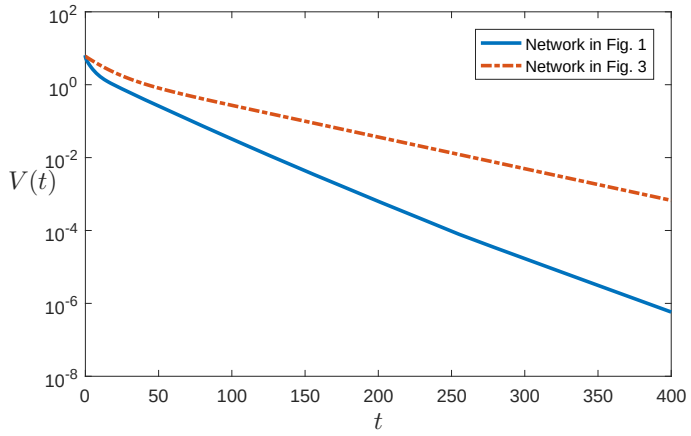

Figure 3.4. Evolution of $V(t)$ under the proposed updates (3.8)-(3.9)

3.4 Global-Conservation and Local-Consensus

In the previous section, agents in the same cluster collectively know a block row of the overall matrix A as indicated in Fig. 3.2. In this section, we consider a different

situation in which agents in the same cluster collectively know a block column of A, for which a different coordination arrangement will be required in the cluster-layer and the agent-layer, as will be shown later.

Suppose each agent i_j in cluster i knows $A_{ij} \in \mathbb{R}^{m_{ij} \times n_i}, b_{ij} \in \mathbb{R}^{m_{ij}}$ such that the collection of them

$$\begin{bmatrix} A_{i1} \\ A_{i2} \\ \vdots \\ A_{ic_i} \end{bmatrix} = A_i \in \mathbb{R}^{m \times n_i} \qquad \begin{bmatrix} b_{i1} \\ b_{i2} \\ \vdots \\ b_{ic_i} \end{bmatrix} = b_i \in \mathbb{R}^{m} \tag{3.39}$$

are parts of the overall linear equation $Ax = b$, where

$$A = \begin{bmatrix} A_1 & A_2 & \cdots & A_c \end{bmatrix}, \quad b = \sum_{i=1}^{c} b_i. \tag{3.40}$$

Note that while b is given, the b_i's are not; they can be any vectors that satisfy (3.40) (a particular case is that one b_i equals b and the others are zero). Besides, one has

$$\sum_{j=1}^{c_i} m_{ij} = m, \ i = 1, 2, \cdots, c, \quad \sum_{i=1}^{c} n_i = n. \tag{3.41}$$

An example of the relation between agents' locally available information A_{ij}, b_{ij} and the overall equation $Ax = b$ is shown in Fig. 3.5. Note that the symbols A_{ij} and b_i here are used differently from their use in previous sections but the notation is convenient.

Suppose each agent i_j controls a state vector $x_{ij}(t) \in \mathbb{R}^{n_i}$. In this section, we aim to devise a distributed update for each agent i_j's state $x_{ij}(t)$ to converge exponentially fast to a constant vector x^*_{ij}, $i = 1, 2, \cdots, c$ and $j = 1, 2, \cdots, c_i$, such that

- All x^*_{ij}, $j = 1, 2, \cdots, c_i$, within each cluster i reach a consensus x^*_i, that is,

$$\textbf{Local Consensus:} \quad x^*_{i1} = x^*_{i2} = \cdots = x^*_{ic_i} = x^*_i \tag{3.42}$$

- All x^*_i, $i = 1, 2, \cdots, c$ among all clusters in the network satisfy the following

$$\textbf{Global Conservation:} \quad \sum_{i=1}^{c} (A_i x^*_i - b_i) = 0 \tag{3.43}$$

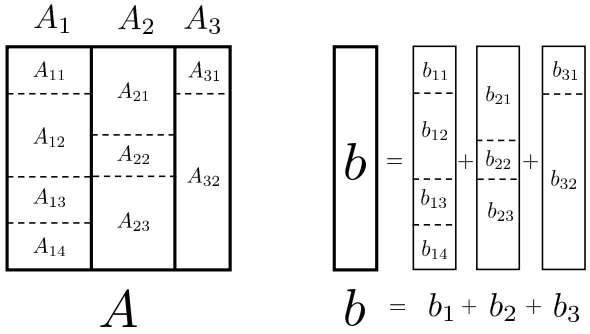

Figure 3.5. An example of the relation between agents' locally available information and the overall equation for the network of Fig. 3.1.

Let $\boldsymbol{x}^* = \text{col}\,\{x_1^*, x_2^*, \cdots, x_c^*\}$ be the column collection of the consensus value to which all agents in the same cluster converge. From (3.40) and (3.43) one has $A\boldsymbol{x}^* = b$. Thus the x_{ij}^* satisfying the local consensus (3.42) and the global conservation (3.43) are said to form a *conservation-consensus solution* $\boldsymbol{x}^*$ to $Ax = b$.

Note that here all x_{ij}^* in the same cluster are the same, which is part of the solution to the overall equation, while, in contrast, in the consensus-conservation solution defined in the previous section, the x_{ij}^* in each cluster i jointly form a solution to the overall equation $Ax = b$.

3.4.1 The Update

In order to achieve the global conservation, we employ the networks $\mathbb{G}$ and $\mathbb{G}_i$ described in the section of Problem Formulation and introduce an additional state $z_{ij}(t) \in \mathbb{R}^{m_{ij}}$ at each agent i_j. Let $E_{ij} \in \mathbb{R}^{m_{ij} \times m}$ consist of rows of the identity matrix I_m such that $\text{col}\,\{E_{ij}, j = 1, 2, \cdots, c_i\} = I_m$ and

$$A_{ij} = E_{ij} A_i.$$

Let $\boldsymbol{z}_i(t) \in \mathbb{R}^m$ denote the column of all coordination states in cluster i, $i = 1, 2, \cdots, c_i$, that is,

$$\boldsymbol{z}_i = \text{col}\,\{z_{i1}, z_{i2}, \cdots, z_{ic_i}\}.$$

Then

$$z_{ij} = E_{ij}\boldsymbol{z}_i.$$

Suppose each cluster aggregator i is able to access its neighbor aggregator's coordination state $\boldsymbol{z}_k(t)$, $k \in \mathcal{N}_i$ through aggregator communication and then distributes $E_{ij}\boldsymbol{z}_k(t)$ to each agent i_j of cluster i. Within each cluster i, each agent i_j is able to access to its neighbors' state $x_{ik}, i_k \in \mathcal{N}_{ij}$, through agent-agent communication in cluster i. Then one proposes the following update for each agent i_j, $i = 1, 2, \cdots, c$ and $j = 1, 2, \cdots, c_i$:

$$\begin{aligned}\dot{x}_{ij} = &- A'_{ij}\left(A_{ij}x_{ij} - b_{ij} - \sum_{k\in\mathcal{N}_i}(z_{ij} - E_{ij}\boldsymbol{z}_k)\right)\\ &- \sum_{i_k\in\mathcal{N}_{ij}}(x_{ij} - x_{ik}) \qquad (3.44)\\ \dot{z}_{ij} = &\ A_{ij}x_{ij} - b_{ij} - \sum_{k\in\mathcal{N}_i}(z_{ij} - E_{ij}\boldsymbol{z}_k) \qquad (3.45)\end{aligned}$$

where the first line of update (3.44) and (3.45) aim to achieve global conservation in (3.43) while the second line of (3.44) aims to achieve the local consensus in (3.42).

Remark 3.4.1 *Note immediately that in implementation of* (3.44)-(3.45)*, information about z_{ij} is only shared between clusters while information about x_{ij} is only shared among agents within the same cluster. Thus, each aggregator i is able to access $\boldsymbol{z}_k(t)$ through aggregator communications when $\boldsymbol{z}_k(t)$ is collected by its aggregator neighbor $k \in \mathcal{N}_i$; aggregator i distributes $E_{ij}\boldsymbol{z}_k(t)$ to each agent i_j of cluster i. Within each cluster i, each agent i_j only needs to access its neighbors' coordination state $x_{ik}, i_k \in \mathcal{N}_{ij}$, through agent-agent communication in cluster i. And thus the proposed updates* (3.44)-(3.45) *are obviously* ***distributed*** *in the sense that only communications among aggregator-neighbors and agent-neighbors are involved. Moreover, the agents i_j have storage and communication requirements which are fully scalable with the size of A. As before, the aggregator nodes will store vectors of dimension determined by the dimension of A. However, as already noted, the number of neighbors of an aggregator node does not need to grow with the dimension of A. Compared with*

existing consensus-based distributed linear equation solvers [12,13,16], distributed updates (3.8)-(3.9)

- **individual agents** *in the distributed solvers require* ***much less knowledge*** *of the overall equation and control states of* ***much smaller dimension****. For a given overall linear equation with* $A \in \mathbb{R}^{m\times n}$*, each agent* i_j *knows* $A_{ij} \in \mathbb{R}^{m_{ij}\times n_i}$ *and* $b_{ij} \in \mathbb{R}^{m_{ij}}$*, and controls states* $x_{ij}(t) \in \mathbb{R}^{n_i}$*, and* $z_{ij}(t) \in \mathbb{R}^{m_{ij}}$*. Sizes of these locally available matrices and state vectors could change with respect to the number of clusters and the number of agents in each cluster. From* (3.41) *and partitions in Fig. 3.5, one has that increasing* c_i *and* c *leads to the decreases of* m_{ij} *and* n_i*, respectively. Specially, when the number of clusters is* n *and the number of agents within each cluster is* m*, that is,* $c = n$ *and* $c_i = m$*, each agent only needs to know* ***two scalar entries*** $A_{ij} \in \mathbb{R}$*,* $b_{ij} \in \mathbb{R}$ *and updates* ***two scalar states****, namely* $x_{ij}(t) \in \mathbb{R},\ z_{ij}(t) \in \mathbb{R}$*.*

- *allow all agents' state vectors to be of* ***different dimensions****, which is the same as the distributed updates* (3.8)-(3.9) *in the previous section.*

3.4.2 Main result

Before proceeding, we first derive a compact form of (3.44)-(3.45). Towards this end, we let $\boldsymbol{x}_i \in \mathbb{R}^{c_i n_i}$ denote the column collection of all agents' states in cluster i, $i = 1, 2, \cdots, c$, that is,

$$\boldsymbol{x}_i = \text{col}\,\{x_{i1}, x_{i2}, \cdots, x_{ic_i}\}.$$

Let

$$\bar{A}_i = \text{diag}\,\{A_{i1}, \cdots, A_{ic_i}\}, \quad \bar{L}_{\mathbb{G}_i} = L_{\mathbb{G}_i} \otimes I_{n_i} \tag{3.46}$$

with $L_{\mathbb{G}_i}$ the Laplacian matrix of the c_i-node connected graph $\mathbb{G}_i$. From equations (3.44)-(3.45) and col $\{E_{ij}, j = 1, 2, \cdots, c_i\} = I_m$, one has

$$\dot{\boldsymbol{x}}_i = -\bar{A}_i' \left(\bar{A}_i \boldsymbol{x}_i - b_i - \sum_{k \in \mathcal{N}_i} (\boldsymbol{z}_i - \boldsymbol{z}_k) \right) - \bar{L}_{\mathbb{G}_i} \boldsymbol{x}_i \tag{3.47}$$

$$\dot{\boldsymbol{z}}_i = \bar{A}_i \boldsymbol{x}_i - b_i - \sum_{k \in \mathcal{N}_i} (\boldsymbol{z}_i - \boldsymbol{z}_k) \tag{3.48}$$

for $i = 1, 2, \cdots, c$. Let $\boldsymbol{x} = \text{col}\,\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_c\}$ and $\boldsymbol{z} = \text{col}\,\{\boldsymbol{z}_1, \cdots, \boldsymbol{z}_c\}$,

$$\hat{A} = \text{diag}\,\{\bar{A}_1, \cdots, \bar{A}_c\}, \quad \hat{b} = \text{col}\,\{b_1, \cdots, b_c\}, \tag{3.49}$$

$$\hat{L} = \text{diag}\,\{\bar{L}_{\mathbb{G}_1}, \cdots, \bar{L}_{\mathbb{G}_c}\}, \quad \hat{L}_{\mathbb{G}} = L_{\mathbb{G}} \otimes I_m \tag{3.50}$$

with $L_{\mathbb{G}}$ the Laplacian matrix of the c-node connected graph $\mathbb{G}$. Equations (3.47)-(3.48) can be written in the following compact form:

$$\dot{\boldsymbol{x}} = -\hat{A}' \left(\hat{A}\boldsymbol{x} - \hat{b} - \hat{L}_{\mathbb{G}} \boldsymbol{z} \right) - \hat{L}\boldsymbol{x} \tag{3.51}$$

$$\dot{\boldsymbol{z}} = \hat{A}\boldsymbol{x} - \hat{b} - \hat{L}_{\mathbb{G}} \boldsymbol{z} \tag{3.52}$$

which is

$$\begin{bmatrix} \dot{\boldsymbol{x}} \\ \dot{\boldsymbol{z}} \end{bmatrix} = Q \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{z} \end{bmatrix} + \begin{bmatrix} \hat{A}'\hat{b} \\ -\hat{b} \end{bmatrix} \tag{3.53}$$

with

$$Q = \begin{bmatrix} -\hat{A}'\hat{A} - \hat{L} & \hat{A}'\hat{L}_{\mathbb{G}} \\ \hat{A} & -\hat{L}_{\mathbb{G}} \end{bmatrix}. \tag{3.54}$$

Theorem 3.4.1 *Suppose $Ax = b$ has at least one solution, and the graphs $\mathbb{G}_i$, $i = 1, 2, \cdots, c$ and $\mathbb{G}$ are connected and bidirectional. Then under the distributed updates* (3.44) *and* (3.45)*, all $x_{ij}(t)$ with $i = 1, 2, \cdots, c$ and $j = 1, 2, \cdots, c_i$ converge exponentially fast to constant vectors x_{ij}^* which satisfy the local consensus (3.42) and the global conservation (3.43) and thus form a conservation-consensus solution $\boldsymbol{x}^*$ to $Ax = b$.*

Proof of Theorem 3.4.1: We first prove that there exists a constant vector col $\{\hat{\boldsymbol{x}}, \hat{\boldsymbol{z}}\}$ which is an equilibrium of (3.53). Since there exists a constant vector $y \in \mathbb{R}^n$ such that $Ay = b$, using the definition of A_i, b_i in (3.40), one has

$$\begin{bmatrix} A_1 & A_2 & \cdots & A_c \end{bmatrix} y = \sum_{i=1}^{c} b_i$$

Partition $y = \text{col}\,\{y_1, \cdots, y_c\}$ with $y_i \in \mathbb{R}^{n_i}$. Then one has

$$\sum_{i=1}^{c} (A_i y_i - b_i) = 0$$

It follows from this and (3.39) that

$$\sum_{i=1}^{c} \left(\begin{bmatrix} A_{i1} \\ \vdots \\ A_{ic_i} \end{bmatrix} y_i - \begin{bmatrix} b_{i1} \\ \vdots \\ b_{ic_i} \end{bmatrix} \right) = 0$$

This and the definitions of $\bar{A}_i$ in (3.46) imply

$$\sum_{i=1}^{c} (\bar{A}_i \boldsymbol{y}_i - b_i) = 0 \tag{3.55}$$

with $\boldsymbol{y}_i = \mathbf{1}_{c_i} \otimes y_i$. Let $\hat{\boldsymbol{x}} = \text{col}\,\{\boldsymbol{y}_1, \cdots, \boldsymbol{y}_c\}$. From (3.55) and the definitions of $\hat{A}, \hat{b}$ in (3.49), one has

$$(\mathbf{1}_c' \otimes I_m) \left(\hat{A}\hat{\boldsymbol{x}} - \hat{b} \right) = 0 \tag{3.56}$$

Recall that $\hat{L}_{\mathbb{G}} = L_{\mathbb{G}} \otimes I_m$ with $L_{\mathbb{G}}$ the Lapalacian matrix of a c-node connected, bidirectional graph $\mathbb{G}$. Then

$$\text{image}\ \hat{L}_{\mathbb{G}} = \ker (\mathbf{1}_c' \otimes I_m) \tag{3.57}$$

which with (3.56) implies

$$\left(\hat{A}\hat{\boldsymbol{x}} - \hat{b} \right) \in \text{image}\ \hat{L}_{\mathbb{G}}. \tag{3.58}$$

Then there exists a constant vector $\hat{\boldsymbol{z}}$ such that

$$\hat{A}\hat{\boldsymbol{x}} - \hat{b} - \hat{L}_{\mathbb{G}}\hat{\boldsymbol{z}} = 0 \tag{3.59}$$

In addition, since $\bar{L}_{\mathbb{G}_i} = L_{\mathbb{G}_i} \otimes I_{n_i}$ with $L_{\mathbb{G}_i} \otimes I_{n_i}$ the Laplacian matrix of the c_i-node connected graph $\mathbb{G}_i$, and since $\boldsymbol{y}_i = \mathbf{1}_{c_i} \otimes y_i$, one has

$$\bar{L}_{\mathbb{G}_i} \boldsymbol{y}_i = 0, \quad i = 1, 2, \cdots, c.$$

Then because $\hat{L} = \text{diag}\,\{\bar{L}_{\mathbb{G}_1}, \cdots, \bar{L}_{\mathbb{G}_c}\}$ and $\hat{\boldsymbol{x}} = \text{col}\,\{\boldsymbol{y}_1, \cdots, \boldsymbol{y}_c\}$, one has

$$\hat{L}\hat{\boldsymbol{x}} = 0 \tag{3.60}$$

together with (3.59), this implies that col $\{\hat{\boldsymbol{x}}, \hat{\boldsymbol{z}}\}$ is an equilibrium of (3.53).

Second, we analyze the convergence of the error

$$\boldsymbol{e}(t) = \begin{bmatrix} \boldsymbol{x}(t) \\ \boldsymbol{z}(t) \end{bmatrix} - \begin{bmatrix} \hat{\boldsymbol{x}} \\ \hat{\boldsymbol{z}} \end{bmatrix}. \tag{3.61}$$

From (3.53) and the fact that col $\{\hat{\boldsymbol{x}}, \hat{\boldsymbol{z}}\}$ is an equilibrium of (3.53), one has

$$\dot{\boldsymbol{e}} = Q\boldsymbol{e} \tag{3.62}$$

From Lemma 3.3.1, the structure of Q in (3.54) and the fact that Laplacian matrices $\hat{L}$ and $\hat{L}_{\mathbb{G}}$ are symmetric and positive semi-definite, one has all eigenvalues of Q are real negative or 0. Moreover, if 0 is an eigenvalue of Q, it must be non-defective. Thus there exists a constant vector $q \in \ker Q$ such that $\boldsymbol{e}(t)$ of the linear time-invariant error system (3.62) converges to q exponentially fast [100]. Thus col $\{\boldsymbol{x}(t), \boldsymbol{z}(t)\}$ converges exponentially fast to a constant vector col $\{\hat{\boldsymbol{x}}^*, \hat{\boldsymbol{z}}^*\}$, where

$$\begin{bmatrix} \hat{\boldsymbol{x}}^* \\ \hat{\boldsymbol{z}}^* \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{x}} \\ \hat{\boldsymbol{z}} \end{bmatrix} + q, \quad q \in \ker Q. \tag{3.63}$$

Partition the constant vector $\hat{\boldsymbol{x}}^*$ such that

$$\hat{\boldsymbol{x}}^* = \text{col}\,\{\bar{\boldsymbol{x}}_1^*, \cdots, \bar{\boldsymbol{x}}_c^*\} \tag{3.64}$$

where $\bar{\boldsymbol{x}}_i^* \in \mathbb{R}^{c_i n_i}$ is further partitioned as

$$\bar{\boldsymbol{x}}_i^* = \text{col}\,\{x_{i1}^*, x_{i2}^*, \cdots, x_{ic_i}^*\} \tag{3.65}$$

with $x_{ij}^* \in \mathbb{R}^{n_i}$. Evidently, $x_{ij}(t)$ converges to x_{ij}^* exponentially fast. In the following, one only needs to show that all these x_{ij}^* satisfy the local consensus (3.42) and the global conservation (3.43).

From (3.63) and the property that col $\{\hat{\boldsymbol{x}}, \hat{\boldsymbol{z}}\}$ is an equilibrium of (3.53), one has col $\{\hat{\boldsymbol{x}}^*, \hat{\boldsymbol{z}}^*\}$ is an equilibrium of (3.53). Then

$$0 = -\hat{A}' \left(\hat{A}\hat{\boldsymbol{x}}^* - \hat{b} - \hat{L}_{\mathbb{G}}\hat{\boldsymbol{z}}^* \right) - \hat{L}\hat{\boldsymbol{x}}^* \tag{3.66}$$

$$0 = \hat{A}\hat{\boldsymbol{x}}^* - \hat{b} - \hat{L}_{\mathbb{G}}\hat{\boldsymbol{z}}^* \tag{3.67}$$

It follows that

$$\hat{L}\hat{\boldsymbol{x}}^* = 0 \tag{3.68}$$

From this, (3.64) and the definition of $\hat{L}$, one has

$$\bar{L}_{\mathbb{G}_i}\bar{\boldsymbol{x}}_i^* = 0, \quad i = 1, 2, \cdots, c. \tag{3.69}$$

Note that $\bar{L}_{\mathbb{G}_i} = L_{\mathbb{G}_i} \otimes I_{n_i}$ with $L_{\mathbb{G}_i}$ the Laplacian matrix of a connected bidirectional graph $\mathbb{G}_i$. Then there must be a constant vectors $x_i^* \in \mathbb{R}^{n_i}$ such that

$$\bar{\boldsymbol{x}}_i^* = \mathbf{1}_{c_i} \otimes x_i^*. \tag{3.70}$$

This and (3.65) imply

$$x_{i1}^* = x_{i2}^* = \cdots = x_{ic_i}^* = x_i^* \tag{3.71}$$

for $i = 1, 2, \cdots, c$.

From (3.70), the partition of A_i in (3.39) and $\bar{A}_i = \text{diag}\,\{A_{i1}, \cdots, A_{ic_i}\}$ in (3.46), one has

$$\bar{A}_i\bar{\boldsymbol{x}}_i^* = A_i x_i^* \tag{3.72}$$

From (3.67) and the definitions of $\hat{A}, \hat{b}, \hat{L}_{\mathbb{G}}$, one has

$$\begin{bmatrix} \bar{A}_1\bar{\boldsymbol{x}}_1^* \\ \bar{A}_2\bar{\boldsymbol{x}}_2^* \\ \vdots \\ \bar{A}_c\bar{\boldsymbol{x}}_c^* \end{bmatrix} - \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_c \end{bmatrix} - (L_{\mathbb{G}} \otimes I_m)\hat{z} = 0 \tag{3.73}$$

This equality and (3.72) imply

$$\begin{bmatrix} A_1x_1^* \\ A_2x_2^* \\ \vdots \\ A_cx_c^* \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_c \end{bmatrix} - (L_{\mathbb{G}} \otimes I_m)\hat{z} = 0 \tag{3.74}$$

Premultiplying by $\mathbf{1}_c' \otimes I_m$ on both sides of (3.74), one has

$$\sum_{i=1}^{c}(A_ix_i^* - b_i) - [(\mathbf{1}_c'L_{\mathbb{G}}) \otimes I_m]\hat{z} = 0. \tag{3.75}$$

Note that $L_{\mathbb{G}}$ is the Laplacian matrix of a c-node connected and bidirectional graph $\mathbb{G}$, one has $\mathbf{1}_c'L_{\mathbb{G}} = 0$. Thus

$$\sum_{i=1}^{c}(A_ix_i^* - b_i) = 0. \tag{3.76}$$

From (3.71) and (3.76), one sees all x_{ij}^* satisfy the local consensus (3.42) and the global conservation (3.43). Therefore all $x_{ij}(t)$ converge to constant vectors and thus form a conservation-consensus solution $\boldsymbol{x}^*$ to $Ax = b$. This completes the proof. ■

3.4.3 Validation

We utilize the double-layer network as in Fig. 3.1 to solve the linear equation $A\boldsymbol{x} = b$, which is partitioned according to the structure in Fig. 3.5 with details as follows:

$$A = \left[\begin{array}{cc|cc|c} 1 & 2 & 1 & 1 & 2 \\ \hline 2 & -1 & 3 & -3 & 0 \\ \hline -6 & 2 & 6 & 1 & -2 \\ \hline 2 & -6 & 3 & 7 & 0 \\ \hline 4 & -4 & 4 & 8 & 2 \end{array}\right], b = \begin{bmatrix} 8 \\ 8 \\ 11 \\ -17 \\ -9 \end{bmatrix} = \left[\begin{array}{c} 8 \\ \hline 4 \\ \hline 0 \\ \hline -5 \\ \hline -5 \end{array}\right] + \left[\begin{array}{c} 0 \\ \hline 4 \\ \hline 5 \\ \hline -9 \\ \hline -1 \end{array}\right] + \left[\begin{array}{c} 0 \\ \hline 0 \\ \hline 6 \\ \hline -3 \\ \hline -3 \end{array}\right]$$

Suppose each agent i_j knows A_{ij} and b_{ij}, and we employ the updates (3.44) and (3.45) with arbitrary initializations. Let

$$V(t) = \frac{1}{2}\sum_{i=1}^{c}\sum_{j=1}^{c_i}\|x_{ij}(t) - x_i^*\|_2^2$$

where $\boldsymbol{x}^* = \begin{bmatrix} 0.77 & 2.79 & 1.98 & -1.10 & 0.38 \end{bmatrix}'$ is a solution to $Ax = b$. Thus, $V(t)$ measures the closeness of all agent states to forming a conservation-consensus solution. We again utilize the double-layer network as in Fig. 3.3, with the following partition:

$$A = \left[\begin{array}{c:cc:cc} 1 & 2 & 1 & 1 & 2 \\ 2 & -1 & 3 & -3 & 0 \\ \hdashline -6 & 2 & 6 & 1 & -2 \\ \hdashline 2 & -6 & 3 & 7 & 0 \\ \hdashline -3 & -4 & 0 & -2 & 6 \end{array}\right], b = \left[\begin{array}{c} 8 \\ 8 \\ \hdashline 11 \\ \hdashline -17 \\ \hdashline -9 \end{array}\right] = \begin{bmatrix} 4 \\ 2 \end{bmatrix} + \begin{bmatrix} 3 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 5 \end{bmatrix}$$
$$= [5] + [0] + [6]$$
$$= [-9] + [-8]$$
$$= [-4] + [-5] + [0] + [0] + [0]$$

Simulations are shown in Fig. 3.6, which suggests that $V(t)$ converges exponentially fast to 0 and thus all $x_{ij}(t)$ converge exponentially fast to constant vectors that form a conservation-consensus solution $\boldsymbol{x}^*$ in both cases. This is in accord with Theorem 3.4.1. Again, more partitions require more clusters for implementing the proposed updates, and this is expected to lead to slower convergence, as suggested by Fig. 3.6.

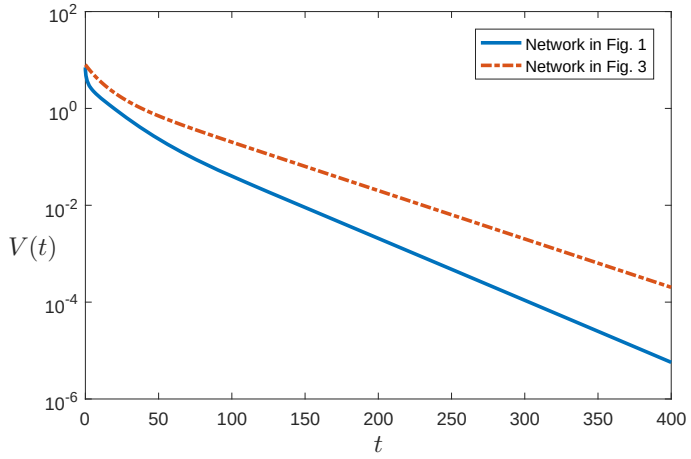

Figure 3.6. Evolution of $V(t)$ under distributed updates (3.44)-(3.45)

3.5 A Simplified Network Structure under Homogeneous Partition

In this section, we will consider a homogeneous partition of the overall equation $Ax = b$ as in Fig. 3.7 and will show that the two distributed algorithms developed in previous sections boil down to be identical; moreover, it can be implemented in a single-layered grid network $\mathbb{G}$ as in Fig. 3.8.

Let A be partitioned into r blocked rows and c blocked columns with each $A_{ij} \in \mathbb{R}^{m_i \times n_j}$ for $i = 1, 2, \cdots, r$ and $j = 1, 2, \cdots, c$; correspondingly, the b vector is partitioned into sub-vectors b_{ij} such that $b = \text{col}\,\{b_1, \cdots, b_r\} = b$ with $\sum_{j=1}^{c} b_{ij} = b_i$. One example of such a homogeneous partition is shown in Fig. 3.7 with $r = 3$ and $c = 4$. Consider a grid network $\mathbb{G}$ consisting of a number of rc agents labeled as i_j with $i = 1, 2, \cdots, r$ and $j = 1, 2, \cdots, c$, as in Fig. 3.8. Let $\mathbb{G}_i^R$ denote the ith row-subnetwork of $\mathbb{G}$, which consists of agents $i_1, i_2, \cdots, i_c$ and all edges among these agents; and let $\mathbb{G}_j^C$ denote the jth column-subnetwork of $\mathbb{G}$, which consists of agents $1_j, 2_j, \cdots, r_j$ and all edges among these agents. Suppose $\mathbb{G}_i^R$, $i = 1, 2, \cdots, r$, and $\mathbb{G}_j^C$, $j = 1, 2, \cdots, c$, are all undirected and connected. Suppose each agent i_j knows $A_{ij} \in \mathbb{R}^{m_i \times n_j}$ and $b_{ij} \in \mathbb{R}^{m_i}$, and controls a state vector $x_{ij}(t) \in \mathbb{R}^{n_i}$. The problem of interest in this section is to develop a distributed update for each $x_{ij}(t)$ converges exponentially fast to a constant x_{ij}^* such that

$$\textbf{Row-Conservation:} \quad \sum_{j=1}^{c} (A_{ij} x_{ij}^* - b_{ij}) = 0, \ \forall i = 1, 2, \cdots, r. \tag{3.77}$$

and

$$\textbf{Column-Consensus:} \quad x_{1j}^* = \cdots = x_{rj}^* = x_j^*, \ \forall j = 1, 2, \cdots, c. \tag{3.78}$$

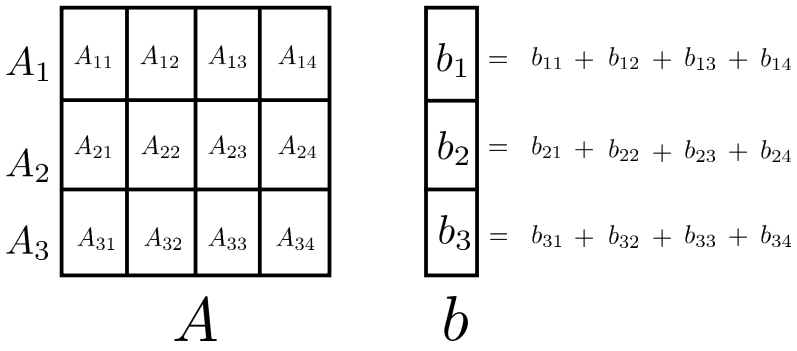

Figure 3.7. A homogeneous partition of the equation, with $r = 3$, $c = 4$.

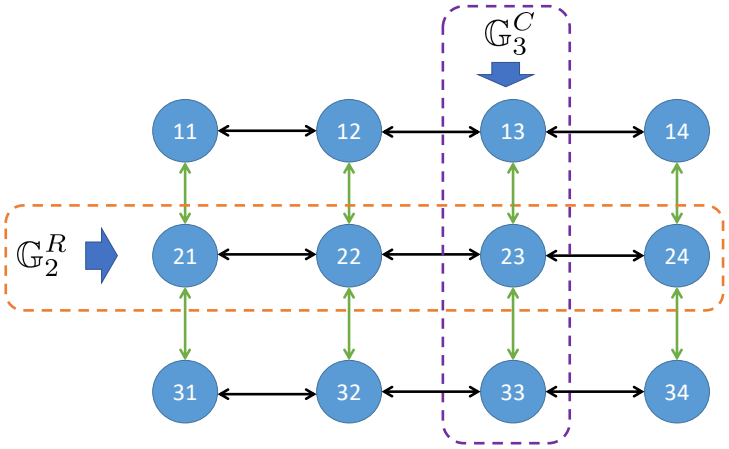

Figure 3.8. A single-layered grid network without clusters or aggregators.

3.5.1 The Update

Motivated by the two distributed updates developed in previous sections, we propose the following

$$\dot{x}_{ij} = -A'_{ij}\left(A_{ij}x_{ij} - b_{ij} - \sum_{ik\in\mathcal{N}_{ij}^R}(z_{ij} - z_{ik})\right) - \sum_{kj\in\mathcal{N}_{ij}^C}(x_{ij} - x_{kj}) \tag{3.79}$$

$$\dot{z}_{ij} = A_{ij}x_{ij} - b_{ij} - \sum_{ik\in\mathcal{N}_{ij}^R}(z_{ij} - z_{ik}) \tag{3.80}$$

where $z_{ij} \in \mathbb{R}^{m_i}$ is an additional state vector introduced agent at i_j for achieving row-conservation; $\mathcal{N}_{ij}^R$ and $\mathcal{N}_{ij}^C$ denote the neighbor set of agent i_j in $\mathbb{G}_{ij}^R$ and $\mathbb{G}_{ij}^C$, respectively.

3.5.2 Main result

Note that the only difference between distributed updates (3.79)-(3.80) and distributed updates (3.8)-(3.9) is that $\sum_{kj\in\mathcal{N}_{ij}^C}(x_{ij}-x_{kj})$ replaces $\sum_{k\in\mathcal{N}_i}(x_{ij}-E_{ij}\boldsymbol{x}_k)$. By looking at each $\mathbb{G}_i^R$ in the single-layered grid as a cluster i, one sees that $E_{ij}\boldsymbol{x}_k$ plays the same role as x_{kj}. Thus distributed updates (3.8)-(3.9) become the distributed updates (3.79)-(3.80) in the single-layered grid network. By Theorem 3.3.1, one has local conservations in (3.4) and the global consensus (3.5) , which are equivalent to (3.77) and (3.78), respectively, in the single-layered grid network. It follows that $\boldsymbol{x}^* = \text{col}\{x_1^*,\cdots,x_c^*\}$ is a solution to $Ax = b$. Similar conclusion can also be drawn by looking at distributed updates (3.79)-(3.80) as a special case of (3.44)-(3.45). To sum up, one has

Corollary 3.5.1 *Under the distributed updates* (3.79) *and* (3.80) *in a single-layered grid network, all* $x_{ij}(t)$ *with* $i = 1, 2, \cdots, r$*;* $j = 1, \cdots, c$ *converge exponentially fast to constant vectors* x_{ij}^* *satisfying (3.77) and (3.78) and thus* $\boldsymbol{x}^* = \text{col}\{x_1^*,\cdots,x_c^*\}$ *is a solution to* $Ax = b$*.*

Remark 3.5.1 *The distributed updates in a single-layered grid network in this section can only be applied to the case that the overall linear equation is partitioned homogeneously and the number of agents is equal to the number of partitions. For more general partitions, one needs to go back to (3.8)-(3.9) or (3.44)-(3.45), for which dimensions of* x_{ij} *or* z_{ij} *are not the same.*

3.5.3 Validation

According to the pattern of Fig. 3.7, we utilize a network of 100 agents with $r = c = 10$. Consider a random linear equation $Ax = b$, where $A \in \mathbb{R}^{1000\times 1000}, b \in \mathbb{R}^{1000}$. The matrix A is partitioned into 100 blocks with each $A_{ij} \in \mathbb{R}^{10\times 10}$, the vector b is partitioned accordingly with each $b_{ij} \in \mathbb{R}^{10}$. Define

$$V(t) = \frac{1}{2}\sum_{i=1}^{10}\sum_{j=1}^{10} \|x_{ij}(t) - x_i^*\|_2^2$$

where $\boldsymbol{x}^* = \text{col}\,\{x_1^*, \cdots, x_c^*\}$ is the true solution to the linear equation. Simulations

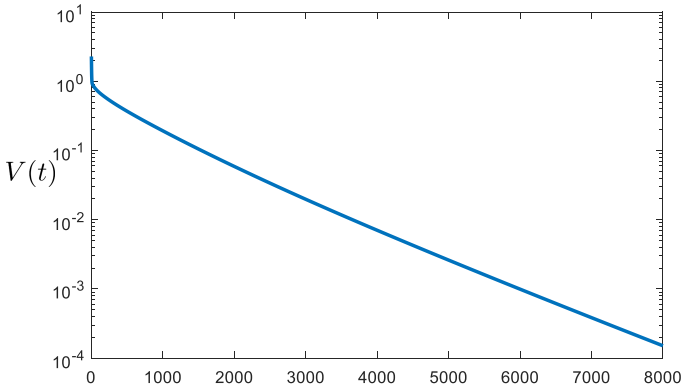

Figure 3.9. A single-layered network without clusters or aggregators.

are shown in Fig. 3.9, which suggests that $V(t)$ converges exponentially fast to 0. This validates Corollary 3.5.1.

Appendix

Proof of Lemma 3.3.1: Let λ denote any eigenvalue of M with a non-zero eigenvector col $\{\boldsymbol{u}, \bar{\boldsymbol{u}}\}$. Then

$$M\begin{bmatrix}\boldsymbol{u}\\ \bar{\boldsymbol{u}}\end{bmatrix} = \lambda\begin{bmatrix}\boldsymbol{u}\\ \bar{\boldsymbol{u}}\end{bmatrix} \tag{3.81}$$

with

$$M = \begin{bmatrix} -M_1'M_1 - M_2 & M_1'M_3 \\ \\ M_1 & -M_3 \end{bmatrix}.$$

Let $\bar{M} = \begin{bmatrix} I & 0 \\ 0 & M_3' \end{bmatrix} M$. Then one has

$$\bar{M} = \begin{bmatrix} -M_1'M_1 - M_2 & M_1'M_3 \\ \\ M_3'M_1 & -M_3'M_3 \end{bmatrix}$$

which can be written as

$$\bar{M} = -\begin{bmatrix} M_1' \\ -M_3' \end{bmatrix}\begin{bmatrix} M_1 & -M_3 \end{bmatrix} - \begin{bmatrix} M_2 & 0 \\ 0 & 0 \end{bmatrix}. \tag{3.82}$$

Thus, $\bar{M}$ is negative semi-definite. Premultiplying by $\begin{bmatrix} \boldsymbol{u} \\ \bar{\boldsymbol{u}} \end{bmatrix}' \begin{bmatrix} I & 0 \\ 0 & M_3' \end{bmatrix}$ on both sides of (3.81), one has

$$\begin{bmatrix} \boldsymbol{u} \\ \bar{\boldsymbol{u}} \end{bmatrix}' \bar{M} \begin{bmatrix} \boldsymbol{u} \\ \bar{\boldsymbol{u}} \end{bmatrix} = \lambda \begin{bmatrix} \boldsymbol{u} \\ \bar{\boldsymbol{u}} \end{bmatrix}' \begin{bmatrix} I & 0 \\ 0 & M_3 \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \bar{\boldsymbol{u}} \end{bmatrix} \tag{3.83}$$

First, we prove that λ must be real by contradiction. Suppose $\lambda = \alpha + \beta \boldsymbol{i}$ where $\beta \neq 0$. Since $\bar{M}$ is negative semi-definite, then the imaginary part of the left-hand side of (3.83) is 0. So therefore is the imaginary part of the right-hand side. It follows that

$$\beta \begin{bmatrix} \boldsymbol{u} \\ \bar{\boldsymbol{u}} \end{bmatrix}' \begin{bmatrix} I & 0 \\ 0 & M_3 \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \bar{\boldsymbol{u}} \end{bmatrix} = 0$$

Since $\beta \neq 0$ there follows $\boldsymbol{u}'\boldsymbol{u} + \bar{\boldsymbol{u}}' M_3 \bar{\boldsymbol{u}} = 0$. Recall that M_3 positive semi-definite. Hence $\boldsymbol{u} = 0, \quad M_3\bar{\boldsymbol{u}} = 0$. Taken with (3.81) and noting $\lambda \neq 0$ since $\beta \neq 0$, one has $\bar{\boldsymbol{u}} = 0$. This and the assumption that $\boldsymbol{u} = 0$ contradict to the fact that col $\{\boldsymbol{u}, \bar{\boldsymbol{u}}\}$ is non-zero. Thus $\beta = 0$. Therefore, λ is real. From this, (3.83), $\bar{M}$ is negative semi-definite and M_3 is positive semi-definite, one further has $\lambda \leq 0$.

Second, if $\lambda = 0$ is an eigenvalue of M, we prove that it must be non-defective by contradiction. Suppose $\lambda = 0$ is defective, then there exists a non-zero vector col $\{\boldsymbol{v}, \bar{\boldsymbol{v}}\}$ such that

$$M \begin{bmatrix} \boldsymbol{v} \\ \bar{\boldsymbol{v}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{u} \\ \bar{\boldsymbol{u}} \end{bmatrix} \tag{3.84}$$

and

$$M \begin{bmatrix} \boldsymbol{u} \\ \bar{\boldsymbol{u}} \end{bmatrix} = 0 \tag{3.85}$$

Premultiplying by $\begin{bmatrix} \boldsymbol{u} \\ \bar{\boldsymbol{u}} \end{bmatrix}' \begin{bmatrix} I & 0 \\ 0 & M_3' \end{bmatrix}$ on both sides of (3.84), one has

$$\begin{bmatrix} \boldsymbol{u} \\ \bar{\boldsymbol{u}} \end{bmatrix}' \bar{M} \begin{bmatrix} \boldsymbol{v} \\ \bar{\boldsymbol{v}} \end{bmatrix} = (\boldsymbol{u}'\boldsymbol{u} + \bar{\boldsymbol{u}}' M_3' \bar{\boldsymbol{u}}) \tag{3.86}$$

Premultiplying by $\begin{bmatrix} I & 0 \\ 0 & M_3' \end{bmatrix}$ on both sides of (3.85), one has

$$\bar{M} \begin{bmatrix} \boldsymbol{u} \\ \bar{\boldsymbol{u}} \end{bmatrix} = 0 \tag{3.87}$$

This and the fact that $\bar{M}$ is symmetric imply that the left hand side of (3.86) is 0. Then

$$(\boldsymbol{u}'\boldsymbol{u} + \bar{\boldsymbol{u}}' M_3' \bar{\boldsymbol{u}}) = 0 \tag{3.88}$$

from which, using the fact that M_3 is positive semi-definite, one has

$$\boldsymbol{u} = 0, \quad M_3' \bar{\boldsymbol{u}} = 0. \tag{3.89}$$

Premultiplying by $\begin{bmatrix} \boldsymbol{v} \\ \bar{\boldsymbol{v}} \end{bmatrix}' \begin{bmatrix} I & 0 \\ 0 & M_3' \end{bmatrix}$ on both sides of (3.84) one has

$$\begin{bmatrix} \boldsymbol{v} \\ \bar{\boldsymbol{v}} \end{bmatrix}' \bar{M} \begin{bmatrix} \boldsymbol{v} \\ \bar{\boldsymbol{v}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{v}'\boldsymbol{u} \\ \bar{\boldsymbol{v}}' M_3' \bar{u} \end{bmatrix}$$

The right-hand side is 0 by (3.89). Thus

$$\begin{bmatrix} \boldsymbol{v} \\ \bar{\boldsymbol{v}} \end{bmatrix}' \bar{M} \begin{bmatrix} \boldsymbol{v} \\ \bar{\boldsymbol{v}} \end{bmatrix} = 0, \tag{3.90}$$

Together with (76), this yields

$$M_1\boldsymbol{v} - M_3\bar{\boldsymbol{v}} = 0, \quad M_2\boldsymbol{v} = 0. \tag{3.91}$$

From this and the definition of M, one has

$$M \begin{bmatrix} \boldsymbol{v} \\ \bar{\boldsymbol{v}} \end{bmatrix} = 0,$$

By (3.84), this yields col $\{\boldsymbol{u}, \bar{\boldsymbol{u}}\} = 0$, contradicting the assumption that col $\{\boldsymbol{u}, \bar{\boldsymbol{u}}\}$ is a non-zero eigenvector. Thus, $\lambda = 0$ is non-defective. ■

4. CONSENSUS-BASED DISTRIBUTED OPTIMIZATION FOR MULTI-AGENT SYSTEMS [1]

4.1 Introduction

Distributed optimization is one of the key problems in multi-agent coordination, where the goal is to minimize the sum of local objective functions, there being one function associated with each agent [62, 102–104]. Specifically, consider a network of m agents, in which each agent i is able to communicate with certain other nearby agents called its neighbors, denoted by $\mathcal{N}_i$. The neighbor relations can be described by a graph $\mathbb{G}$ such that there is an edge from j to i if and only if $j \in \mathcal{N}_i$. Then under network $\mathbb{G}$, a distributed optimization problem has the following characteristics:

$$\textbf{Global Optimization Objective:} \quad \min \quad \sum_{i=1}^{m} f_i(x_i) \in \mathbb{R}. \tag{4.1}$$

subject to

$$\textbf{Consensus Rule:} \quad x_1 = x_2 \cdots = x_m, \tag{4.2}$$

where $f_i(\cdot)$ is is associated with the local performance index for each agent, which is usually assumed to be convex and continuously differentiable. Considering the fact that each state x_i, in some cases, can only be chosen from a certain domain, we also introduce the following

$$\textbf{Local Constraints:} \quad x_i \in \mathcal{X}_i \subset \mathbb{R}^n, \ i = 1, 2, ..., m. \tag{4.3}$$

where each local constraint $\mathcal{X}_i$ is convex and $\bigcap_{i=1}^{m} \mathcal{X}_i \neq \emptyset$.

[1] Resarch in this section has been published in the papers [62,101] with me as the leading author.

Among the abundant literature, great efforts have been made, attempting to solve the distributed optimization problem characterized by (4.1)-(4.3). The key idea to enable distributed coordination is consensus (4.2), which has the following form [105]

$$\dot{x}_i = -\sum_{j\in\mathcal{N}_i}(x_i - x_j), \tag{4.4}$$

where each agent in the network tries to reduce the distances between itself and its neighbors. Update (4.4) will drive all states to a consensus value, i.e. there exists a certain $\boldsymbol{x}^* = \mathbf{1}_m \otimes x^*$, such that $x_i - x^* \to 0$ for all i and the convergence is exponentially fast. Then, to handle the optimization goal (4.1), the common method to solve the optimization problem is gradient descent. Thus, Leaving aside temporarily the constraints (4.3), the attempt to find an algorithm also achieving the optimization objective (4.1) requires the introduction of a gradient term, so that the update of each agent becomes

$$\dot{x}_i = -\alpha(t)\nabla f_i(x_i) - \sum_{j\in\mathcal{N}_i}(x_i - x_j), \tag{4.5}$$

where $\alpha(t)$ is a positive step-size shared by all the agents and $\nabla f_i(x_i)$ is a sub-gradient[2] of $f_i(x_i)$. Note that if $\alpha(t)$ is chosen as a fixed positive constant, unless all f_i are minimized by a common vector, it is clear that there can be no steady state x^* satisfying this equation which also has the consensus property.

In order to solve this issue, a novel algorithm is presented in [37–39], where the authors smartly apply a diminishing step-size $\alpha(t)$ to the discrete-time version of (4.5) to eliminate the consensus error. Furthermore, by an additional projection operator, this algorithm is also able to handle local constraints. A continuous version the algorithm developed in [37–39] is

$$\dot{x}_i = -\mathcal{P}_i\left[-\alpha(t)\nabla f_i(x_i) - \sum_{j\in\mathcal{N}_i}(x_i - x_j)\right], \tag{4.6}$$

[2]For a given convex function $f_i(\cdot) : \mathbb{R}^n \to \mathbb{R}$, $\nabla f_i(x_i)$ is called a sub-gradient of $f_i(\cdot)$ at x_i if $f_i(x_i+a) \geq f_i(x_i)+a^\top \nabla f_i(x_i)$ holds for all vectors a in its domain. Specially, if $f_i(\cdot)$ is continuously differentiable at x_i, there holds $\nabla f_i(x_i) \triangleq \frac{\partial f_i(x_i)}{\partial x_i}$, and it is unique.

where $\mathcal{P}_i[\cdot]$ is a projection operator that projects any vector to the tangent space[3] of the agent's local constraint at the point $x_i(t)$, this guarantee that x_i always satisfies the local constraint of agent i. By letting all agents to share a diminishing step-size such that $\alpha(t) \to 0$ and $\int_0^\infty \alpha(t) \to \infty$, it has been theoretically proved that the states will asymptotically reach a consensus at the minimizer of $\sum_{i=1}^n f_i(x)$ subject to all agents' local constraints [37, 38]. Meanwhile, since the effect of gradient term is discounted by the diminishing step-size, the convergence rate of the algorithm is at most $\mathcal{O}(1/\sqrt{t})$. Except for these, many recent works have shown that by doubling the dimension of the state vector, the exponential convergence rate and exact optimal solution can be achieved simultaneously in a fully distributed fashion [41–43]. However, this idea will lead to the following sides effects

- For the distributed algorithms equipped with doubled state vectors, since the primal and dual vectors are interactive, it is very difficult to choose step-sizes to guarantee that the algorithm is stable.
- In these algorithms, the extra states have to be exchanged across the network, necessitating duplication of the network bandwidth requirement.

In this chapter, we aim to propose new approached that can solve the mentioned issues.

4.2 A Distributed Algorithm for Least Squares Solutions

4.2.1 The Problem

In Chapters 2-3, we have introduced distributed algorithms for solving linear equations. These results are based on a critical assumption that is the original linear equation has at least one solution. Thus, these algorithms are not directly applicable

[3]For the linear constraints $A_i x_i = b_i$, one has $\mathcal{P}_i[s] = P_i \cdot s$, where $P_i \in \mathbb{R}^{n\times n}$ is a projection matrix to $\ker A_i$

to over-determined linear equations, which usually arise in many engineering applications such as parameter estimation [8], mode estimation in power networks [106] and real-time data fitting of financial models [107]. One idea for dealing with over-determined linear equations is briefly discussed in [19], in which the size of an agent's state increases with the number of agents in the network; this does not scale well with the network size. The projection-consensus flow proposed in [40, 108, 109] achieves a least squares solution only in a neighborhood of an exact one. A distributed algorithm for a least squares solution in [110] introduces a decaying weight to the local gradient, which causes the loss of exponential convergence.

$$\text{minimize} \qquad \frac{1}{2}\sum_{i=1}^{m} |A_i x_i - b_i|_2^2 \tag{4.7}$$

$$\text{subject to} \qquad x_1 = x_2 = \cdots = x_m. \tag{4.8}$$

Another common approach to achieve a least squares solution is distributed optimization. Along this direction, ADMM [9, 111] could be applied here, but usually needs a centralized agent for coordination. By a combination of consensus and ADMM, researchers have developed a fully distributed ADMM [112], which requires a specific order for all agents to perform updates and does not converge exponentially fast. Recent distributed optimization algorithms [37, 113–115] are able to achieve exact least squares solutions exponentially fast, but the convergence relies on all agents to share a common, time-varying small step size. Such a requirement is also usually introduced when discretizing classical continuous algorithms for least squares solutions in [41, 116, 117].

4.2.2 The Update

In this section, we present a discrete-time, distributed algorithm for all agents in a multi-agent network to achieve a common least squares solution exponentially fast.

Let $\mathcal{N}_i$ denote the set of neighbors of agent i. Since i is a neighbor of itself, $i \in \mathcal{N}_i$. Define $W \in \mathbb{R}^{m\times m}$ to be an *adjacency matrix* for the graph $\mathbb{G}$, that is,

$$w_{ij} \begin{cases} > 0 & \text{if} \quad j \in \mathcal{N}_i \\ = 0 & \text{if} \quad j \notin \mathcal{N}_i \end{cases} \tag{4.9}$$

We assume that $w_{ji} = w_{ij}$ for all i, j, that is, W is symmetric and each diagonal element of W is positive. Let $D \in \mathbb{R}^{m\times m}$ be the diagonal matrix with the i-th diagonal entry d_i given by

$$d_i = \sum_{j=1}^{n} w_{ij} = \sum_{j\in\mathcal{N}_i} w_{ij} \tag{4.10}$$

Then, the *Laplacian* (matrix) for $\mathbb{G}$ is given by

$$L = D - W \tag{4.11}$$

which is symmetric and positive semi-definite. Since $\mathbb{G}$ is connected, the kernel of L is spanned by $\mathbf{1}_m$ [118]. Let $\bar{L} = L \otimes I_n \in \mathbb{R}^{mn\times mn}$ where $\otimes$ denotes the Kronecker product. It follows that the kernel of $\bar{L}$ is the image of $\mathbf{1}_m \otimes I_n$. Then $\boldsymbol{x}^*$ satisfies

$$\bar{L}\boldsymbol{x}^* = 0 \tag{4.12}$$

if and only if

$$\boldsymbol{x}^* = \mathbf{1}_m \otimes x^* \tag{4.13}$$

for some $x^* \in \mathbb{R}^n$. Our first result motivates our proposed algorithm.

Lemma 4.2.1 A vector $x^* \in \mathbb{R}^n$ is a least squares solution to (4.7)-(4.8) if and only if for any arbitrary positive constant c, there exists a vector $\boldsymbol{z}^* \in \mathbb{R}^n$ such that

$$c(\bar{A}'\bar{A}\boldsymbol{x}^* - \bar{A}'b) + \bar{L}'\boldsymbol{z}^* = 0 \tag{4.14}$$

with $\boldsymbol{x}^*$ is given by (4.13) and

$$\bar{A} = \text{diag}\,\{A_1, A_2, \cdots, A_m\}. \tag{4.15}$$

A proof of Lemma 4.2.1 is in the Appendix. Since $\boldsymbol{x}^*$ has the structure given (4.13) if and only if it satisfies (4.12), Lemma 4.2.1 implies that the problem of obtaining a least squares solution x^* to (4.7)-(4.8) is equivalent to finding $\boldsymbol{x}^*$ and $\boldsymbol{z}^*$ satisfying (4.14) and (4.12). Letting $\boldsymbol{x}^* = \text{col}\,\{x_1^*, x_2^*, \cdots, x_m^*\}$ and $\boldsymbol{z}^* = \text{col}\,\{z_1^*, z_2^*, \cdots, z_m^*\}$ with each $x_i^*, z_i^* \in \mathbb{R}^n$, and noticing that $\bar{L}$ is symmetric, (4.14) and (4.12) are equivalent to

$$c(A_i' A_i x_i^* - A_i' b_i) + \sum_{j \in \mathcal{N}_i} w_{ij} \left(z_i^* - z_j^*\right) = 0 \tag{4.16}$$

$$\sum_{j \in \mathcal{N}_i} w_{ij} \left(x_i^* - x_j^*\right) = 0 \tag{4.17}$$

for $i = 1, 2, \cdots, m$. So, we have reduced the original problem to that of solving (4.16) and (4.17) in a distributed fashion. To accomplish this we have introduced one additional state vector $z_i(t) \in \mathbb{R}^n$ at each agent i.

A distributed solution to (4.16)-(4.17) can be achieved by using the following continuous-time saddle-point dynamics [41]:

$$\dot{x}_i(t) = -c\left[A_i' A_i x_i(t) - A_i' b_i\right] - \sum_{j \in \mathcal{N}_i} w_{ij}\left[z_i(t) - z_j(t)\right] \tag{4.18}$$

$$\dot{z}_i(t) = \sum_{j \in \mathcal{N}_i} w_{ij}\left[x_i(t) - x_j(t)\right] \tag{4.19}$$

for $i = 1, \cdots, m$ where c is an arbitrary positive constant. A simple discretization of (4.18)-(4.19) can be achieved by replacing $\dot{x}_i(t)$ and $\dot{z}_i(t)$ with $\frac{x_i(t+1) - x_i(t)}{\Delta t}$ and $\frac{z_i(t+1) - z_i(t)}{\Delta t}$, respectively. This method usually requires a careful choice of a sufficiently small step size Δt for convergence [119], which can hardly be accomplished in a distributed way. Moreover, such a discretization usually comes with the cost of losing exponential convergence. In the following, we present a way to remove such small step size while maintaining exponential convergence of the proposed distributed algorithm.

Motivated by the implicit-explicit iteration methods (IMEX) of [120], we replace $x_i(t)$ and $z_i(t)$ in the right-hand side of (4.18)-(4.19) with $x_i(t+1)$ and $z_i(t+1)$, respectively. The mixed use of $x_i(t+1), z_i(t+1)$ and $x_j(t), z_j(t), j \in \mathcal{N}_i$ on the right-hand side to (4.18)-(4.19) enables us to replace $\dot{x}_i(t)$ and $\dot{z}_i(t)$ in the left-hand side by

$\frac{x_i(t+1)-x_i(t)}{\Delta t}$ and $\frac{z_i(t+1)-z_i(t)}{\Delta t}$, respectively, where $\Delta t = \frac{1}{d_i}$. This results in the following discrete-time algorithm. Let c be an arbitrary positive constant and for $i = 1, \cdots, m$,

$$\begin{aligned} x_i(t+1) = x_i(t) &- \frac{c}{d_i}\left[A_i' A_i x_i(t+1) - A_i' b_i\right] \\ &- \frac{1}{d_i}\sum_{j\in\mathcal{N}_i} w_{ij}\left[z_i(t+1) - z_j(t)\right] \end{aligned} \tag{4.20}$$

$$z_i(t+1) = z_i(t) + \frac{1}{d_i}\sum_{j\in\mathcal{N}_i} w_{ij}\left[x_i(t+1) - x_j(t)\right] \tag{4.21}$$

that is,

$$\begin{bmatrix} x_i(t+1) \\ z_i(t+1) \end{bmatrix} = F_i^{-1}\begin{bmatrix} d_i x_i(t) + \sum\limits_{j\in\mathcal{N}_i} w_{ij} z_j(t) + cA_i' b_i \\ -\sum\limits_{j\in\mathcal{N}_i} w_{ij} x_j(t) + d_i z_i(t) \end{bmatrix} \tag{4.22}$$

where

$$F_i = \begin{bmatrix} d_i I_n + cA_i' A_i & d_i I_n \\ -d_i I_n & d_i I_n \end{bmatrix} \tag{4.23}$$

Note that the state update includes a matrix multiplicity of $F_i^{-1} \in \mathbb{R}^{2n\times 2n}$, where F_i^{-1} always exists and can be achieved by simply computing the inverse of a $n_i \times n_i$ matrix. As

$$F_i = d_i \begin{bmatrix} I_n & I_n \\ -I_n & I_n \end{bmatrix} + \begin{bmatrix} A_i' \\ 0 \end{bmatrix} cI_{n_i} \begin{bmatrix} A_i & 0 \end{bmatrix}$$

Using the Woodbury matrix identity [121], we obtain that

$$\begin{aligned} F_i^{-1} = &-\frac{1}{4d_i^2}\begin{bmatrix} I_n \\ I_n \end{bmatrix} A_i' \left(\frac{1}{c} I_{n_i} + \frac{1}{2d_i} A_i A_i'\right)^{-1} A_i \begin{bmatrix} I_n & I_n \end{bmatrix} \\ &+ \frac{1}{2d_i}\begin{bmatrix} I_n & -I_n \\ I_n & I_n \end{bmatrix} \end{aligned} \tag{4.24}$$

The matrix in parentheses is invertible because it is positive definite. Equation (4.24) implies that computing F_i^{-1} involves computing the inverse of a $n_i \times n_i$ matrix. Here n_i is the number of rows of A_i, which is usually small, and can even be equal to 1. Moreover, this inverse only needs to be computed once by each agent since F_i is fixed

for each agent. Another possible way of implementing update (4.22) is to factorize F_i and then solve triangular systems instead of computing F_i^{-1} explicitly [121].

Remark 4.2.1 *Note immediately that the update (4.22) is distributed since each agent i only uses A_i, b_i and the states of its neighbors. The number of state variables controlled by each agent is $2n$, which is independent of the underlying network size. Although it is a discrete-time algorithm, a small step-size is not required for convergence. The positive parameter c introduced here is for adjusting the convergence rate, which can be simply chosen as $c = 1$. As long as c is strictly positive, exponential convergence is guaranteed as shown in next section.*

4.2.3 Main Result

In this section, we demonstrate the exponential convergence of the proposed distributed algorithm (4.22). With

$$\bar{W} = W \otimes I_n, \qquad \bar{D} = D \otimes I_n \tag{4.25}$$

and

$$\boldsymbol{x} = \text{col}\left\{x_1, x_2, \cdots, x_m\right\},\ \boldsymbol{z} = \text{col}\left\{z_1, z_2, \cdots, z_m\right\}, \tag{4.26}$$

algorithm (4.22), or equivalently (4.20)-(4.21), can be written as

$$\begin{aligned} \boldsymbol{x}\left(t+1\right) = \boldsymbol{x}(t) - c\bar{D}^{-1}\left[\bar{A}'\bar{A}\boldsymbol{x}\left(t+1\right) - \bar{A}'b\right] \\ - \bar{D}^{-1}\left[\bar{D}\boldsymbol{z}\left(t+1\right) - \bar{W}\boldsymbol{z}\left(t\right)\right] \end{aligned} \tag{4.27}$$

$$\boldsymbol{z}\left(t+1\right) = \boldsymbol{z}\left(t\right) + \bar{D}^{-1}\left[\bar{D}\boldsymbol{x}\left(t+1\right) - \bar{W}\boldsymbol{x}\left(t\right)\right] \tag{4.28}$$

or $F\boldsymbol{y}(t+1) = \bar{Q}\boldsymbol{y}(t) + \bar{\boldsymbol{h}}$ where $\boldsymbol{y} = \text{col}\left\{\boldsymbol{x}, \boldsymbol{z}\right\}$ and

$$F = \begin{bmatrix} \bar{D} + c\bar{A}'\bar{A} & \bar{D} \\ -\bar{D} & \bar{D} \end{bmatrix},\ \bar{Q} = \begin{bmatrix} \bar{D} & \bar{W} \\ -\bar{W} & \bar{D} \end{bmatrix},\ \bar{\boldsymbol{h}} = \begin{bmatrix} c\bar{A}'b \\ 0 \end{bmatrix}. \tag{4.29}$$

We see that F has the same structure as F_i in (4.23) and one may readily show that it is invertible. Thus,

$$\boldsymbol{y}(t+1) = Q\boldsymbol{y}(t) + \boldsymbol{h} \tag{4.30}$$

where

$$Q = F^{-1}\bar{Q}, \quad \boldsymbol{h} = F^{-1}\bar{\boldsymbol{h}} \tag{4.31}$$

Convergence of system (4.30) depends on the properties of Q, which are summarized as follows.

Lemma 4.2.2 (a) For any $\lambda \in$ eig (Q), if $\lambda \neq 1$, then $|\lambda| < 1$.

(b) $1 \in$ eig (Q) and is non-defective, that is, its algebraic multiplicity is equal to its geometric multiplicity.

A proof of Lemma 4.2.2 is in the Appendix. This lemma tells us that all non-unity eigenvalues of Q have magnitudes less than one, that is,

$$\rho_2(Q) := \max\{|\lambda| : \lambda \in \text{eig } (Q),\ \lambda \neq 1\} < 1\,. \tag{4.32}$$

We now have the main result of the section.

Theorem 4.2.1 Suppose $\mathbb{G}$ is undirected and connected, and $c > 0$ is any positive constant. Then under the distributed algorithm (4.22), all $x_i(t)$, $i = 1, 2, ..., m$, converge exponentially fast to the same least squares solution to (4.7)-(4.8), as fast as $\rho_2(Q)^t \to 0$.

The effectiveness of the proposed distributed algorithm (4.22) is validated by Theorem 4.2.1. Detailed comparisons with existing distributed algorithms for achieving least squares solutions are given in Table I.

Proof of Theorem 4.2.1: Suppose that $x_i(t), z_i(t), i = 1, \cdots, m$ is any solution to (4.22). Then, $\boldsymbol{y}(t) = \text{col}\,\{\boldsymbol{x}(t), \boldsymbol{z}(t)\}$ is a solution to (4.30) where $\boldsymbol{x}$ and $\boldsymbol{z}$ are given by (4.26). Since $\bar{L} = \bar{D} - \bar{W}$ and (4.30) is equivalent to (4.27)-(4.28), we see that

$\boldsymbol{y}^* = \text{col}\,\{\boldsymbol{x}^*, \boldsymbol{z}^*\}$ is an equilibrium state to (4.30) if and only if $\boldsymbol{x}^*$ and $\boldsymbol{z}^*$ satisfy (4.14) and (4.12). By Lemma 4.2.1 and noting that (4.12) is equivalent to (4.13), one has proving Theorem 4.2.1 can be achieved by showing that there exists a constant vector $\bar{\boldsymbol{y}}^*$ such that

$$\boldsymbol{y}(t) \to \bar{\boldsymbol{y}}^* \text{ as fast as } \rho_2(Q)^t \to 0. \tag{4.33}$$

and

$$(I - Q)\bar{\boldsymbol{y}}^* = \boldsymbol{h} \tag{4.34}$$

Let x^* be any least squares solution to (4.7)-(4.8). Then Lemma 4.2.1 tells us that there exists $\boldsymbol{z}^*$ such that $\boldsymbol{x}^* = \mathbf{1}_m \otimes x^*$ and $\boldsymbol{z}^*$ satisfy (4.14). Thus $\boldsymbol{y}^* = \text{col}\,\{\boldsymbol{x}^*, \boldsymbol{z}^*\}$ is an equilibrium state for (4.30). Hence

$$(I - Q)\boldsymbol{y}^* = \boldsymbol{h}. \tag{4.35}$$

and recalling (4.30), the evolution of $\boldsymbol{e}(t) = \boldsymbol{y}(t) - \boldsymbol{y}^*$ is governed by

$$\boldsymbol{e}(t+1) = Q\boldsymbol{e}(t). \tag{4.36}$$

Lemma 4.2.2 tells us that one is a non-defective eigenvalue of Q and the magnitude of any other eigenvalue of Q is less than one. This implies that exists a non-singular matrix T such that

$$Q = T \begin{bmatrix} I & 0 \\ 0 & R \end{bmatrix} T^{-1} \tag{4.37}$$

Table 4.1. Comparing Algorithm (4.27)-(4.28) with existing algorithms

Algorithm	**Distributed Network**	**Convergence Rate**	**Step Size**
The proposed algorithm	applicable	Exponential	not required
ADMM [9, 111]	require centralized agent	Exponential	fixed, assigned by central agents
Distributed-ADMM [112]	specific update order	$\mathcal{O}(1/t)$	fixed, shared by all agents
Projection Flow [40, 108, 109]	applicable, not exact solution	Exponential	fixed, require global information
Improved Projection Flow [37]	applicable	$\mathcal{O}(1/t)$	time-varying
Methods of [113–115]	applicable	Exponential	time-varying, global information

and the eigenvalues of R are the eigenvalues of Q which are not equal to 1. Then $\boldsymbol{e}(t) = \boldsymbol{e}^* + \boldsymbol{\eta}(t)$ with

$$\boldsymbol{e}^* = T \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} T^{-1}\boldsymbol{e}(0), \qquad \boldsymbol{\eta}(t) = T \begin{bmatrix} 0 & 0 \\ 0 & R^t \end{bmatrix} T^{-1}\boldsymbol{e}(0) \tag{4.38}$$

where $R^t \to 0$ as fast as $\rho_2(Q)^t \to 0$. Thus one has (4.33) with $\bar{\boldsymbol{y}}^* = \boldsymbol{y}^* + \boldsymbol{e}^*$. Equations (4.37) and (4.38) imply that $Q\boldsymbol{e}^* = \boldsymbol{e}^*$, which and (4.35) imply (4.34). We complete the proof. ■

4.2.4 Validation

In this section, we provide numerical simulations for a five-agent network as shown in Fig. 4.1 to illustrate Theorem 4.2.1. The weights of all edges are set to be 1. We

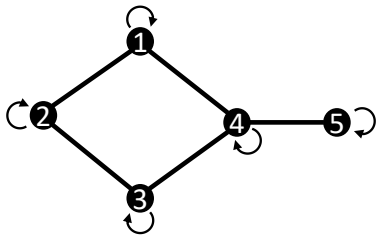

Figure 4.1. A five-agent connected network

wish to solve the least squares problem (4.7)-(4.8) in a distributed fashion with

$$\begin{aligned}
A_1 &= \begin{bmatrix} 1 & 2 & 1 & 1 \end{bmatrix}, & b_1 &= 10 \\
A_2 &= \begin{bmatrix} 2 & -1 & -1 & 1 \end{bmatrix}, & b_2 &= 20 \\
A_3 &= \begin{bmatrix} 1 & -2 & 4 & -1 \end{bmatrix}, & b_3 &= 15 \\
A_4 &= \begin{bmatrix} -1 & -0.6 & 0.4 & 1.8 \end{bmatrix}, & b_4 &= 17 \\
A_5 &= \begin{bmatrix} 2 & 2 & -2 & 1 \end{bmatrix}, & b_5 &= 11.
\end{aligned}$$

Here (4.7)-(4.8) has a unique solution $x^* = \begin{bmatrix} 5.51 & -3.38 & 2.91 & 10.10 \end{bmatrix}$. Suppose each agent i in the multi-agent network in Fig. 4.1 updates its state according to the proposed distributed update (4.22). Let

$$V(t) = \frac{1}{2m} \sum_{i=1}^{m} |x_i(t) - x^*|_2^2 \tag{4.39}$$

measure the average distance between agents' states and the unique least squares solution x^*. Simulation results with different choices of c are as shown in Fig. 4.2, in which the vertical axis uses a log scale, and the curve marked by IPF represents the convergence of the Improved Projection Flow method proposed in [37].

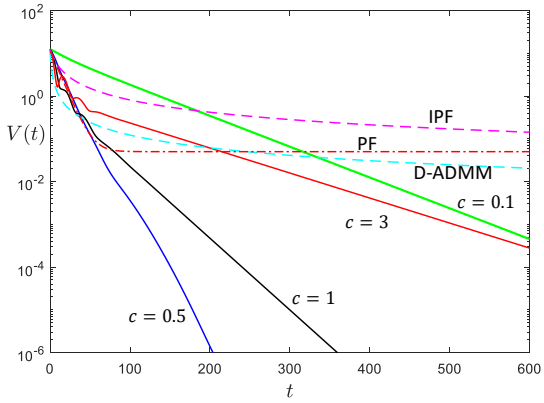

Figure 4.2. Simulations in the case of unique least squares solutions with different choices of c.

The evolution of V in Fig. 4.2 suggests that the proposed distributed update (4.22) drives all agents' states exponentially fast to the unique least squares solution x^*. Note that the choice of c impacts the rate of convergence, which results from the fact that the choice of c impacts the eigenvalues of Q in (4.30). For simplicity, one could always choose $c = 1$ and still guarantee exponential convergence. To find an optimal choice for c to achieve the fastest rate of exponential convergence, one needs to find c to minimize $\rho_2(Q)$, which is not trivial and usually requires global information such as the whole matrix Q. Moreover, the simulations in Fig. 4.2 indicates that the proposed algorithm in (4.22) converges faster than the Distributed-ADMM ($\beta = 0.2$) and Improved Projection Flow (IPF), whose convergence rate are

$\mathcal{O}(1/t)$ [37, 112]. Even though the Projection Flow (PF) method has an exponential convergence rate [40], it can only achieve an approximation but not the exact least squares solution to the equation.

The simulations in Fig. 4.2 show the effectiveness of the proposed algorithm in (4.22) when the least squares solution is unique. To demonstrate the effectiveness of the algorithm in the case of multiple least squares solutions, we replace A_2, A_3 with $A_2 = \begin{bmatrix} 1 & 1.4 & -1.6 & 2.8 \end{bmatrix}$ and $A_3 = \begin{bmatrix} 3 & 3.4 & -3.6 & 3.8 \end{bmatrix}$, respectively. Then (4.7)-(4.8) has multiple least squares solutions. Simulations in Fig. 4.3 suggests that

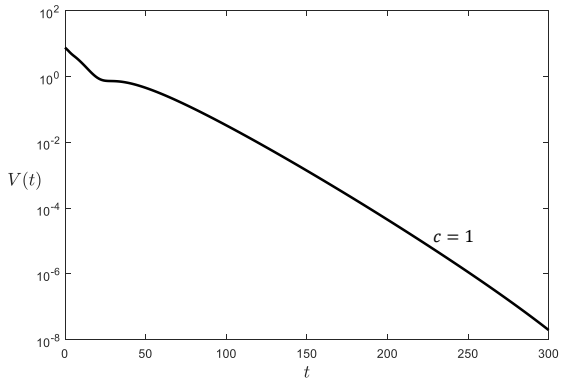

Figure 4.3. Simulations in the case of multiple least squares solutions with $c = 1$

all $x_i(t)$ converges to the same $x^* = \begin{bmatrix} -1.34 & 1.14 & 1.89 & 7.18 \end{bmatrix}$, which is one least squares solution to (4.7)-(4.8).

Appendix

Proof of Lemma 4.2.1. We first note that x^* satisfies (4.7)-(4.8) if and only if

$$x^* = \arg \min_{x \in \mathbb{R}^{nm}} \frac{1}{2} |Ax - b|_2^2 \tag{4.40}$$

where

$$A = \text{col}\,\{A_1, A_2, \cdots, A_m\},\ b = \text{col}\,\{b_1, b_2, \cdots, b_m\} \tag{4.41}$$

Also x^* satisfies (4.40) if and only if $A'Ax^* = A'b$, that is,

$$\sum_{i=1}^{m}(A_i'A_ix^* - A_i'b_i) = 0. \tag{4.42}$$

With $\boldsymbol{x}^* = \mathbf{1}_m \otimes x^*$, (4.42) is equivalent to

$$(\mathbf{1}_m' \otimes I_n)(\bar{A}'\bar{A}\boldsymbol{x}^* - \bar{A}'b) = 0. \tag{4.43}$$

that is,

$$\bar{A}'\bar{A}\boldsymbol{x}^* - \bar{A}'b \in \ker(\mathbf{1}_m' \otimes I_n) \tag{4.44}$$

Recall that the kernel of $\bar{L} = L \otimes I_n$ is the image of $\mathbf{1}_m \otimes I_n$; hence

$$\ker(\mathbf{1}_m' \otimes I_n) = \text{image } \bar{L}' \tag{4.45}$$

It now follows from (4.44) and (4.45) that (4.42) is equivalent to

$$\bar{A}'\bar{A}\boldsymbol{x}^* - \bar{A}'b \in \text{image } \bar{L}'. \tag{4.46}$$

This is equivalent to that for any $c > 0$, there exists a vector $\boldsymbol{z}^*$ such that

$$c(\bar{A}'\bar{A}\boldsymbol{x^*} - \bar{A}'b) + \bar{L}'\boldsymbol{z}^* = 0. \tag{4.47}$$

■

Proof of Lemma 4.2.2. For any $\lambda \in \text{eig } (Q)$, there is a nonzero vector $\boldsymbol{v}$ such that $Q\boldsymbol{v} = \lambda\boldsymbol{v}$. Since $Q = F^{-1}\bar{Q}$ with F and $\bar{Q}$ given in (2.52), we must have

$$\begin{bmatrix} \bar{D} & \bar{W} \\ -\bar{W} & \bar{D} \end{bmatrix} \boldsymbol{v} = \lambda \begin{bmatrix} \bar{D} + c\bar{A}'\bar{A} & \bar{D} \\ -\bar{D} & \bar{D} \end{bmatrix} \boldsymbol{v}$$

that is,

$$\left[\bar{D} - \lambda\left(\bar{D} + c\bar{A}'\bar{A}\right)\right]\boldsymbol{u} = \left(\lambda\bar{D} - \bar{W}\right)\bar{\boldsymbol{u}} \tag{4.48}$$

$$\left(\lambda\bar{D} - \bar{W}\right)\boldsymbol{u} = (\lambda - 1)\,\bar{D}\bar{\boldsymbol{u}}. \tag{4.49}$$

where $\boldsymbol{v} = \text{col } \{\boldsymbol{u}, \bar{\boldsymbol{u}}\}$.

Proof of (a) Suppose $\lambda \neq 1$ is an eigenvalue of Q. Then (4.49) and (4.48) are equivalent to

$$\bar{\boldsymbol{u}} = \frac{1}{\lambda - 1}\bar{D}^{-1}\left(\lambda\bar{D} - \bar{W}\right)\boldsymbol{u} \tag{4.50}$$

$$M(\lambda)\boldsymbol{u} = 0 \tag{4.51}$$

where

$$M(\lambda) = \lambda^2 M_2 + \lambda M_1 + M_0 \tag{4.52}$$

and

$$\begin{aligned} M_0 &= \bar{W}\bar{D}^{-1}\bar{W} + \bar{D} \\ M_1 &= -2(\bar{D} + \bar{W}) - c\bar{A}'\bar{A} \\ M_2 &= 2\bar{D} + c\bar{A}'\bar{A} \end{aligned} \tag{4.53}$$

are symmetric and $M_2 > 0$ since $\bar{D} > 0$. Equation (4.51) implies that $\boldsymbol{u}'M(\lambda)\boldsymbol{u} = 0$, that is,

$$p(\lambda) := c_2\lambda^2 + c_1\lambda + c_0 = 0 \tag{4.54}$$

where

$$c_i = \boldsymbol{u}'M_i\boldsymbol{u} \quad \text{for } i = 0, 1, 2 \tag{4.55}$$

are real and $c_2 > 0$ since $M_2 > 0$. Since $c_2 > 0$, using properties of second order polynomials (Jury stability criterion) , $|\lambda| < 1$ if and only if

$$c_0 - c_2 < 0 \tag{4.56}$$

$$|c_1| < c_0 + c_2 \tag{4.57}$$

From (4.53) and (4.55), one has $|c_1| = \boldsymbol{u}'(2\bar{D} + 2\bar{W} + c\bar{A}'\bar{A})\boldsymbol{u}$ and $c_0 + c_2 = \boldsymbol{u}'(\bar{W}\bar{D}^{-1}\bar{W} + 3\bar{D} + c\bar{A}'\bar{A})\boldsymbol{u}$. Hence, (4.57) holds if and only if

$$\boldsymbol{u}'(\bar{W}\bar{D}^{-1}\bar{W} - 2\bar{W} + \bar{D})\boldsymbol{u} > 0$$

that is.

$$((\bar{D}-\bar{W})\boldsymbol{u})'\bar{D}^{-1}((\bar{D}-\bar{W})\boldsymbol{u})>0 \tag{4.58}$$

Inequality (4.58) holds unless $\bar{W}\boldsymbol{u}=\bar{D}\boldsymbol{u}$. In this case, (4.54) implies that

$$p(\lambda)=\left((\boldsymbol{u}'\bar{D}\boldsymbol{u}+c\boldsymbol{u}\bar{A}'\bar{A}\boldsymbol{u})\lambda-\boldsymbol{u}'\bar{D}\boldsymbol{u}\right)(\lambda-1)=0 \tag{4.59}$$

Since $\lambda\neq 1$ the above equation is not satisfied unless $\bar{A}\boldsymbol{u}\neq 0$ in which case it is satisfied with $|\lambda|<1$.

From (4.53) and (4.55), inequality (4.56) is equivalent to

$$\boldsymbol{u}'\left(\bar{D}-\bar{W}\bar{D}^{-1}\bar{W}\right)\boldsymbol{u}+c\boldsymbol{u}'\left(\bar{A}'\bar{A}\right)\boldsymbol{u}>0. \tag{4.60}$$

Recall that $\bar{D}-\bar{W}=\bar{L}\geq 0$; hence $\bar{D}+\bar{W}=\bar{L}+2\bar{D}>0$.

Thus $-I\leq\bar{D}^{-\frac{1}{2}}\bar{W}\bar{D}^{-\frac{1}{2}}<I$ and

$$(\bar{D}^{-\frac{1}{2}}\bar{W}\bar{D}^{-\frac{1}{2}})^2\leq I\,. \tag{4.61}$$

Pre- and post-multiplying (4.61) by $\bar{D}^{\frac{1}{2}}$ yields

$$\bar{D}-\bar{W}\bar{D}^{-1}\bar{W}\geq 0. \tag{4.62}$$

Thus, inequality (4.60) holds unless

$$(\bar{D}-\bar{W}\bar{D}^{-1}\bar{W})\boldsymbol{u}=0 \tag{4.63}$$

$$\bar{A}\boldsymbol{u}=0 \tag{4.64}$$

Since $\bar{D}-\bar{W}\bar{D}^{-1}W=(\bar{D}+\bar{W})\bar{D}^{-1}(\bar{D}-\bar{W})$ and $\bar{D}+\bar{W}>0$, (4.63) is equivalent to

$$\bar{W}\boldsymbol{u}=\bar{D}\boldsymbol{u} \tag{4.65}$$

Recalling (4.59) we obtain that

$$p(\lambda)=\boldsymbol{u}'\bar{D}\boldsymbol{u}(\lambda-1)^2=0$$

Since $\lambda\neq 1$ and $\boldsymbol{u}\neq 0$, this inequality does not hold; hence inequality (4.56) holds and $|\lambda|<1$.

Proof of (b) It follows from (4.48) and (4.49) and $\bar{D} - \bar{W} = \bar{L}$ that one is an eigenvalue of Q if and only if there is a vector col $\{\boldsymbol{u}, \bar{\boldsymbol{u}}\} \neq 0$ such that

$$-c\bar{A}'\bar{A}\boldsymbol{u} = \bar{L}\bar{\boldsymbol{u}} \tag{4.66}$$

$$\bar{L}\boldsymbol{u} = 0 \tag{4.67}$$

Clearly this is satisfied with $\boldsymbol{u} = 0$ and $\bar{\boldsymbol{u}} = \mathbf{1}_m \otimes z$ for any $z \in \mathbb{R}^n$. Thus one is an eigenvalue for Q

To prove that one is non-defective, suppose on the contrary that it is defective. Then there is a vector col $\{\boldsymbol{v}, \bar{\boldsymbol{v}}\} \neq 0$ [121] such that

$$(Q - I) \begin{bmatrix} \boldsymbol{v} \\ \bar{\boldsymbol{v}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{u} \\ \bar{\boldsymbol{u}} \end{bmatrix} \tag{4.68}$$

where col $\{\boldsymbol{u}, \bar{\boldsymbol{u}}\}$ is an eigenvector corresponding to eigenvalue one. Recalling the definition of Q in (4.31) and (2.52), (4.68) implies that

$$-c\bar{D}^{-1}\bar{A}'\bar{A}\boldsymbol{v} - \left(\bar{D}^{-1}\bar{W} - I_{mn}\right)\bar{\boldsymbol{v}} = \left(I_{mn} + c\bar{D}^{-1}\bar{A}'\bar{A}\right)\boldsymbol{u} + \bar{\boldsymbol{u}}$$

$$(I_{mn} - \bar{D}^{-1}\bar{W})\boldsymbol{v} = -\boldsymbol{u} + \bar{\boldsymbol{u}}$$

and using, $\bar{L} = \bar{D} - \bar{W}$,

$$-c\bar{A}'\bar{A}\boldsymbol{v} + \bar{L}\bar{\boldsymbol{v}} = \left(\bar{D} + c\bar{A}'\bar{A}\right)\boldsymbol{u} + \bar{D}\bar{\boldsymbol{u}} \tag{4.69}$$

$$\bar{L}\boldsymbol{v} = -\bar{D}\boldsymbol{u} + \bar{D}\bar{\boldsymbol{u}} \tag{4.70}$$

Pre-multiplying (4.69) by $\boldsymbol{u}$ and (4.70) by $\bar{\boldsymbol{u}}'$, using (4.66) and (4.67) and the symmetry of $\bar{L}$ results in

$$\bar{\boldsymbol{u}}'\bar{L}\bar{\boldsymbol{v}} = c\boldsymbol{u}'\bar{A}'\bar{A}\boldsymbol{u} + \boldsymbol{u}'\bar{D}\boldsymbol{u} + \boldsymbol{u}'\bar{D}\bar{\boldsymbol{u}}$$

$$\bar{\boldsymbol{u}}'\bar{L}\bar{\boldsymbol{v}} = -\bar{\boldsymbol{u}}'\bar{D}\boldsymbol{u} + \bar{\boldsymbol{u}}'\bar{D}\bar{\boldsymbol{u}}.$$

Hence

$$\boldsymbol{u}'\bar{D}\boldsymbol{u} + \bar{\boldsymbol{u}}'\bar{D}\bar{\boldsymbol{u}} = -c\boldsymbol{u}'\bar{A}'\bar{A}\boldsymbol{u} \leq 0 \tag{4.71}$$

Since $\bar{D} > 0$, we obtain the contradiction that $\boldsymbol{u} = \bar{\boldsymbol{u}} = 0$. Thus one is a non defective eigenvalue. ■

4.3 Distributed Optimization Enhanced by Integral Feedback

4.3.1 The Problem

In this Section, we propose a distributed algorithm for constrained optimization that is neither based on diminishing step-sizes nor a doubled dimension of the vectors shared between agents. Actually, by comparing the very fundamental mechanisms of these algorithms, we notice that the common reason why the latter category of the algorithms can achieve an improved convergence performance arises from elevating the type of the update to the second order, and thereby can effectively eliminate the accumulated consensus error. Inspired by this, in this section, we propose a continuous-time consensus-based algorithm for constrained distributed optimization based on integral feedback within each agent's controller, and with the integrated signal not being shared with other agents. Without a time-variant step-size that needs to be shared by agents, the algorithm is capable of achieving the optimum solution with an exponential convergence rate. Furthermore, inherited from the benefit of integral feedback, the proposed algorithm has good robustness against disturbance. Note that the algorithm of this section is evidently related to a discrete-time algorithm for unconstrained optimization [42]. These authors increased the state dimension by using a form of gradient descent including the last two iterates. On the other hand as noted already, our algorithm is motivated by the very old principle of using integral feedback to cancel steady state errors. Different from [42], our algorithm can additionally handle local linear constraints, which commonly exists in many engineering application [122, 123]. Also, apart from requiring the storing at each agent of the integral of the state vector in addition to the state vector itself, *the algorithm does not introduce an extra state vector which has to be exchanged among the agents of the network*. This further distinguishes the work with the existing results based on (primal–dual) saddle point dynamics in [44–46].

Consider a network of m agents, we assume $\mathbb{G}$ is *connected* and *undirected*. Associated with each agent is a local state $x_i \in \mathbb{R}^n$; a convex function $f_i(\cdot) : \mathbb{R}^n \to \mathbb{R}$;

and a linear constraint $A_i x_i = b_i$, where $A_i \in \mathbb{R}^{n_i \times n}$ and $b_i \in$ image $A_i \subset \mathbb{R}^{n_i}$. The problem of *interest* is to develop a *distributed* algorithm which enables all nodes of $\mathbb{G}$ to reach a consensus value such that

$$\text{minimize} \qquad \sum_{i=1}^{n} f_i(x_i). \tag{4.72}$$

$$\text{subject to} \qquad A_i x_i = b_i, \tag{4.73}$$

$$x_1 = x_2 = \cdots = x_m. \tag{4.74}$$

Remark 4.3.1 *The linear constraint in* (4.73) *arises naturally from many engineering applications [122, 123]. Here, to avoid trivialities, we assume* $b_i \in$ image A_i, *and* rank $(A) < n$, *where* $A = \text{col}\,\{A_1, \cdots, A_m\}$. *This guarantees the optimization domain defined by equations* (4.73)-(4.74) *is non-empty and non-unique.*

4.3.2 The Update

Here, instead of using a diminishing step-size or an additional state vector to each agent which has to be exchanged with neighbors, our key idea stems from introducing an additional integral term to effectively eliminate the steady state error on consensus. We propose the following continuous-time distributed algorithm,

$$\dot{x}_i = -P_i \left(\nabla f_i(x_i) + \sum_{j \in \mathcal{N}_i} (x_i - x_j) + \int_0^t \sum_{j \in \mathcal{N}_i} (x_i - x_j) \right) \tag{4.75}$$

where $x_i(0)$ are initialized such that $A_i x_i(0) = b_i$; and $P_i \in \mathbb{R}^{n \times n}$ is a projection matrix to $\ker A_i$.

Table 4.2. Comparing Algorithm (4.75) with existing algorithms

Algorithm	Key idea	State	Local constraints	Update	Exponential Convergence
The proposed algorithm	Integral Feedback	Dime. n	Linear, closed	Continuous	Yes
Algorithms in [38]	Diminishing step-size	Dim. n	Compact	Discrete	No, $\mathcal{O}(1/\sqrt{t})$
Algorithms in [42]	Gradient tracking	Dim. $2n$	Not applicable	Discrete	Yes
Algorithm in [41]	Saddle point dynamics	Dim. $2n$	Not applicable	Continuous	Yes
Algorithms in [44–46]	Saddle point dynamics	Dim. $2n$	Closed	Continuous	No theoretical guarantee

Remark 4.3.2 *Obviously, the proposed algorithm is distributed, because the state update of each agent only relies on the information of itself and that of its neighbors. In update* (4.75)*, the projection matrix* P_i *and the special initialization on* $x_i(0)$ *are used to handle the linear constraint* (4.73)*. As a special case, if for one or more agents, the linear constraint does not exist, then one can correspondingly initialize* $x_i(0)$ *as an arbitrary value and replace the projection matrix by an identity matrix.*

4.3.3 Main Result

Algorithm (4.75) allows us to propose the main result by the following Theorem.

Theorem 4.3.1 *Suppose* $\mathbb{G}$ *is connected and undirected,* $f_i(x)$ *is convex for* $i = 1, \cdots, m$ *and* $F(x) = \sum_{i=1}^{m} f_i(x)$ *is strongly convex* [4] *and twice differentiable at* x^**, where* x^* *is the minimizer of* $\sum_{i=1}^{n} f_i(x)$ *subject to* $A_i x^* = b_i$*,* $i = 1, \cdots, m$*. Then given any* $x_i(0)$ *such that* $A_i x_i(0) = b_i$*, update* (4.75) *drives the states of each agent exponentially fast to* x^**.*

In the following, we will prove Theorem 4.3.1.

Steady-state Analysis

We first propose the following lemma, which characterizes the existence and uniqueness properties of the equilibrium point of (4.75).

Lemma 4.3.1 *Consider the updates* (4.75)*, where* $\mathbb{G}$ *is connected and undirected;* $f_i(x)$ *is convex for* $i = 1, \cdots, m$ *and* $F(x) = \sum_{i=1}^{m} f_i(x)$ *is strongly convex and twice differentiable at* x^**, where* x^* *is the minimizer of* $\sum_{i=1}^{n} f_i(x)$ *subject to* $A_i x^* = b_i$*,* $i = 1, \cdots, m$*. Then an equilibrium point* x_i^* *of the equations exists and is unique. Furthermore, for all* $i = 1, \cdots, m$*, there holds* $x_i^* = x^*$*, ensuring that the equilibrium*

[4] A differentiable function $F(\cdot)$ is called strongly convex at x^* with parameter $\omega > 0$ if $a^\top [\nabla F(a + x^*) - \nabla F(x^*)] \geq \omega \|a\|_2^2$ holds for all vectors a in its domain.

point obeys the consensus property and optimizes the constrained optimization problem (4.72)-(4.74).

For simplicity in analyzing the proposed algorithm from a global prospect define $\boldsymbol{x} = \text{col}\,\{x_1, \cdots, x_m\} \in \mathbb{R}^{mn}$, $\bar{P} = \text{diag}\,\{P_1, \cdots, P_m\} \in \mathbb{R}^{mn\times mn}$ and $\bar{L} = L \otimes I_n \in \mathbb{R}^{mn\times mn}$, where $L \in \mathbb{R}^{m\times m}$ is the Laplacian matrix of the graph $\mathbb{G}$. [We will frequently use below the property of L that it is symmetric with kernel spanned by $\mathbf{1_m}$, the m-vector of all 1's.] Then update (4.75) can be rewritten as

$$\dot{\boldsymbol{x}} = -\bar{P}\left(\nabla f(\boldsymbol{x}) + \bar{L}\boldsymbol{x} + \int_0^t \bar{L}\boldsymbol{x}\right). \tag{4.76}$$

This is further equivalent to

$$\dot{\boldsymbol{x}} = -\bar{P}\left(\nabla f(\boldsymbol{x}) + \bar{L}\boldsymbol{x} + \boldsymbol{y}\right) \tag{4.77}$$

$$\dot{\boldsymbol{y}} = \bar{L}\boldsymbol{x} \tag{4.78}$$

where $\boldsymbol{y} \in \mathbb{R}^{mn}$ and $\boldsymbol{y}(0) = 0$.

Remark 4.3.3 *Evidently the dynamics* (4.77)-(4.78) *is equivalent to* (4.75)*, where the integral term implicitly introduces an extra state* $\boldsymbol{y}$*. Each component of this extra state can be obtained via local computations and stored by each agent, and does not have to be exchanged across the network. For existing algorithms characterized by saddle-point dynamics, the extra states must be exchanged across the network [41,124].*

Proof of Lemma 4.3.1: In order to prove Lemma 4.3.1, since (4.77)-(4.78) and (4.75) are equivalent, it is sufficient to show that there exist equilibrium points $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ with unique $\boldsymbol{x}^*$ for (4.77)-(4.78) such that

$$0 = -\bar{P}\left(\nabla f(\boldsymbol{x}^*) + \bar{L}\boldsymbol{x}^* + \boldsymbol{y}^*\right) \tag{4.79}$$

$$0 = \bar{L}\boldsymbol{x}^* \tag{4.80}$$

where $\boldsymbol{x}^* = \mathbf{1}_m \otimes x^*$ and x^* is the minimizer of $F(x) = \sum_{i=1}^m f_i(x)$ subject to $A_i x^* = b_i$ for all i; $\nabla f(\boldsymbol{x}^*)$ is the column stack of the vectors $\left.\frac{\partial f_i(x_i)}{\partial x_i}\right|_{x^*}$. Furthermore, note that

in (4.79), the value of $\boldsymbol{y}^*$ cannot be taken arbitrarily. Since $\boldsymbol{y}(0) = 0$, the integration of $\boldsymbol{y}$ in equation (4.78) indicates that for all t, $\boldsymbol{y}(t) \in$ image $\bar{L}$. Hence, $\boldsymbol{y}^* \in$ image $\bar{L}$.

In the following, we first assume the equilibrium $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ described in (4.79)-(4.80) exists and show that $\boldsymbol{x}^*$ is unique, taking the form set out in the lemma statement. Since $\ker \bar{L} =$ image $(\mathbf{1} \otimes I_n)$, from equation (4.80), one has $x_1^* = \cdots = x_m^*$, which is the consensus property. Next, recall that image $P_i = \ker A_i$, then $\ker P_i =$ image $A_i^\top$. Let $\bar{A} = \text{diag}\,\{A_1, \cdots, A_m\}$. From equation (4.79) and $\bar{P} = \text{diag}\,\{P_1, \cdots, P_m\}$, there exists a $\boldsymbol{z}^* = \text{col}\,\{z_1^*, \cdots, z_m^*\}$, $z_i^* \in \mathbb{R}^{n_i}$ such that

$$\nabla f(\boldsymbol{x}^*) + \bar{L}\boldsymbol{x}^* + \boldsymbol{y}^* = \bar{A}^\top \boldsymbol{z}^*. \tag{4.81}$$

Further recall that $\nabla f(\boldsymbol{x}^*)$ is the column stack of the vectors $\frac{\partial f_i(x_i)}{\partial x_i}|_{x^*}$, then multiplying equation (4.81) on the left by $(\mathbf{1}_m \otimes I_n)^\top$ yields

$$(\mathbf{1}_m \otimes I_n)^\top \nabla f(\boldsymbol{x}^*) = \sum_{i=1}^m \nabla f_i(x^*) = \sum_{i=1}^m A_i^\top z_i. \tag{4.82}$$

This, by standard Lagrange multiplier theory, tells us x^* is a critical point for $F(x) = \sum_{i=1}^n f_i(x)$ on the manifold defined by $A_i x^* = b_i$ for all i. Because we have assumed that $\sum_{i=1}^n f_i(x)$ is strongly convex at x^*, x^* is unique and is a minimizer.

Now we establish the existence of $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ as an equilibrium point, which is an assumption we previously made in the proof. Because x^* minimizes $\sum_i f_i(x)$ subject to $A_i x^* = b_i$, by standard Lagrange multiplier theory, there exist Lagrange multipliers z_i^* [125] such that

$$\sum_{i=1}^m \nabla f_i(x^*) - \sum_{i=1}^m A_i^\top z_i^* = 0 \tag{4.83}$$

Using the z_i^*, we now make the definition

$$y_i^* = A_i^\top z_i^* - \nabla f_i(x^*) \tag{4.84}$$

As a side remark, we observe that the uniqueness, or otherwise, of y_i^* does not influence the result of Lemma 4.3.1, as it does not originally appear in the update (4.75). Note immediately that in (4.83), if $\left[A_1^\top, \cdots, A_m^\top\right]$ does not have linearly independent

columns, the value of z_i^* is non-unique [125]. Consequently, the value of y_i^* is also non-unique.

From equation (4.84), we have $\nabla f_i(x^*) + y_i^* = A_i^\top z_i^*$, or

$$\nabla f(\boldsymbol{x}^*) + \boldsymbol{y}^* = \bar{A}^\top \boldsymbol{z}^* \tag{4.85}$$

and indeed

$$\nabla f(\boldsymbol{x}^*) + \bar{L}\boldsymbol{x}^* + \boldsymbol{y}^* = \bar{A}^\top \boldsymbol{z}^* \tag{4.86}$$

From this, since $\bar{P}\bar{A}^\top = 0$, we obtain

$$0 = -\bar{P}(\nabla f(\boldsymbol{x}^*) + \bar{L}\boldsymbol{x}^* + \boldsymbol{y}^*) \tag{4.87}$$

This ensures the satisfaction of (4.79). Furthermore, from (4.83) and (4.84) one has

$$\begin{aligned}\sum_{i=1}^m y_i^* &= \sum_{i=1}^m \left[A_i^\top z_i^* - \nabla f_i(x^*)\right] \\ &= -\left(\sum_{i=1}^m \nabla f_i(x^*) - \sum_{i=1}^m A_i^\top z_i^*\right) = 0\end{aligned}$$

This ensures the satisfaction of $\boldsymbol{y}^* \in$ image $\bar{L}$ and completes the proof. ■

Change of coordinate frame

In order to examine the transient behavior of (4.77)-(4.78), it is convenient to make a coordinate transformation which ensures that in the new coordinates, the equilibrium point corresponding to $\boldsymbol{x}^*$ is moved to zero. Here, we change the origin of updates (4.77)-(4.78), by defining vectors $\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}$ as follows:

$$\begin{aligned}\tilde{\boldsymbol{x}} &= \boldsymbol{x} - \boldsymbol{x}^* \\ \tilde{\boldsymbol{y}} &= \boldsymbol{y} - \boldsymbol{y}^*\end{aligned} \tag{4.88}$$

where, as above, $\boldsymbol{x}^* = \mathbf{1}_m \otimes x^*$ and $\boldsymbol{y}^*$ satisfies (4.87) with a certain Lagrange multiplier $\boldsymbol{z}^*$. Note that when $\boldsymbol{y}^*$ is non-unique, one can make an arbitrary choice consistent with (4.84). Evidently,

$$\begin{aligned}\dot{\tilde{\boldsymbol{x}}} =& -\bar{P}\left(\nabla f(\tilde{\boldsymbol{x}}+\boldsymbol{x}^*) - \bar{L}(\tilde{\boldsymbol{x}}+\boldsymbol{x}^*) - (\tilde{\boldsymbol{y}}+\boldsymbol{y}^*)\right)\\ =& -\bar{P}\left([\nabla f(\tilde{\boldsymbol{x}}+\boldsymbol{x}^*) - \nabla f(\boldsymbol{x}^*)] - \bar{L}\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{y}}\right) \qquad (4.89)\\ \dot{\tilde{\boldsymbol{y}}} =& \ \bar{L}(\tilde{\boldsymbol{x}}+\boldsymbol{x}^*) = \bar{L}\tilde{\boldsymbol{x}} \qquad (4.90)\end{aligned}$$

To continue, we further modify updates (4.89)-(4.90) by a frame transformation. Since the linear equitation set $A_i x = b_i$, $i = 1, \cdots, m$ has multiple solutions, there exists a single nonzero vector in the kernel of every A_i, i.e. in the range of every P_i. Observe then that if $v = P_i y_i$ is such a vector, then $P_i v = v$, $\forall i \in \{1, \cdots, m\}$. Recalling that $\ker \bar{L} = \text{image}\ (\mathbf{1} \otimes I_n)$, then, $\bar{P}\bar{L}\bar{P}(\mathbf{1} \otimes v) = 0$, which means $\bar{P}\bar{L}\bar{P}$ is singular. Thus, there exists an orthogonal matrix $Q = \begin{bmatrix} R_1 & R_2 \end{bmatrix}$, with $R_1 \in \mathbb{R}^{mn \times \bar{n}_1}$, $R_2 \in \mathbb{R}^{mn \times \bar{n}_2}$, $\bar{n}_1 + \bar{n}_2 = mn$, such that

$$Q^\top \bar{P}\bar{L}\bar{P}Q = \begin{bmatrix} 0 & 0 \\ 0 & R_2^\top \bar{P}\bar{L}\bar{P}R_2 \end{bmatrix}, \qquad (4.91)$$

where the matrix $R_2^\top \bar{P}\bar{L}\bar{P}R_2$ is non-singular. Now define new vectors X, Y by the transformations

$$X = Q^\top \tilde{\boldsymbol{x}}, \ \ Y = Q^\top \bar{P}\tilde{\boldsymbol{y}}. \qquad (4.92)$$

Multiplying the differential equations (4.89)-(4.90) on the left, by $Q^\top$ and $Q^\top \bar{P}$, respectively, yields

$$\begin{aligned}\dot{X} =& -Q^\top \bar{P}[\nabla f(QX+\boldsymbol{x}^*) - \nabla f(\boldsymbol{x}^*)] - Q^\top \bar{P}\bar{L}\bar{P}QX\\ & - Y \qquad (4.93)\\ \dot{Y} =& \ Q^\top \bar{P}\bar{L}\bar{P}QX \qquad (4.94)\end{aligned}$$

Note that in the derivation of (4.93)-(4.94), we have replaced QX with $\bar{P}QX$. This equality holds because both $x_i(t)$ for all t and x^* are solutions to $A_i x = b_i$; then

$P_i(x_1 - x^*) = (x_1 - x^*)$, that is, $\bar{P}QX = \bar{P}\tilde{\boldsymbol{x}} = \text{col}\,\{P_1(x_1 - x^*), \cdots, P_m(x_m - x^*)\} = \tilde{\boldsymbol{x}} = QX$. Based on (4.93)-(4.94), further partition the vectors X, Y as

$$X = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}, \quad Y = \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} \tag{4.95}$$

where $X_1, Y_1 \in \mathbb{R}^{\bar{n}_1}$ and $X_2, Y_2 \in \mathbb{R}^{\bar{n}_2}$. Consider now the equations for X_1, Y_1 alone. Using equation (4.91), there results

$$\dot{X}_1 = -\, R_1^\top \bar{P}\,([\nabla f(\tilde{\boldsymbol{x}} + \boldsymbol{x}^*) - \nabla f(\boldsymbol{x}^*)] - Y_1) \tag{4.96}$$

$$\dot{Y}_1 = \; 0 \tag{4.97}$$

and

$$\begin{aligned} \dot{X}_2 = &-\, R_2^\top \bar{P}[\nabla f(QX + \boldsymbol{x}^*) - \nabla f(\boldsymbol{x}^*)] \\ &-\, R_2^\top \bar{P}\bar{L}\bar{P}R_2 X_2 - Y_2 \end{aligned} \tag{4.98}$$

$$\dot{Y}_2 = \; R_2^\top \bar{P}\bar{L}\bar{P}R_2 X_2 \tag{4.99}$$

Now observe that $Y_1(0) = 0$. [The argument is as follows. Because $Q = \begin{bmatrix} R_1 & R_2 \end{bmatrix}$ and $\bar{P}\bar{L}\bar{P}$, from equation (4.91), one has $R_1^\top \bar{P}\bar{L} = 0$. Recall that $\boldsymbol{y}(0) = 0$ and $\boldsymbol{y}^* \in$ image $\bar{L}$, then $Y_1(0) = R_1^\top \bar{P}\tilde{\boldsymbol{y}}(0) = R_1^\top \bar{P}\,(\boldsymbol{y}(0) - \boldsymbol{y}^*) = 0$.] In light of (4.97), this means that $Y_1 = 0$ for all t, furthermore, $X^\top Y = X_1^\top Y_1 + X_2^\top Y_2 = X_2^\top Y_2$.

Stability questions concerning (4.89) and (4.90) thus can be treated as stability questions concerning the equations (4.96), (4.98) and (4.99), leaving out (4.97).

Proof of Theorem 4.3.1

The proof comprises two main steps. In the first step, Lyapunov theory is used to establish asymptotic stability of (4.96), (4.98) and (4.99). In the second step, the equilibrium is shown to be locally exponentially stable through examination of the linearization of the system at the equilibrium point. Together, this implies that all trajectories of the nonlinear system converge exponentially fast to the equilibrium.

First, noting that $R_2^\top \bar{P}\bar{L}\bar{P}R_2$ is symmetric positive definite, we can define a positive definite function V of X and Y_2 as follows:

$$V = \frac{1}{2}\left[X^\top X + Y_2^\top (R_2^\top \bar{P}\bar{L}\bar{P}R_2)^{-1} Y_2\right]. \tag{4.100}$$

We will show V is a Lyapunov function from which asymptotic stability can be concluded. Computing the derivative along motions of equations (4.93)-(4.94) gives us

$$\begin{aligned}\dot{V} =& - X^\top Q^\top \bar{P}[\nabla f(QX+x^*) - \nabla f(x^*)] - X^\top Q^\top \bar{P}\bar{L}\bar{P}QX \\ & - X^\top Y + Y_2^\top (R_2^\top \bar{P}\bar{L}\bar{P}R_2)^{-1}(R_2^\top \bar{P}\bar{L}\bar{P}R_2)X_2 \\ =& - X^\top Q^\top [\nabla f(QX+x^*) - \nabla f(x^*)] - X^\top Q^\top \bar{L}QX \end{aligned} \tag{4.101}$$

Note that the last equality holds because $QX = \bar{P}QX$ and $X^\top Y = X_1^\top Y_1 + X_2^\top Y_2 = X_2^\top Y_2$.

Since $\nabla f(\boldsymbol{x}) = \text{col}\,\{\nabla f_1(x_1), \cdots, \nabla f_m(x_m)\}$ and each $f_i(x_i)$ is convex, one has $-X^\top Q^\top [\nabla f(QX+x^*) - \nabla f(x^*)] \leq 0$. Thus, $\dot{V}$ in (4.101) is non-positive. We will now apply LaSalle's Theorem [126]. Observe that, $\dot{V} = 0$ if and only if $X^\top Q^\top [\nabla f(QX + x^*) - \nabla f(x^*)] = 0$, and $\bar{L}QX = 0$. Because $\ker \bar{L} = \text{image}\,(\mathbf{1}_m \otimes I_n)$, one has $QX = \mathbf{1}_m \otimes u$, $u \in \mathbb{R}^n$. Recall also that $F(x) = \sum_{i=1}^m f_i(x)$ is strongly convex at x^*, thus, there exists a positive ω such that for any $q \in \mathbb{R}^n$,

$$\begin{aligned} & \sum_{i=1}^m \left[q^\top (\nabla f_i(q + x^*) - \nabla f_i(x^*))\right] \\ = \;& (\mathbf{1}_m \otimes q)^\top (\nabla F(q + x^*) - \nabla F(x^*)) \geq \omega \|q\|_2^2 \end{aligned} \tag{4.102}$$

Due to the convexity of each $f_i(x_i)$ such that $q^\top(\nabla f_i(q + x^*) - \nabla f_i(x^*)) \geq 0$, in (4.102), there exists at least one $j \in \{1, \cdots, m\}$ such that

$$q^\top (\nabla f_j(q + x^*) - \nabla f_j(x^*)) \geq \frac{\omega}{m}\|q\|_2^2.$$

Thus, for $QX = \mathbf{1}_m \otimes u$, $u \neq 0$,

$$\begin{aligned} X^\top Q^\top [\nabla f(QX+x^*) - \nabla f(x^*)] &= \sum_{i=1}^m u^\top [\nabla f_i(u+x^*) - \nabla f_i(x^*)] \\ \geq u^\top [\nabla f_j(u + x^*) - \nabla f_j(x^*)] &\geq \frac{\omega}{m}\|u\|_2^2 > 0 \end{aligned} \tag{4.103}$$

From equation (4.103) and the fact that Q is nonsingular, we know that $\dot{V}$ is strictly negative when $X \neq 0$. Then, by LaSalle's Theorem, the update (4.93)-(4.94) will converge asymptotically to the trajectories where $X = \text{col}\,\{X_1, X_2\}$ is identically zero. Since Q is non-singular, one concludes that the $\tilde{\boldsymbol{x}}$ in (4.89) converges asymptotically to the trajectories where $\tilde{\boldsymbol{x}} = 0$. Hence, update (4.75) converges asymptotically to its equilibrium point. Then by Lemma 4.3.1, the states of each agent converge to constant vectors x_i^* which obeys the consensus property such that for all $i = 1, \cdots, m$, $x_i^* = x^*$ and x^* is the minimizer of $F(x) = \sum_{i=1}^{n} f_i(x)$. Note that at this point, by the asymptotic convergence of $\tilde{\boldsymbol{x}} \to 0$ in (4.89)-(4.90), one can conclude that $\bar{P}\tilde{\boldsymbol{y}}$ converges asymptotically to 0. However, this is not sufficient to assert that $\tilde{\boldsymbol{y}}$, nor that $\int_0^t \bar{L}\boldsymbol{x}(\tau)d\tau = \boldsymbol{y} = \tilde{\boldsymbol{y}} + \boldsymbol{y}^*$ in (4.76), converges to a constant. In the following, we further establish the exponential convergence rate of the proposed update and with such result, show that the $\int_0^t \bar{L}\boldsymbol{x}(\tau)d\tau$ in update (4.76) converges exactly to a constant.

To further establish the exponential convergence rate of the proposed update, consider the following linear time-invariant system, which linearizes the X dynamics of (4.93)-(4.94) at its equilibrium point obtained in Lemma 4.3.1:

$$\dot{\widetilde{X}} = -Q^\top \bar{P} H_{f(\mathbf{0})} Q\widetilde{X} - Q^\top \bar{P}\bar{L}\bar{P}Q\widetilde{X} - \widetilde{Y} \tag{4.104}$$

$$\dot{\widetilde{Y}} = Q^\top \bar{P}\bar{L}\bar{P}Q\widetilde{X} \tag{4.105}$$

where $\widetilde{X},\ \widetilde{Y} \in \mathbb{R}^{mn}$; $H_{f(\mathbf{0})} \triangleq \left.\frac{\partial^2 f(\boldsymbol{x})}{\partial \boldsymbol{x}^2}\right|_{\boldsymbol{x}=\boldsymbol{x}^*} \in \mathbb{R}^{mn\times mn}$. Furthermore, according to the property of X of the nonlinear dynamics, $\widetilde{X}$ satisfies $\bar{P}Q\widetilde{X} = Q\widetilde{X}$. Partition the vectors $\widetilde{X}, \widetilde{Y}$ as $\widetilde{X} = \text{col}\,\{\widetilde{X}_1, \widetilde{X}_2\}$ and $\widetilde{Y} = \text{col}\,\{\widetilde{Y}_1, \widetilde{Y}_2\}$, where $\widetilde{X}_1, \widetilde{Y}_1 \in \mathbb{R}^{\bar{n}_1}$ and $\widetilde{X}_2, \widetilde{Y}_2 \in \mathbb{R}^{\bar{n}_2}$. As for the nonlinear system, we can drop consideration of $\widetilde{Y}_1$ and we define a positive definite Lyapunov function of $\widetilde{X}$ and $\widetilde{Y}_2$ by

$$\widetilde{V} = \frac{1}{2}\left[\widetilde{X}^\top\widetilde{X} + \widetilde{Y}_2^\top (R_2^\top \bar{P}\bar{L}\bar{P}R_2)^{-1}\widetilde{Y}_2\right] \tag{4.106}$$

Computing the derivative along motions of equations (4.104)-(4.105) and using the property of $\bar{P}Q\widetilde{X} = Q\widetilde{X}$ gives us

$$\dot{\widetilde{V}} = -\widetilde{X}^\top Q^\top H_{f(\mathbf{0})} Q\widetilde{X} - \widetilde{X}^\top Q^\top \bar{L} Q\widetilde{X} \tag{4.107}$$

Note that $\dot{\widetilde{V}} = 0$ if and only if $H_{f(\mathbf{0})}Q\widetilde{X} = 0$ and $\bar{L}Q\widetilde{X} = 0$. Because $\ker \bar{L} =$ image $(\mathbf{1}_m \otimes I_n)$, one has $Q\widetilde{X} = \mathbf{1}_m \otimes \widetilde{u}$, $\widetilde{u} \in \mathbb{R}^n$. Recall that $F(x) = \sum_{i=1}^m f_i(x)$ is strongly convex at x^*, thus

$$\left.\frac{\sum_{i=1}^m \partial^2 f_i(x)}{\partial x^2}\right|_{x=x^*} > 0 \tag{4.108}$$

That is, for any $Q\widetilde{X} \neq 0$,

$$\begin{aligned} &\widetilde{X}^\top Q^\top H_{f(\mathbf{0})} Q\widetilde{X} = (\mathbf{1}_m \otimes \widetilde{u})^\top H_{f(\mathbf{0})} (\mathbf{1}_m \otimes \widetilde{u}) \\ =\ & \widetilde{u}^\top \left(\left.\frac{\sum_{i=1}^m \partial^2 f_i(x)}{\partial x^2}\right|_{x=x^*} \right) \widetilde{u} > 0 \end{aligned} \tag{4.109}$$

This, along with the fact that Q is non-singular, tells us $\dot{\widetilde{V}} < 0$ for any nonzero X. Once again, LaSalle's Theorem delivers asymptotic stability of the system with state vector comprising the entries of $\widetilde{X}_1, \widetilde{X}_2, \widetilde{Y}_2$. But since the system is linear and time-invariant, such stability is also exponential. This, along with Hartman-Grobman theorem [127], gives us the local exponential stability of the non-linear dynamics (4.93)-(4.94) associated with X_1, X_2, Y_2. That is, there exists certain a $\delta > 0$, such that if $\|[X_1^\top(0), X_2^\top(0), Y_2^\top(0)]\|_2 \leq \delta$, then $X_1(t), X_2(t), Y_2(t)$ converges to zero exponentially fast. Recall also that the update equations for $X_1(t), X_2(t), Y_2(t)$ are globally asymptotically stable; thus, given any initial state outside the ball of radius δ surrounding the origin, the associated trajectory converges to the boundary of the ball in a finite time. More generally (and as a result), given an arbitrary but fixed compact set $\mathcal{K}$ of possible initial conditions $[X_1^\top(0), X_2^\top(0), Y_2^\top(0)]$, it follows that there exists a finite time T, depending on $\mathcal{K}$, such that every trajectory from any initial condition in $\mathcal{K}$ has reached the ball at or before time T. The combination of this and the local exponential stability implies the exponential convergence rate of

the nonlinear equations for X_1, X_2, Y_2 for all initial conditions in a fixed but arbitrary compact set $\mathcal{K}$, with the decay rate depending on $\mathcal{K}$.

To further show that the $\int_0^t \bar{L}\boldsymbol{x}(\tau)d\tau$ in update (4.76) converges to a constant, note that by equation (4.92), the exponential convergence of $X = \text{col}\,\{X_1,\ X_2\}$ leads to the exponential convergence of $\tilde{\boldsymbol{x}}$. Thus, from (4.90), the exponential decaying of $\dot{\tilde{\boldsymbol{y}}}$ implies that $\tilde{\boldsymbol{y}}$ converges to a constant exponentially fast, which also holds for $\int_0^t \bar{L}\boldsymbol{x}(\tau)d\tau$. This completes the proof. ■

4.3.4 Validation

In this section, we provide numerical simulations to validate Theorem 4.3.1. Let $m = 5$, $n = 20$, $n_i = 3$, for $i = 1, \cdots, 5$. Consider the five-agent network shown in Fig 4.4. Suppose each agent i knows a local objective function $f_i(x)$ and a local

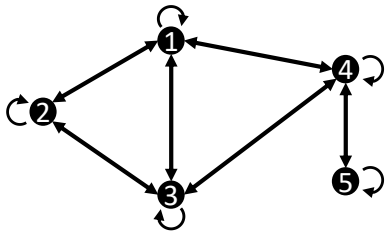

Figure 4.4. An undirected connected network of five agents.

constraint $A_i x = b_i$ such that

$$\begin{aligned} f_1(x) &= \|x\|_2^2 \qquad & f_2(x) &= \|x - c_2\|_2^2 \\ f_3(x) &= \sum_{k=1}^{20} e^{x[k]} \qquad & f_4(x) &= \sum_{k=1}^{20} e^{-2x[k]} \\ f_5(x) &= \|x - c_5\|_2^4 \end{aligned} \tag{4.110}$$

where $x[k]$ denotes the kth entry of vector x; $A_i \in \mathbb{R}^{3\times 20}$, $b_i \in \text{image}\, A_i \subset \mathbb{R}^3$ and $c_2, c_5 \in \mathbb{R}^{20}$ are constant matrices/vectors. Given the convex $f_i(x)$ defined in (4.110) and the fact that the constraint set $A_i x = b_i$, $i = 1 \cdots, 5$ is under-determined, there exists a unique solution $x^* \in \mathbb{R}^{20}$ that minimizes $F(x) = \sum_{i=1}^n f_i(x)$. Furthermore,

since $F(x)$ is strictly convex and twice differentiable everywhere in $\mathbb{R}^{20}$, it is also strictly convex and twice differentiable at x^*.

The exponential convergence rate

In order to validate Theorem 4.3.1, we let each agent initialize its local state $x_i \in \mathbb{R}^{20}$ as $A_i x_i(0) = b_i$ and then update its state by equation (4.75). Define the following function:

$$W(t) = \sum_{i=1}^{5} \|x_i(t) - x^*\|_2^2, \tag{4.111}$$

for which $W(t) = 0$ if and only if all $x_i(t) = x^*$ for all $i = 1, \cdots, 5$, where x^* is the unique minimizer of $F(x) = \sum_{i=1}^{n} f_i(x)$ subject to $A_i x = b_i$, $i = 1 \cdots, 5$. The simulation result is given by the Ode45 solver of MATLAB, and is shown in Fig. 4.5, where the Y-axis is scaled by $\log(\cdot)$. The constant slope of the curve $W(t)$ (with $W(t)$ converging to 0) indicates the exponential convergence of the algorithm, which validates Theorem 4.3.1. Note that Fig. 4.5 corresponds to the particular objective function in (4.110) of a 5-agent network, as well as a certain choice of initial states $x_i(0)$ which satisfy $A_i x_i(0) = b_i$. The convergence property shown here is absolutely representative of what happens with almost all randomly chosen examples and initial states.

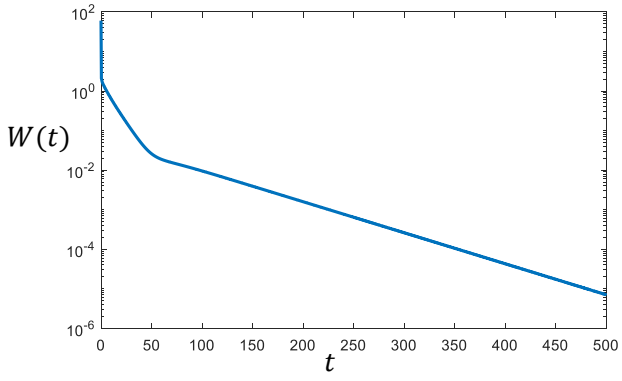

Figure 4.5. The exponential convergence rate of the proposed algorithm.

Robustness towards disturbance

In practice, due to sensor error and measurement mismatch, multi-agent systems are usually accompanied by disturbances. In this sub-section, as in [128], we investigate the impact of constant disturbance to the proposed algorithm. Consider a constant vector $\boldsymbol{v} \in \mathbb{R}^{100}$, where entry of the vector $\boldsymbol{v}$ is randomly and uniformly chosen from $[0, 0.05]$. We introduce t he constant disturbance $\boldsymbol{v}$ to the state $\boldsymbol{x}(t)$ when it is transferred through the network.

$$\dot{\boldsymbol{x}} = -\bar{P}\left(\nabla f(\boldsymbol{x}) + \bar{L}(\boldsymbol{x}+\boldsymbol{v}) + \int_0^t \bar{L}(\boldsymbol{x}+\boldsymbol{v})\right), \tag{4.112}$$

$$\dot{\boldsymbol{x}} = -\bar{P}\left(\alpha(t)\nabla f(\boldsymbol{x}) + \bar{L}(\boldsymbol{x}+\boldsymbol{v})\right), \tag{4.113}$$

where $\alpha(t)$ is chosen as $1/t$, $\bar{L} = L \otimes I_n$ is the augmented Laplacian matrix of the network, and $\boldsymbol{v} \neq 0$ is the constant disturbance we have introduced. Note that the update (4.112) corresponds to the proposed algorithm in this section, and the update (4.113) corresponds to the algorithm (4.6), where the discrete-time versions of the update are proposed in [37, 38].

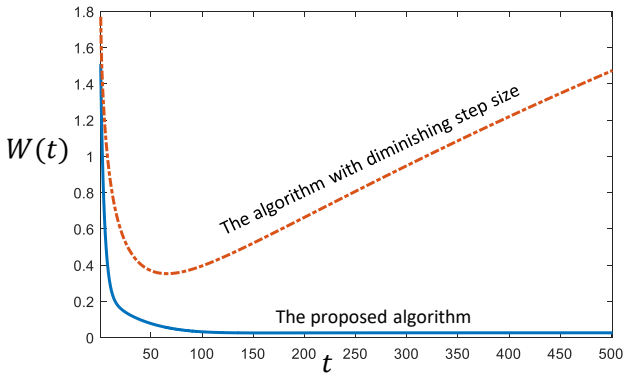

Figure 4.6. An undirected connected network of five agents.

For both algorithms, we use the same initial state the Ode45 solver of MATLAB to simulate the dynamics of (4.112) and (4.113), where obtained error curves $W(t)$ are shown in Fig. 4.6. It can be observed that for the update with diminishing step-size, the non-zero disturbance will accumulate with time t and finally leads the curve $W(t)$

to blow up. This means the agents' states are not able to converge to the optimum point x^*. For the proposed update, the curve of $W(t)$ does not grow with time t, instead, it converges to a constant value that is close to 0. This means that the states of all agents will converge to a small neighborhood of x^* (They do not converge to the exact optimum value due to the existence of the constant disturbance). By such comparison, it can be validates that update (4.112) has strikingly better robustness against disturbance than update (4.113).

5. A RESILIENT CONVEX COMBINATION FOR CONSENSUS-BASED DISTRIBUTED ALGORITHMS [1]

5.1 Introduction

Consensus-based distributed algorithms for multi-agent networks enable all agents in the network to reach an agreement regarding a certain quantity of interest, which could be an unconstrained value [129, 130], an average of all agents' initial states [131, 132] a solution to a group of linear equations [12, 13, 16, 17, 40], or a constant for optimizing an objective function [37, 39, 115]. Success of these updates heavily depends on the utilization of convex combinations of nearby neighbors' states. When one or more agents become malicious under cyber-attacks, false information will be injected into the convex combination and usually lead to failures of consensus-based distributed algorithms [47, 53, 133]. Considering the fact that many multi-agent networks in practice such as distributed power grids or robotic networks, are large-scale and often operate in open and hostile environments, the exposure to cyber-attacks is inevitable. Moreover, in a fully distributed scenario, the lack of global information makes it significantly challenging (and in some cases, impossible) to identify or isolate those malicious agents, especially when the cyber-attack is very sophisticated such as Byzantine attack [134]. Although significant progress has recently been achieved by a combination of cyber and system-theoretic approaches in [135–137], these methods are either computationally expensive, assume the network topology to be fully connected, or require the normal nodes to be aware of nonlocal information such as independent paths between themselves and other nodes. Recognition of this has motivated us to achieve a *resilient convex combination*, which refers to the convex combination of normal states that have not been manipulated by cyber-attacks, only

[1] Resarch in this section has been published in the papers [59] with me as the leading author.

knowing the upper bound to the number of malicious agents. Very nice results for achieving such a resilient convex combination have recently been developed based on Tverberg points [54–56, 58, 138]. As mentioned in [57], except for some specific values of n [57, 138], the computational complexity of achieving exact Tverberg points is usually high. Thus, one major goal of this paper is to develop an algorithm with low computational complexity for achieving a resilient convex combination. We will also apply the resilient convex combination in providing safety for consensus-based distributed algorithms in the adversarial environment.

5.2 Problem Formulation

Let $x_{\mathcal{A}} = \{x_1, x_2, \cdots, x_m\}$ denote a set of vectors in $\mathbb{R}^n$, where $\mathcal{A} = \{1, 2, ..., m\}$. Suppose one knows that at most a number of κ vectors in $x_{\mathcal{A}}$ are malicious, but the labels of malicious vectors are not known. Then there are at least a number of $p = m - \kappa$ normal vectors in $x_{\mathcal{A}}$. Suppose one knows a subset $\bar{\mathcal{A}} \subset \mathcal{A}$, which is empty or only contains labels of normal vectors in $x_{\mathcal{A}}$ but

$$|\bar{\mathcal{A}}| = \sigma \leq p. \tag{5.1}$$

The **problem of interest** is to develop an algorithm with low-computational complexity to achieve a *resilient convex combination*, which is defined as follows

Definition 5.2.1 *(resilient convex combination) A vector is a resilient convex combination of $x_{\mathcal{A}}$, if it is a convex combination of only normal vectors in $x_{\mathcal{A}}$.*

The problem is trivial when $\kappa = 0$, for which a resilient convex combination simply becomes a convex combination of $x_{\mathcal{A}}$. When $\kappa > 0$, a resilient convex combination can technically place nonzero weights only on a small number ($< p$) of normal vectors, and so trivially taking any convex combination of the elements of $x_{\bar{\mathcal{A}}}$ would yield such a combination. However, the approach that we discuss in the paper can also, in some cases, yield non-trivial convex combinations of a large number of normal vectors than just those contained in $x_{\bar{\mathcal{A}}}$, with applications in a variety of scenarios (as discussed

in the simulations section). One way to achieve such a non-trivial resilient convex combination is through Tverberg points as in [54–56, 58, 138]. For any $\mathcal{S} \subset \mathcal{A}$, let $\mathcal{H}(x_{\mathcal{S}})$ denote the convex hull of vectors in $x_{\mathcal{S}}$, that is,

$$\mathcal{H}(x_{\mathcal{S}}) = \{\sum_{k=1}^{|x_{\mathcal{S}}|} \alpha_k s_k : s_k \in x_{\mathcal{S}}, \alpha_k \geq 0, \sum_{k=1}^{|x_{\mathcal{S}}|} \alpha_k = 1\}. \tag{5.2}$$

Then the existence of Tverberg points is guaranteed by the following theorem.

Tverberg Theorem [139]: Suppose $m \geq \kappa(n+1)+1$ for the given set $x_{\mathcal{A}}$. Then there must exist a partition of $\mathcal{A}$ into $\kappa+1$ disjoint subsets $\mathcal{B}_1, \cdots, \mathcal{B}_{\kappa+1}$ such that

$$\mathcal{T} = \bigcap_{j=1}^{\kappa+1} \mathcal{H}(x_{\mathcal{B}_j}) \neq \emptyset, \tag{5.3}$$

where

$$\bigcup_{j=1}^{\kappa+1} \mathcal{B}_j = \mathcal{A}$$

and

$$\mathcal{B}_j \cap \mathcal{B}_k = \emptyset, \quad \forall j \neq k.$$

Points in the non-empty intersection $\mathcal{T}$ in (5.3) are called *Tverberg points* of the $(\kappa+1)$-partition of $\mathcal{A}$.

While results in [54, 55, 58] are elegant, one major concern of applying Tverberg points lies in the requirement of high computational complexity. As mentioned in [57], except for some specific values of n, the computational complexity of calculating Tverberg points grows exponentially with the dimension n. In the following, we will develop a low-complexity algorithm for achieving resilient convex combinations based on the intersection of convex hulls.

5.3 Resilient Convex Combination

5.3.1 A Resilient Convex Combination through Intersection of Convex Hulls

Let

$$\mathcal{R} = \bigcap_{j=1}^{r} \mathcal{H}(x_{\mathcal{A}_j}), \tag{5.4}$$

where $r = \binom{m-\sigma}{m-\sigma-\kappa}$ and $\mathcal{A}_j$, $j = 1, 2, ..., r$, denote all subsets of $\mathcal{A}$ such that

$$\bar{\mathcal{A}} \subset \mathcal{A}_j \subset \mathcal{A},\ |\mathcal{A}_j| = m - \kappa. \tag{5.5}$$

Then one has the following lemma:

Lemma 5.3.1 *If $\mathcal{R} \neq \emptyset$, then any point in $\mathcal{R}$ is a resilient convex combination.*

Proof of Lemma 5.3.1: Since the number of malicious points in $x_{\mathcal{A}}$ is upper bounded by κ, there must exist at least one subset $\mathcal{A}_{j^*}$ which consists of only normal points. As long as $\mathcal{R} \neq \emptyset$, for any vector $q \in \mathcal{R}$, it must be true that $q \in \mathcal{H}(x_{\mathcal{A}_{j^*}})$. Thus, q is a resilient convex combination. ■

Compared with the Tverberg points set in (5.3), the $\mathcal{R}$ in (5.4) defines a larger set for choosing resilient convex combinations, as indicated by the following lemma.

Lemma 5.3.2 *The set $\mathcal{T}$ in* (5.3) *and the set $\mathcal{R}$ in* (5.4) *satisfy*

$$\mathcal{T} \subset \mathcal{R}. \tag{5.6}$$

Proof of Lemma 5.3.2: We first claim that for each $\mathcal{A}_j$, $j = 1, 2, ..., r$, defined in (5.5), one of $\mathcal{B}_1, \cdots, \mathcal{B}_{\kappa+1}$ must be its subset. We prove this by contradiction. Suppose there exists a $\mathcal{A}_{j^\dagger}$ such that none of $\mathcal{B}_1, \cdots, \mathcal{B}_{\kappa+1}$ is a subset of $\mathcal{A}_{j^\dagger}$. Then each $\mathcal{B}_j$, $j = 1, 2, \cdots, \kappa + 1$, must have at least one element that is not in $\mathcal{A}_{j^\dagger}$. Note that the sets $\mathcal{B}_1, \cdots, \mathcal{B}_{\kappa+1}$ are disjoint. Then there are at least $\kappa + 1$ elements that are not in $\mathcal{A}_{j^\dagger}$. Then $|\mathcal{A}_{j^\dagger}| \leq m-\kappa-1$, which contradicts the fact that $|\mathcal{A}_{j^\dagger}| = m-\kappa$. Thus for each $\mathcal{A}_j$, one of $\mathcal{B}_1, \cdots, \mathcal{B}_{\kappa+1}$ must be a subset of $\mathcal{A}_j$. From this and the

definition of $\mathcal{R}$ in (5.4), one has $\bigcap_{j=1}^{\kappa+1} \mathcal{H}(x_{\mathcal{B}_j}) \subset \mathcal{R}$, which is (5.6). This completes the proof. ■

To guarantee that $\mathcal{R}$ is not empty, one has the following lemma:

Lemma 5.3.3 *If $\bar{\mathcal{A}} \neq \emptyset$, then $\mathcal{R} \neq \emptyset$; If $\bar{\mathcal{A}} = \emptyset$, but $m \geq (\kappa(n+1)+1)$, then $\mathcal{R} \neq \emptyset$.*

Proof of Lemma 5.3.3: For the case of $\bar{\mathcal{A}} \neq \emptyset$, recall equation (5.5) that $\bar{\mathcal{A}} \subset \mathcal{A}_j$. Then for any $j = 1, 2, ..., r$, one has

$$\mathcal{H}(x_{\bar{\mathcal{A}}}) \subset \mathcal{H}(x_{\mathcal{A}_j}).$$

It follows that

$$\mathcal{H}(x_{\bar{\mathcal{A}}}) \subset \bigcap_{j=1}^{r} \mathcal{H}(x_{\mathcal{A}_j}) = \mathcal{R}$$

Thus, $\bar{\mathcal{A}} \neq \emptyset$ leads to $\mathcal{R} \neq \emptyset$.

For the case of $\bar{\mathcal{A}} = \emptyset$, if $m \geq (\kappa(n+1)+1)$, one has from the Tverberg Theorem that $\mathcal{T} \neq \emptyset$. Thus, Lemma 5.3.2 leads to

$$\mathcal{T} \subset \mathcal{R} \neq \emptyset$$

This completes the proof. ■

5.3.2 A Low-Complexity Algorithm to Calculate $\mathcal{R}$

Since any point in $\mathcal{R}$ is a resilient convex combination (by Lemma 5.3.1), it is desirable to calculate the set $\mathcal{R}$, which by (5.4) is the intersection of a group of convex hulls. Existing approaches for the computation of intersection of convex hulls are usually computationally complex (#p-hard in [140]). Thus, in this section, we will develop an algorithm with low computational complexity for calculating a point in $\mathcal{R}$.

First, we will propose an equivalent expression of the set $\mathcal{R}$ in terms of equality and inequality **constraints**. For each $\mathcal{A}_j = \{j_1, j_2, \cdots, j_p\}$, $j = 1, 2, ..., r$, we define the following matrix

$$Y_j = \begin{bmatrix} x_{j_1}(t) & x_{j_2}(t) & \cdots & x_{j_p}(t) \end{bmatrix} \in \mathbb{R}^{n \times p}. \tag{5.7}$$

We call

$$X = \text{diag}\{Y_j,\ j = 1, 2, ..., r\} \in \mathbb{R}^{nr \times pr} \tag{5.8}$$

the *coordinate matrix*. For example, suppose $\mathcal{A} = \{1,\ 2,\ 3\}$, $\bar{\mathcal{A}} = \{1\}$ and $\kappa = 1$, then $p = 2$, $r = 2$ and one has:

$$\begin{aligned} &\mathcal{A}_1 = \{1, 2\}, \quad \mathcal{A}_2 = \{1, 3\} \\ &Y_1 = \begin{bmatrix} x_1 & x_2 \end{bmatrix}, \quad Y_2 = \begin{bmatrix} x_1 & x_3 \end{bmatrix} \\ &X = \begin{bmatrix} Y_1 & 0 \\ 0 & Y_2 \end{bmatrix}. \end{aligned}$$

This coordinate matrix allows us to characterize the set $\mathcal{R}$ with the following lemma

Lemma 5.3.4 *Let $C \in \mathbb{R}^{r \times r}$ be the circulant matrix with the first row in the form of $\begin{bmatrix} 1 & -1 & 0 & \cdots & 0 \end{bmatrix}$. Then*

$$\mathcal{R} = \left\{ \frac{1}{r} (\mathbf{1}_r' \otimes I_n) X \boldsymbol{\beta} \right\}, \tag{5.9}$$

for all $\boldsymbol{\beta} \in \mathbb{R}^{pr}$ that satisfies

$$(C \otimes I_n) X \boldsymbol{\beta} = 0 \tag{5.10}$$

$$(I_r \otimes \mathbf{1}_p') \boldsymbol{\beta} = \mathbf{1}_r \tag{5.11}$$

$$\boldsymbol{\beta} \geq 0. \tag{5.12}$$

Before proving Lemma 5.3.4, note that if we let $\boldsymbol{\beta} = \text{col}\ \{\beta_j,\ j = 1, 2, ..., r\}$ and $y_j = Y_j \beta_j \in \mathbb{R}^n$, with $\beta_j \in \mathbb{R}^p$, then

$$X\boldsymbol{\beta} = \text{col}\ \{y_j,\ j = 1, 2, ..., r\}. \tag{5.13}$$

This indicates that the components y_j of $X\boldsymbol{\beta}$ are linear combinations of vectors stored in Y_j according to the coefficient vector β_j. With this, we carry out the following proof.

Proof of Lemma 5.3.4: From (5.13), the definition of the circulant matrix C, and $(C \otimes I_n) X \boldsymbol{\beta} = 0$ in (5.10), one has for all $j = 1, 2, ..., r$, there exists a y^* such that

$$y_j = y^*. \tag{5.14}$$

Let $\mathcal{Y}_j$ denote the set of y_j for all $\boldsymbol{\beta}$ satisfying (5.11)-(5.12). Note that these equations ensure β_j is a nonnegative vector with entries summing to 1, which guarantees the combinations $y_j = Y_j\beta_j$ are convex. Then, given the definitions (5.2) and (5.7), it is true that

$$\mathcal{Y}_j = \mathcal{H}(x_{\mathcal{A}_j}), \quad j = 1, 2, ..., r. \tag{5.15}$$

This along with (5.14) indicate that the set of all feasible y^* is given by

$$\{y^*\} = \bigcap_{j=1}^{r} \mathcal{Y}_j = \bigcap_{j=1}^{r} \mathcal{H}(x_{\mathcal{A}_j}) = \mathcal{R}. \tag{5.16}$$

Recall that $(\mathbf{1}_r' \otimes I_n)X\boldsymbol{\beta} = y_1 + y_2 + ... + y_r = ry^*$. Thus,

$$\left\{\frac{1}{r}(\mathbf{1}_r' \otimes I_n)X\boldsymbol{\beta}\right\} = \mathcal{R}.$$

This completes the proof. ■

Lemma 5.3.4 tells us that computing the set $\mathcal{R}$ is equivalent to solving equations (5.9)-(5.12). However, to obtain a particular resilient convex combination, one does not necessarily have to find all points in this set. Here, we consider a point u, which tends to equally use all vectors in $x_{\mathcal{A}}$, namely

$$u = \frac{1}{r}(\mathbf{1}_r' \otimes I_n)X\boldsymbol{\beta}^* \tag{5.17}$$

where $\boldsymbol{\beta}^*$ is computed by the following quadratic programming problem:

$$\text{minimize} \quad J(\boldsymbol{\beta}) = \frac{1}{p}\|\boldsymbol{\beta} - \frac{1}{p}\mathbf{1}_{pr}\|_2^2 \tag{5.18}$$

subject to constraints (5.10)-(5.12). Specially when $\kappa = 0$, one has $r = 1$, $p = m$. Then $\boldsymbol{\beta^*} = \frac{1}{m}\mathbf{1}_m$, and $u = \frac{1}{m}\sum_{j=1}^{m} x_j$, which is the average of all vectors in $x_{\mathcal{A}}$.

To reveal the mechanism of (5.18), note that

$$\frac{1}{p}\|\boldsymbol{\beta} - \frac{1}{p}\mathbf{1}_{pr}\|_2^2 = \frac{1}{p}\sum_{j=1}^{r}\|\beta_j - \frac{1}{p}\mathbf{1}_r\|_2^2$$

which minimizes the sum of variances of coefficient vectors β_j. Recall that since the entries of β_j sum up to 1, its average should be $\frac{1}{p}$ so when the value of $J(\boldsymbol{\beta})$ approaches

0, the weights are equally distributed to all states. In this way, the u in (5.17) can be viewed as an unbiased choice of the resilient convex combination that lies in the region $\mathcal{R}$. It is worth mentioning that the complexity of achieving the $\boldsymbol{\beta}^*$ in (5.18) is $\mathcal{O}(n \cdot (mr)^3)$ [141].

5.3.3 Main Result

The main result of the paper is the following theorem, which summarizes Lemmas 5.3.1-5.3.4:

Theorem 5.3.1 *Consider a set $x_{\mathcal{A}} = \{x_1, x_2, \cdots, x_m\}$ of m vectors in $\mathbb{R}^n$, where $\mathcal{A} = \{1, 2, ..., m\}$. Suppose at most κ vectors in $x_{\mathcal{A}}$ are malicious and $\bar{\mathcal{A}}$ is a known label set that only contains normal vectors, which can also be empty. If $\bar{\mathcal{A}} \neq \emptyset$ or $m \geq (\kappa(n+1)+1)$, any point in the non-empty set $\mathcal{R}$ defined in* (5.4) *is a resilient convex combination. One specific point $u \in \mathcal{R}$ defined in* (5.17) *can be computed by solving the quadratic programming problem* (5.18) *subject to constraints* (5.10)-(5.12).

Recall that the set of Tverberg points $\mathcal{T}$, which is a subset of $\mathcal{R}$, also provides a region for choosing resilient convex combinations. Comparisons between $\mathcal{T}$ and $\mathcal{R}$ are provided as follows:

- First, determining Tverberg points requires high computational complexity. Although the Tverberg Theorem provides a sufficient condition for the existence of a partition leading to Tverberg point, the theorem does not provide an algorithm for finding the partition for achieving Tverberg points, apart from enumerating all possible partitions and checking the intersection of convex hulls for each partition. As mentioned in [57], except for some specific values of n, the computational complexity of achieving Tverberg points is $\mathcal{O}(m^{n-1})$ [57], which grows exponentially with $n-1$, where n is the dimension. In contrast, it has been shown that the resilient convex combination proposed in (5.17) can be computed by solving a standard quadratic programming problem, whose computational complexity is polynomial in n.

- Second, the existence of an non-empty Tverberg point set $\mathcal{T}$ requires that $m \geq \kappa(n+1)+1$, while one has $\mathcal{R} \neq \emptyset$ if $\bar{\mathcal{A}} \neq \emptyset$ or $m \geq \kappa(n+1)+1$. In achieving resilience for distributed algorithms, one aims to guarantee all normal agents' states to converge to a consensus. Then each normal agent at least has one element (which is itself) in $\bar{\mathcal{A}}$. Then the existence of an non-empty $\mathcal{R}$ is automatically guaranteed. Please refer to Fig. 5.1. (A) and (B) for an example when $\mathcal{R}$ is non-empty while $\mathcal{T} = \emptyset$, and an example $\mathcal{T} \subset \mathcal{R}$, respectively.

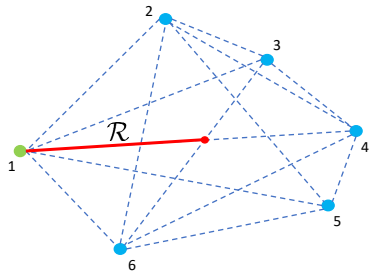

Figure 5.1. Finding Tverberg point $\mathcal{T}$ (yellow) in a 2-D space, with $\bar{\mathcal{A}} = \{1\}$. $[\kappa = 2,\ m = 6,\ m < (\kappa(n+1)+1)]$

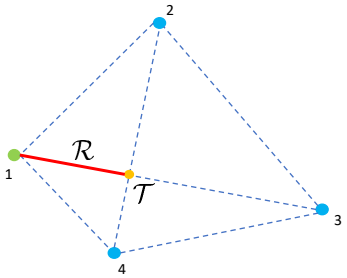

Figure 5.2. Finding $\mathcal{R}$ (red) in a 2-D space, with $\bar{\mathcal{A}} = \{1\}$. $[\kappa = 1,\ m = 4,\ m = (\kappa(n+1)+1)]$

5.4 Application of the Resilient Convex Combination into Consensus-Based Distributed Algorithms

Consider a network of $\bar{m}$ agents in which each agent i is able to sense or receive information from certain other nearby agents, termed agent i's neighbors. We suppose agent i is always a neighbor of itself and we let $\mathcal{N}_i(t)$ denote the set of agent i's neighbors at time t, $i = 1, 2, \cdots, \bar{m}$. The neighbor relations can be described by a time-dependent graph $\mathbb{G}(t)$ such that there is a directed edge from j to i in $\mathbb{G}(t)$ if and only if $j \in \mathcal{N}_i(t)$. Suppose each agent i controls a state vector $x_i(t) \in \mathbb{R}^n$. Consensus-based distributed algorithms have been proposed in the literature that solve problems that are unconstrained, constrained by linear or nonlinear constraints [12], and/or minimize a global objective function [37]. These algorithms share a common form

$$x_i(t+1) = f_i(x_i(t), v_i(t)) \tag{5.19}$$

where $v_i(t)$ is a convex combination of all agent i's neighbors' states, that is,

$$v_i(t) = \sum_{j \in \mathcal{N}_i(t)} w_{ij}(t) x_j(t)$$

with $\sum_{j \in \mathcal{N}_i} w_{ij}(t) = 1$.

Since each agent updates its state by a convex combination of all its neighbors' states, when one or more neighbors are malicious, the convex combination also contains false information, which may prevent the overall consensus goal from being reached. This motivates the key idea to replace the convex combination $v_i(t)$ with a resilient convex combination $u_i(t)$ defined in (5.17). To be more specific, in each time step, we assume that agent i knows $\mathcal{A} = \mathcal{N}_i(t)$, $\bar{\mathcal{A}} = \{i\}$ and the upper bound of agent's malicious neighbors κ. Then it computes $u_i(t)$ by solving the quadratic programming problem (5.18) under constraints (5.10)-(5.12). In this way, the malicious information is automatically isolated by $u_i(t)$.

5.5 Validation

In this section, we provide simulations for an 11-agent time-varying network consisting of both directed and undirected edges as indicated in Fig. 5.3, in which agent 10 and 11 are malicious agents and connect themselves to different normal agents as time evolves. By replacing $v_i(t)$ in (5.19) with $u_i(t)$, the problem of interest is to check whether all normal agents from 1 to 9 under this update still reach the desired consensus in the presence of malicious agents. In the following examples, we suppose each $x_i(t) \in \mathbb{R}^2$, $\kappa = 1$ (even though there are two malicious agents, for each agent, the upper bound of malicious neighbor is 1). Each malicious agent sends a state to its neighbors which is randomly chosen from the set $[0, 2] \times [0, 2]$.

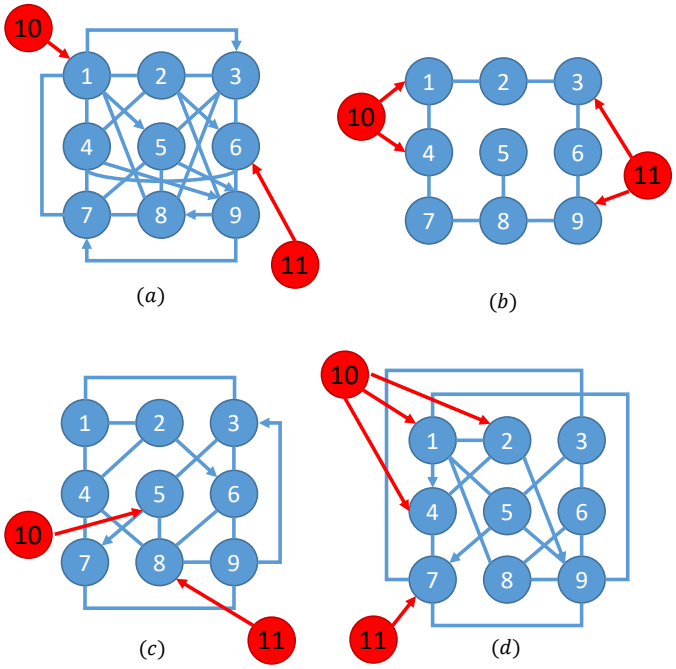

Figure 5.3. A network of 11 agents with malicious agents marked in red.

Example 1 (Unconstrained Consensus).

We first consider the unconstrained consensus problem, in which all $x_i(t) \in \mathbb{R}^2$, $i = 1, 2, \cdots, 9$, aim to reach consensus by the following update

$$x_i(t+1) = \frac{1}{d_i(t)} \sum_{j \in \mathcal{N}_i(t)} x_i(t) \tag{5.20}$$

with initialization $x_i(0)$ randomly chosen from the areas of $[0, 2] \times [0, 2]$. Let

$$V(t) = \frac{1}{2} \sum_{i=1}^{9} \|x_i(t) - x_{i+1}(t)\|_2^2 \tag{5.21}$$

which measures the closeness of all normal agents' states to a consensus.

A. Under Fixed Graph

Suppose the network is a fixed one as in Fig. 5.3-(a). Simulation results for the consensus update (5.20) with and without the presence of malicious agents are shown in Fig. 5.4, which indicates unsurprisingly that the traditional consensus update (5.20) could easily fail in the presence of malicious agents.

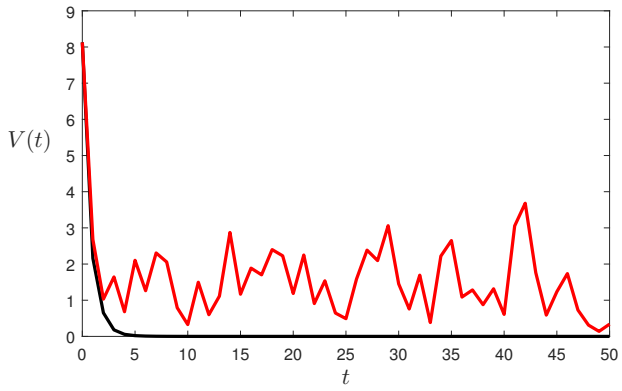

Figure 5.4. Simulations of normal agents under the consensus update (5.20) without malicious agents (blank line) and with malicious agents 10 and 11 (red line).

By introducing a new resilient convex combination $u_i(t)$ at each agent i, which is a convex combination of normal agents, one could employ the following update

$$x_i(t+1) = u_i(t) \tag{5.22}$$

where $u_i(t)$ could be chosen as Tverberg points or as the resilient convex combination in (5.17). Consensus is reached in the presence of malicious agents in both cases as shown by simulations in Fig. 5.5. It is also worth mentioning that using u_i defined in (5.17) to replace the original convex combination $v_i(t)$ could lead to faster convergence than using Tverberg points as also indicated in Fig. 5.5

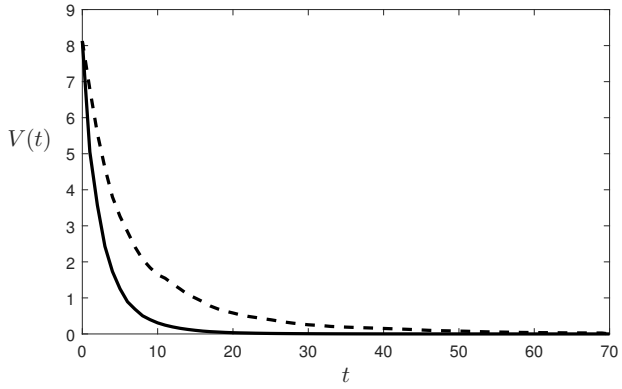

Figure 5.5. Consensus is reached by introducing $u_i(t)$ as Tverberg points (indicated by the dashed line) or as the resilient convex combination (5.17) (indicated by the solid line).

B. Under Time-varying Graph

We perform simulations of unconstrained consensus on a periodic sequence of time-varying networks as in Fig. 5.3. The method based on Tverberg point is not applicable here since the number of each agent's neighbors is not always greater than the condition required by the Tverberg Theorem. However, one could still reach consensus by introducing the resilient convex combination (5.17) into (5.22). Simulation results are shown in Fig. 5.6.

Example 2 (Constrained Consensus).

We consider the distributed algorithm for solving linear equations. Suppose each agent i knows

$$A_i x_i = b_i$$

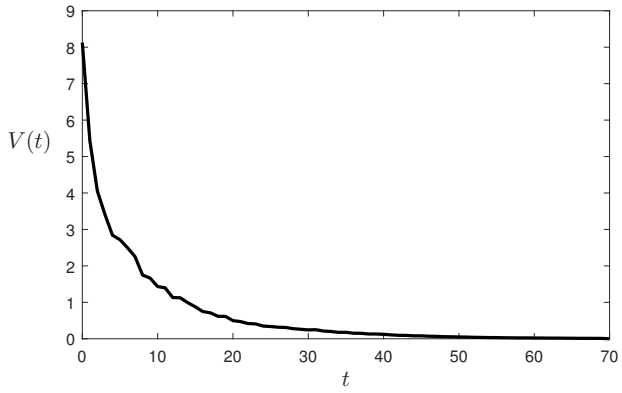

Figure 5.6. Simulations by using the resilient convex combination $u_i(t)$ of (5.17) into (5.22).

where

$$A_1 = A_2 = A_3 = \begin{bmatrix} 3 & -1 \end{bmatrix} \qquad b_1 = b_2 = b_3 = 2$$
$$A_4 = A_5 = A_6 = \begin{bmatrix} 0 & 1 \end{bmatrix} \qquad b_4 = b_5 = b_6 = 1$$
$$A_7 = A_8 = A_9 = \begin{bmatrix} -1 & 3 \end{bmatrix} \qquad b_7 = b_8 = b_9 = 2$$

and updates its state according to

$$x_i(t+1) = x_i(t) - P_i(x_i(t) - v_i(t)) \tag{5.23}$$

where P_i is the orthogonal projection on the kernel of A_i and $v_i(t) = \frac{1}{d_i(t)} \sum\limits_{j \in \mathcal{N}_i} x_i(t)$. Simulations are still performed on a periodic sequence of time-varying networks as in Fig. 5.3. Let $x^* = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}'$ denote a solution to $Ax = b$, which is also the desired consensus value. Let

$$V(t) = \frac{1}{2} \sum_{i=1}^{9} \|x_i(t) - x^*\|_2^2$$

which measures the closeness between all agents' states to x^*. Simulation results are as shown in Fig. 5.7 for the cases with and without malicious agents, respectively. The presence of malicious agents also disrupts the distributed algorithm (5.23) for solving linear equations. By using the resilient convex combination $u_i(t)$ of (5.17) at

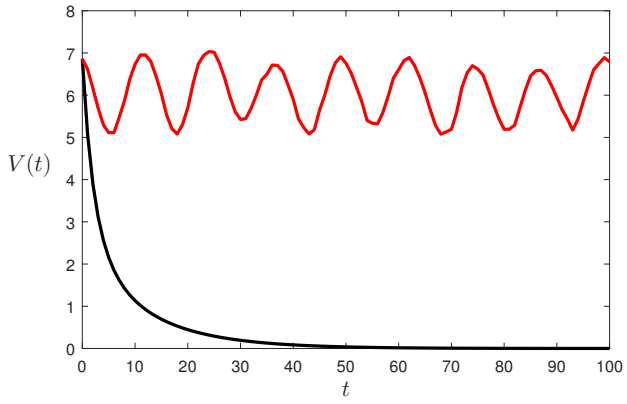

Figure 5.7. Simulation results under the update (5.23) with no malicious agents (indicated by the black line) or with malicious agents (indicated by the red line).

each agent i in (5.23), one still enables all agents to achieve x^* exponentially fast in the presence of malicious agents as shown in Fig. 5.8.

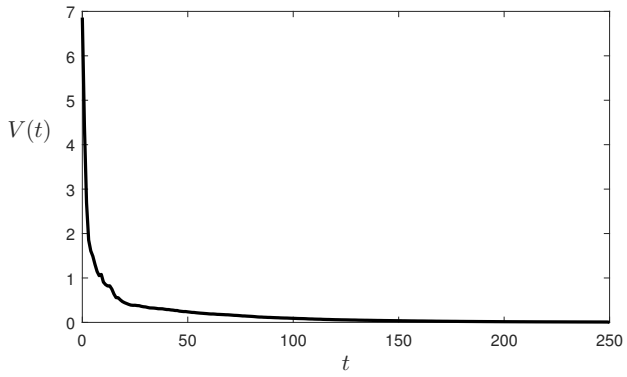

Figure 5.8. Simulations by using the resilient convex combination $u_i(t)$ of (5.17) in (5.23).

6. CONCLUSION REMARKS AND FUTURE DIRECTIONS

6.1 Conclusion Remarks

This dissertation introduces new distributed algorithms to address the distributed computation, distributed optimization and resilient distributed consensus in multi-agent systems.

- **Consensus-based distributed algorithms for solving linear equations.** In this chapter, we have further improved an existing distributed algorithm for solving linear equations (DALE) in three ways. First, for agents with limited computation capability, the improved DALE enables all agents to achieve exponentially fast a solution to the whole linear equations without any initialization step. Second, if the linear equation to be solved has more than one solution, with a modified initialization step, the improved algorithm can find the solution with minimum l_2 norm. Third, by combining the projection-consensus and sub-gradient descent aspects, the improved algorithm can find the solution with minimum l_1 norm, if the linear equation to be solved has more than one solution.

- **Scalable, Distributed Algorithms for Solving Linear Equations via Double-Layered Networks.** This chapter has devised distributed algorithms in a double-layered multi-agent framework for solving linear equations, which consists of clusters and each cluster is composed of an aggregator and a network of agents. In these distributed algorithms, each agent is not required to know as much as a complete row or column of the overall linear equation. Both analytical proof and simulation results are provided to validate exponential convergence.

- **Consensus-based Distributed Optimization for Multi-agent Systems** The first part of this chapter has proposed a distributed algorithm for multi-agent networks to achieve least squares solutions exponentially fast when the networks are undirected and connected. Although the proposed algorithm is in discrete-time, its exponential convergence does not rely on any time-varying or small step-size. In the second part of this chapter, by incorporating the idea of integral feedback, we proposed a continuous-time distributed algorithm which is able to solve a constrained distributed optimization problem. This algorithm does not impose an additional burden on the communication bandwidth, ensures exponentially fast convergence, and offers robustness against disturbance.

- **A Resilient Convex Combination for consensus-based distributed algorithms.** In this chapter, given a set of vectors that includes both normal and malicious information, this paper has proposed a way to determine a resilient convex combination by using the intersection of convex hulls. By formulating the set of such combinations as linear constraints, a vector inside this set can be computed by quadratic programming. It has been shown that the obtained resilient convex combination can isolate harmful state information injected by cyber-attacks. In addition, since no identification process is required, the method has promise for dealing with time-varying attacks for consensus-based distributed algorithms, as shown by simulations.

6.2 Future Directions

Originated from my Ph.D. research progress, my future research goal mainly focuses on bridging the gap between the theoretical advances and engineering practice of multi-agent control, and further applies these techniques in AI-based machine learning/data driven control applications, achieving network resilience under sophisticated cyber-attacks, and studying the Human in loop for multi-agent systems. Towards

these ends, the major issues that I identify and seek to address are summarized as follows.

6.2.1 Distributed Data-driven Multi-agent Learning and Control

The field of control and autonomy is evolving from model-based control to non-model based autonomy, where the key techniques to fill the gap are machine learning and data-driven control. Modern machine learning algorithms are challenged by both model complexity and training efficiency. This has motivated many research teams and companies, including *Google-DistBelief, Microsoft-DMTK, Apache-Spark, OpenAI-ADAM, and Tencent-Angel*, to develop distributed machine learning algorithms, which allow the modes being cooperatively trained by multiple computational agents within a distributed network. Here, we want to tackle this problem from the aspect of distributed optimization as it shares a same mathematic nature with machine learning and data-driven control. i.e., the local objection function in optimization problems, can be replaced by the risk function for distributed learning or the reward function for data-driven control.

Challenges: One of the challenges in distributed data-driven multi-agent learning and control is to effectively synchronize the large parameter set of learning models in large-scale multi-agent systems. In addition, for learning based algorithms, one usually uses a sampled gradient, which is usually computed from one or a batch of samples, to estimate the true gradient, so the second challenge is how to handle the uncertainty created during such sampling process.

Payoffs: By incorporating multi-agent systems to data-driven control and learning, key benefits one can expect are a more abundant distributed data set for training and better learning efficiency. This is achieved by assigning the overall computation/data storage load to different agents.

6.2.2 The Cyber-Security of Autonomous Multi-agent Systems

Cyber-attack resilience is a fundamental demand for multi-agent systems, especially with applications in large-scale autonomous multi-agent swarms. Although the existing paradigms of data encryption, access control, and fault-tolerant control can provide certain level resilience, these approaches will become ineffective as the attacking methods are becoming more difficult to identify and can impact a larger number of other normal agents. To further elevate the resilience of multi-agent systems, based on the *resilient consensus* technique we have developed in [59], the idea is to enhance the adaptation of this technique by incorporating learning-based features(AI), which allow the algorithm to learn from the dynamics of attackers and repeatedly improve its performance over time.

Challenges: First, sophisticated cyber-attacks are usually not constrained by geography. A malicious agent under such attacks could change its location over time and connect itself to different normal agents. Second, achieving resilience in a distributed scenario relies on greater connectivity and redundancy, which may be challenging to maintain in a dynamic network where agents are free to join and leave. Third, it is known that learning approaches rely critically on data availability. However, since the agents in multi-agent systems suffer limited sensing capability, the learning model only has partial access to the behaviors of a cyber-attack.

Payoffs: The proposed research will establish a resilient framework for multi-agent coordination, which is the building block for the development of a wide range of future distributed algorithms. Further, it has the potential to significantly impact certain engineering areas, where reliability is the main concern that blocks the deployment of multi-agent systems.

6.2.3 Human in the Loop for Multi-agent Systems

The current concept of multi-agent system is only defined as a group of heterogeneous autonomous agents. However, it is also quite intuitive that human can also

join the network and play some roles in the decision making of the overall systems. Motivated by this, one of my future research direction aims to study:

- What are the roles a human can play in a multi-agent system.
- How Human-machine interaction can improve the performance and intelligence of multi-agent systems.

Challenges: One key challenge of this project is the interface between the human and the machine. Secondly, it is difficult to find a measure that can simultaneously parameterize the behaviors of both humans and machines. Finally, the impact of the human to the system should greatly depend on the network topology of the multi-agent system, which is difficult to quantify and analyze.

Payoffs: The result of this research will build the fundamental of the methodologies for the collaboration of humasn and machines under a same environment, allowing humans to use their high level knowledge to assist the connected autonomous agents in completing highly sophistic tasks.

REFERENCES

[1] F. Bullo, J. Cortes, and S. Martinez. *Distributed Control of Robotic Networks*. Princeton University Press, 2009.

[2] Ming Cao, A Stephen Morse, and Brian DO Anderson. Reaching a consensus in a dynamically changing environment: A graphical approach. *SIAM Journal on Control and Optimization*, 47(2):575–600, 2008.

[3] Xudong Chen, Mohamed-Ali Belabbas, and Tamer Başar. Controllability of formations over directed time-varying graphs. *IEEE Transactions on Control of Network Systems*, 4(3):407–416, 2015.

[4] Chuan Yan and Huazhen Fang. A new encounter between leader–follower tracking and observer-based control: Towards enhancing robustness against disturbances. *Systems & Control Letters*, 129:1–9, 2019.

[5] M. Mesbahi and M. Egerstedt. *Graph Theoretic Methods in Multi-Agent Networks*. Princeton University Press, 2010.

[6] X. Wang, S. Mou, and B. D. O. Anderson. Scalable, distributed algorithms for solving linear equations via double-layered networks. *IEEE Transactions on Automatic Control*, 65(3):1132–1143, 2020.

[7] J. A. Fax and R.M. Murray. Information flow and cooperative control of vehicle formations. *IEEE Transactions on Automatic Control*, 49(9):1465–1476, 2004.

[8] S. Kar, J. M. F. Moura, and K. Ramanan. Distributed parameter estimation in sensor networks: Nonlinear observations models and imperfect communication. *IEEE Transactions on Information Theory*, 58(6):1–52, 2012.

[9] 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.

[10] D. H. Lee, S. A. Zaheer, and J. H. Kim. Ad hoc network-based task allocation with resource-aware cost generation for multirobot systems. *IEEE Transactions on Industrial Electronics*, 61(12):6871–6881, Dec 2014.

[11] F. Dorfler, M. Chertkov, and F. Bullo. Synchronization in complex oscillator networks and smart grids. *Proceedings of the National Academy of Sciences*, 110(6):2005–2010, 2013.

[12] 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.

[13] B. D. O. Anderson, S. Mou, U. R. Helmke, and A. S. Morse. Decentralized gradient algorithm for solution of a linear equation. *Numerical Algebra, Control and Optimization*, 6(3):319–328, 2016.

[14] Jie Lu and Choon Yik Tang. A distributed algorithm for solving positive definite linear equations over networks with membership dynamics. *IEEE Transactions on Control of Network Systems*, 5(1):215–227, 2018.

[15] J. Wang and N. Elia. Distributed solution of linear equations over unreliable networks. *Proceedings of American Control Conference*, pages 6471–6476, July 2016.

[16] Xuan Wang, Shaoshuai Mou, and Dengfeng Sun. Improvement of a distributed algorithm for solving linear equations. *IEEE Transactions on Industrial Electronics*, 64(4):3113–3117, 2017.

[17] S. Mou, Z. Lin, L. Wang, D. Fullmer, and A. S. Morse. A distributed algorithm for efficiently solving linear equations and its applications (special issue jcw). *Systems & Control Letters*, 91:21–27, 2016.

[18] Shaoshuai Mou and Brian D. O. Anderson. Eigenvalue invariance of inhomogeneous matrix products in distributed algorithms. *IEEE control systems letters*, 1(1):8–13, 2017.

[19] J. Liu, A. S. Morse, A. Nedić, and T. Başar. Exponential convergence of a distributed algorithm for solving linear algebraic equations. *Automatica*, 83:37–46, 2017.

[20] S Sh Alaviani and Nicola Elia. A distributed algorithm for solving linear algebraic equations over random networks. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 83–88. IEEE, 2018.

[21] J. Liu, S. Mou, and A. S. Morse. Asynchronous distributed algorithms for solving linear algebraic equations. *IEEE Transactions on Automatic Control*, 63(2):372–385, 2018.

[22] Wenlong Shen, Bo Yin, Xianghui Cao, Yu Cheng, and Xuemin Sherman Shen. A distributed secure outsourcing scheme for solving linear algebraic equations in ad hoc clouds. *IEEE Transactions on Cloud Computing*, 2019.

[23] Qingchen Liu, Yang Liu, Deming Yuan, and Jiahu Qin. Distributedly solving network linear equations with event-based algorithms. *IET Control Theory & Applications*, 2019.

[24] Bo Yin, Wenlong Shen, Xianghui Cao, Yu Cheng, and Qing Li. Securely solving linear algebraic equations in a distributed framework enhanced with communication-efficient algorithms. *IEEE Transactions on Network Science and Engineering*, 2019.

[25] J. Zhou, X. Wang, S. Mou, and B. D. O. Anderson. Finite-time distributed linear equation solver for solutions with minimum l_1-norm. *IEEE Transactions on Automatic Control*, 65(4):1691–1696, 2020.

[26] Peng Wang, Wei Ren, and Zhisheng Duan. Distributed algorithm to solve a system of linear equations with unique or multiple solutions from arbitrary initializations. *IEEE Transactions on Control of Network Systems*, 6(1):82–93, 2019.

[27] Jie Lu and Choon Yik Tang. A distributed algorithm for solving positive definite linear equations over networks with membership dynamics. *IEEE Transactions on Control of Network Systems*, 5(1):215–227, 2018.

[28] Navid Azizan-Ruhi, Farshad Lahouti, Amir Salman Avestimehr, and Babak Hassibi. Distributed solution of large-scale linear systems via accelerated projection-based consensus. *IEEE Transactions on Signal Processing*, 67(14):3806–3817, 2019.

[29] Qianqian Cai, Zhaorong Zhang, and Minyue Fu. A fast converging distributed solver for linear systems with generalised diagonal dominance. *arXiv preprint arXiv:1904.12313*, 2019.

[30] Xianlin Zeng, Shu Liang, Yiguang Hong, and Jie Chen. Distributed computation of linear matrix equations: An optimization perspective. *IEEE Transactions on Automatic Control*, 64(5):1858–1873, 2019.

[31] Xuan Wang and Shaoshuai Mou. A distributed algorithm for achieving the conservation principle. In *2018 Annual American Control Conference (ACC)*, pages 5863–5867, 2018.

[32] F. Dorfler and F. Bullo. Synchronization in complex networks of phase oscillators: a survey. *Automatica*, 50(6):1539–1564, 2014.

[33] R. Olfati-Saber. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Transactions on Automatic Control*, 51(3):401–420, 2006.

[34] J. Cortes, S. Martinez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation*, 20(2):243–255, 2004.

[35] Naman Agarwal, Ananda Theertha Suresh, Felix Yu, Sanjiv Kumar, and Brendan McMahan. cpsgd: Communication-efficient and differentially-private distributed sgd. In *Advances in Neural Information Processing Systems*, pages 7575–7586, 2018.

[36] Ryan McDonald, Keith Hall, and Gideon Mann. Distributed training strategies for the structured perceptron. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 456–464. Association for Computational Linguistics, 2010.

[37] Angelia Nedic, Asuman Ozdaglar, and Pablo A Parrilo. Constrained consensus and optimization in multi-agent networks. *IEEE Transactions on Automatic Control*, 55(4):922–938, 2010.

[38] Peng Lin, Wei Ren, and Yongduan Song. Distributed multi-agent optimization subject to nonidentical constraints and communication delays. *Automatica*, 65:120–131, 2016.

[39] Angelia Nedic and Asuman Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.

[40] Guodong Shi, Brian D. O. Anderson, and Uwe Helmke. Network flows that solve linear equations. *IEEE Transactions on Automatic Control*, 62(6):2659–2674, 2017.

[41] Bahman Gharesifard and Jorge Cortés. Distributed continuous-time convex optimization on weight-balanced digraphs. *IEEE Transactions on Automatic Control*, 59(3):781–786, 2014.

[42] Wei Shi, Qing Ling, Gang Wu, and Wotao Yin. Extra: An exact first-order algorithm for decentralized consensus optimization. *SIAM Journal on Optimization*, 25(2):944–966, 2015.

[43] Ying Sun, Amir Daneshmand, and Gesualdo Scutari. Convergence rate of distributed optimization algorithms based on gradient tracking. *arXiv preprint arXiv:1905.02637*, 2019.

[44] Zhirong Qiu, Shuai Liu, and Lihua Xie. Distributed constrained optimal consensus of multi-agent systems. *Automatica*, 68:209–215, 2016.

[45] Xianlin Zeng, Peng Yi, and Yiguang Hong. Distributed continuous-time algorithm for constrained convex optimizations via nonsmooth analysis approach. *IEEE Transactions on Automatic Control*, 62(10):5227–5233, 2016.

[46] Qingshan Liu and Jun Wang. A second-order multi-agent network for bound-constrained distributed optimization. *IEEE Transactions on Automatic Control*, 60(12):3310–3315, 2015.

[47] Shreyas Sundaram and Bahman Gharesifard. Distributed optimization under adversarial nodes. *IEEE Transactions on Automatic Control*, 2018. DOI: 10.1109/TAC.2018.2836919.

[48] Martin E Liggins, Chee-Yee Chong, Ivan Kadar, Mark G Alford, Vincent Vannicola, and Stelios Thomopoulos. Distributed fusion architectures and algorithms for target tracking. *Proceedings of the IEEE*, 85(1):95–107, 1997.

[49] Michael Ben-Or. Another advantage of free choice (extended abstract): Completely asynchronous agreement protocols. In *Proceedings of the second annual ACM symposium on Principles of distributed computing*, pages 27–30. ACM, 1983.

[50] Gabriel Bracha. An asynchronous $[(n-1)/3]$-resilient consensus protocol. In *Proceedings of the third annual ACM symposium on Principles of distributed computing*, pages 154–162. ACM, 1984.

[51] Seyed Mehran Dibaji and Hideaki Ishii. Resilient consensus of second-order agent networks: Asynchronous update rules with delays. *Automatica*, 81:123–132, 2017.

[52] Peva Blanchard, Rachid Guerraoui, Julien Stainer, et al. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems*, pages 119–129, 2017.

[53] H. J. LeBlane, H.Zhang, X.Koutsoukos, and S.Sundaram. Resilient asymptotic consensus in robust networks. *IEEE Journal on Selected Areas in Communications*, 31(4):766–781, 2013.

[54] H. Mendes, M. Herlihy, N. Vaidya, and V.K. Garg. Multidimensional agreement in byzantine systems. *Distributed Computing*, 28(6):423–441, 2015.

[55] Nitin H Vaidya. Iterative byzantine vector consensus in incomplete graphs. In *International Conference on Distributed Computing and Networking*, pages 14–28. Springer, 2014.

[56] Hyongju Park and Seth A Hutchinson. Fault-tolerant rendezvous of multirobot systems. *IEEE transactions on robotics*, 33(3):565–582, 2017.

[57] W. Mulzer and D. Werner. Approximating tverberg points in linear time for any fixed dimension. *Discrete & Computational Geometry*, 50(2):520–535, 2013.

[58] L. Tseng and N. H. Vaidya. Asynchronous convex hull consensus in the presence of crash faults. In *ACM symposium on Principles of distributed computing*, pages 396–405, 2014.

[59] Xuan Wang, Shaoshuai Mou, and Shreyas Sundaram. A resilient convex combination for consensus-based distributed algorithms. *Numerical Algebra, Control & Optimization*, 9(3):269–281, 2019.

[60] Jingqiu Zhou, Xuan Wang, Shaoshuai Mou, and Brian DO Anderson. Distributed algorithm for achieving minimum l_1 norm solutions of linear equation. In *2018 Annual American Control Conference (ACC)*, pages 5857–5862, 2018.

[61] Xuan Wang, Shaoshuai Mou, and Dengfeng Sun. Further discussions on a distributed algorithm for solving linear algebra equations. In *2017 American Control Conference (ACC)*, pages 4274–4278, 2017.

[62] Xuan Wang, Jingqiu Zhou, Shaoshuai Mou, and Martin J Corless. A distributed algorithm for least squares solutions. *IEEE Transactions on Automatic Control*, 64(10):4217–4222, 2019.

[63] Xuan Wang, Jingqiu Zhou, Shaoshuai Mou, and Martin J Corless. A distributed linear equation solver for least square solutions. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 5955–5960, 2017.

[64] Xuan Wang, Shaoshuai Mou, and Shreyas Sundaram. Towards resilience for consensus-based multi-agent learning/planning. In *57th Annual Allerton Conference on Communication, Control, and Computing*, Sep. 2019.

[65] Xuan Wang, Shaoshuai Mou, and Brian DO Anderson. A discrete-time distributed algorithm for minimum l_1-norm solution of an under-determined linear equation set. In *21st IFAC World Congress*, Jul. 2020. Accepted.

[66] Ori Shental, Paul H. Siegel, Jack K. Wolf, Danny Bickson, and Danny Dolev. Gaussian Belief Propagation Solver for Systems of Linear Equations. *arXiv:0810.1736 [cs, math]*, pages 1863–1867, July 2008.

[67] Christina E. Lee, Asuman Ozdaglar, and Devavrat Shah. Asynchronous Approximation of a Single Component of the Solution to a Linear System. *arXiv:1411.2647 [cs]*, November 2014.

[68] M. F. Iacchetti, R. Perini, M. S. Carmeli, F. Castelli-Dezza, and N. Bressan. Numerical integration of odes in real-time systems like state observers: Stability aspects. *IEEE Transactions on Industry Applications*, 48(1):132–141, Jan 2012.

[69] C. O. Martinez and V. Puig. Piece-wise linear functions-based model predictive control of large-scale sewage systems. *IET Control Theory Applications*, 4:1581–1593, 2010.

[70] Peter M Shearer. Improving local earthquake locations using the l1 norm and waveform cross correlation: Application to the whittier narrows, california, aftershock sequence. *Journal of Geophysical Research: Solid Earth*, 102(B4):8269–8283, 1997.

[71] Yadolah Dodge. *Statistical data analysis based on the L1-norm and related methods*. Birkhäuser, 2012.

[72] Roland Beucker and HA Schlitt. On minimal lp-norm solutions of the biomagnetic inverse problem. Technical report, Zentralinstitut für Angewandte Mathematik, 1996.

[73] H. H. Bauschke and J. M. Borwein. Dykstra's alternating projection algorithm for two sets. *Journal of Approximation Theory*, 79(3):418–443, 1994.

[74] Dongdong Ge, Xiaoye Jiang, and Yinyu Ye. A note on the complexity of l p minimization. *Mathematical programming*, 129(2):285–299, 2011.

[75] E. J. Candes and T. Tao. Decoding by linear programming. *IEEE Transactions on Information Theory*, 51(12):4203–4215, Dec 2005.

[76] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[77] Alejandro D Dominguez-Garcia and Christoforos N Hadjicostis. Distributed matrix scaling and application to average consensus in directed graphs. *IEEE Transactions on Automatic Control*, 58(3):667–681, 2013.

[78] Oskar Perron. Zur theorie der matrices. *Mathematische Annalen*, 64(2):248–263, 1907.

[79] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

[80] D. A. Schmidt, C. Shi, R. A. Berry, M. L. Honig, and W. Utschick. Distributed resource allocation schemes. *IEEE Signal Processing Magazine*, 26(5):53–63, 2009.

[81] J. Liu and H. Peng. Modeling and control of a power-split hybrid vehicle. *IEEE Transactions on Control Systems Technology*, 16(6):1242–1251, Nov 2008.

[82] M. Treiber and A. Kesting. *Traffic flow dynamics: data, models and simulation*. Springer Science & Business Media, 2012.

[83] M. D. Domenico, A. Solé-Ribalta, E. Cozzo, M. Kivelä, Y. Moreno, M. A. Porter, S. Gómez, and A. Arenas. Mathematical formulation of multilayer networks. *Physical Review X*, 3(4):041022, 2013.

[84] Stefano Boccaletti, Ginestra Bianconi, Regino Criado, Charo I Del Genio, Jesús Gómez-Gardenes, Miguel Romance, Irene Sendina-Nadal, Zhen Wang, and Massimiliano Zanin. The structure and dynamics of multilayer networks. *Physics Reports*, 544(1):1–122, 2014.

[85] R. Simmons, T. Smith, M. D. Dias, D. Goldberg, D. Hershberger, A. Stentz, and R. Zlot. A layered architecture for coordination of mobile robots. *Multi-Robot Systems: From Swarms to Intelligent Automata*, pages 103–112, 2002.

[86] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle. Layering as optimization decomposition: A mathematical theory of network architechtures. *Proceedings of the IEEE*, 95(1):255–312, 2007.

[87] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[88] S. Gomez, A. Diaz-Guilera, J. Gomez-Gardenes, C. J. Perez-Vicente, Y. Moreno, and A. Arenas. Diffusion dynamics on multiplex networks. *Physical review letters*, 110(2):028701, 2013.

[89] J. M. Jeanne and R. I. Wilson. Convergence, divergence, and reconvergence in a feedforward network improves neural speed and accuracy. *Neuron*, 88(5):1014–1026, 2015.

[90] K. I. Tsianos and M. G. Rabbat. Multiscale gossip for efficient decentralized averaging in wireless packet networks. *IEEE Transactions on Signal Processing*, 61(9):2137–2149, 2013.

[91] J. Chow and P. Kokotovic. Time scale modeling of sparse dynamic networks. *IEEE Transactions on Automatic Control*, 30(8):714–722, August 1985.

[92] I. C. Morărescu, S. Martin, A. Girard, and A. Muller-Gueudin. Coordination in Networks of Linear Impulsive Agents. *IEEE Transactions on Automatic Control*, 61(9):2402–2415, September 2016.

[93] Marcos Cesar Bragagnolo, Irinel-Constantin Morărescu, Jamal Daafouz, and Pierre Riedinger. Reset strategy for consensus in networks of clusters. *Automatica*, 65:53–63, March 2016.

[94] Emrah Bıyık and Murat Arcak. Area aggregation and time-scale modeling for sparse nonlinear networks. *Systems & Control Letters*, 57(2):142–149, February 2008.

[95] D. Romeres, F. Dörfler, and F. Bullo. Novel results on slow coherency in consensus and power networks. In *2013 European Control Conference (ECC)*, pages 742–747, July 2013.

[96] Aihua Hu, Jinde Cao, Manfeng Hu, and Liuxiao Guo. Cluster synchronization in directed networks of non-identical systems with noises via random pinning control. *Physica A: Statistical Mechanics and its Applications*, 395:537–548, February 2014.

[97] Yangling Wang and Jinde Cao. Cluster synchronization in nonlinearly coupled delayed networks of non-identical dynamic systems. *Nonlinear Analysis: Real World Applications*, 14(1):842–851, February 2013.

[98] W. He, B. Zhang, Q. L. Han, F. Qian, J. Kurths, and J. Cao. Leader-Following Consensus of Nonlinear Multiagent Systems With Stochastic Sampling. *IEEE Transactions on Cybernetics*, 47(2):327–338, February 2017.

[99] G. Mao, Z. Zhang, and B. D. O. Anderson. Cooperative content dissemination and offloading in heterogeneous mobile networks. *IEEE Transactions on Vehicular Technology*, 65(8):6573–6587, 2016.

[100] M. Hautus H. Trentelman, A. A. Stoorvogel. *Control theory for linear systems*. Springer Science & Business Media, 2012.

[101] X. Wang, S. Mou, and B. D. O. Anderson. Consensus-based distributed optimization enhanced by integral feedback. *IEEE Transactions on Automatic Control*, 2020. Submitted.

[102] Guannan Qu and Na Li. Harnessing smoothness to accelerate distributed optimization. *IEEE Transactions on Control of Network Systems*, 5(3):1245–1260, 2017.

[103] Tao Yang, Xinlei Yi, Junfeng Wu, Ye Yuan, Di Wu, Ziyang Meng, Yiguang Hong, Hong Wang, Zongli Lin, and Karl H Johansson. A survey of distributed optimization. *Annual Reviews in Control*, 2019.

[104] Kevin Scaman, Francis Bach, Sébastien Bubeck, Laurent Massoulié, and Yin Tat Lee. Optimal algorithms for non-smooth distributed optimization in networks. In *Advances in Neural Information Processing Systems*, pages 2740–2749, 2018.

[105] Luc Moreau. Stability of continuous-time distributed consensus algorithms. In *2004 43rd IEEE conference on decision and control (CDC)*, volume 4, pages 3998–4003, 2004.

[106] S. Nabavi, J. Zhang, and A. Chakrabortty. Distributed optimization algorithms for wide-area oscillation monitoring in power systems using interregional pmu-pdc architectures. *IEEE Transactions on Smart Grid*, 6(5):2529–2538, Sept 2015.

[107] F.A. Longstaff and E.S. Schwartz. Valuing american options by simulation: a simple least-squares approach. *Review of Financial studies*, 14(1):113–147, 2001.

[108] C. G. Lopes and A. H. Sayed. Diffusion least-mean squares over adaptive networks: Formulation and performance analysis. *IEEE Transactions on Signal Processing*, 56(7):3122–3136, July 2008.

[109] G. Mateos, I. D. Schizas, and G. B. Giannakis. Distributed recursive least-squares for consensus-based in-network adaptive estimation. *IEEE Transactions on Signal Processing*, 57(11):4583–4588, Nov 2009.

[110] Anit Kumar Sahu, Soummya Kar, José MF Moura, and H Vincent Poor. Distributed constrained recursive nonlinear least-squares estimation: Algorithms and asymptotics. *IEEE Transactions on Signal and Information Processing over Networks*, 2(4):426–441, 2016.

[111] R. Zhang and J. Kwok. Asynchronous distributed admm for consensus optimization. In *International Conference on Machine Learning*, pages 1701–1709, 2014.

[112] E. Wei and A. Ozdaglar. Distributed alternating direction method of multipliers. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 5445–5450. IEEE, 2012.

[113] 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.

[114] D. Jakovetic, J. M. F. Moura, and J. Xavier. Fast distributed gradient methods. *IEEE Transactions on Automatic Control*, 59(5):1131–1146, 2014.

[115] Tsung-Hui Chang, Angelia Nedić, and Anna Scaglione. Distributed constrained optimization by consensus-based primal-dual perturbation method. *IEEE Transactions on Automatic Control*, 59(6):1524–1538, 2014.

[116] J. Wang and N. Elia. Distributed least square with intermittent communications. In *2012 American Control Conference (ACC)*, pages 6479–6484, June 2012.

[117] J. Wang and N. Elia. A control perspective for centralized and distributed convex optimization. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pages 3800–3805, Dec 2011.

[118] F.R. Chung. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997.

[119] J. Wang and N. Elia. Control approach to distributed optimization. In *2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 557–561, Sept 2010.

[120] U.M. Ascher, S.J. Ruuth, and R.J. Spiteri. Implicit-explicit runge-kutta methods for time-dependent partial differential equations. *Applied Numerical Mathematics*, 25(2):151–167, 1997.

[121] G. H. Golub and C. F. Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

[122] Jo Bo Rosen. The gradient projection method for nonlinear programming. part i. linear constraints. *Journal of the society for industrial and applied mathematics*, 8(1):181–217, 1960.

[123] Jiliang Luo and Kenzo Nonami. Approach for transforming linear constraints on petri nets. *IEEE Transactions on Automatic Control*, 56(12):2751–2765, 2011.

[124] Angelia Nedic, Alex Olshevsky, and Wei Shi. Achieving geometric convergence for distributed optimization over time-varying graphs. *SIAM Journal on Optimization*, 27(4):2597–2633, 2017.

[125] Dimitri P Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997.

[126] Hassan K Khalil. Nonlinear systems. *Upper Saddle River*, 2002.

[127] Stephen Wiggins. *Introduction to applied nonlinear dynamical systems and chaos*, volume 2. Springer Science & Business Media, 2003.

[128] Tansel Yucelen and Magnus Egerstedt. Control of multiagent systems under persistent disturbances. In *2012 American Control Conference (ACC)*, pages 5264–5269. IEEE, 2012.

[129] A Jadbabaie, J Lin, and AS Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6):988–1001, 2003.

[130] S. Sundaram and C. N. Hadjicostis. Finite-time distributed consensus in graphs with time-invarient topologies. In *Proceedings of American Control Conference*, pages 711–716, July 2007.

[131] S. Mou and M. Cao. Distributed averaging using compensation. *IEEE Communication Letters*, 17(8):1672–1675, 2013.

[132] X. Chen, M. A. Belabbas, and T. Basar. Distributed averaging with linear objective maps. *Automatica*, 70(3):179–188, August 2016.

[133] S. Sundaram and C. N. Hadjicostis. Distributed function calculation via linear iterative strategies in the presence of malicious agents. *IEEE Transactions on Automatic Control*, 56(7):1495–1508, 2011.

[134] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(2):382–401, 1982.

[135] Z. Bouzid, M. G. Potop-Butucaru, and S. Tixeuil. Optimal byzantine resilient convergence in uni-dimensional robot networks. *Theoretical Computer Science*, 411(34):3154–3168, 2010.

[136] F. Pasqualetti, A. Bicchi, and F. Bullo. Consensus computation in unreliable networks: a system theoretic approach. *IEEE Transactions on Automatic Control*, 57(1):90–104, 2012.

[137] N. H. Vaidya, L. Tseng, and G. Liang. Iterative approximate byzantine consensus in arbitrary directed graphs. In *Proceedings of ACM Symposium on Principles of Distributed Computing*, pages 365–374, 2012.

[138] Hyongju Park and Seth Hutchinson. An efficient algorithm for fault-tolerant rendezvous of multi-robot systems with controllable sensing range. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 358–365, 2016.

[139] H. Tverberg. A generalization of radon's theorem. *Journal of the London Mathematical Society*, 41:123–128, 1966.

[140] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)*, 22(4):469–483, 1996.

[141] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

VITA

Xuan Wang was born in Luanxian, Hebei Province, China. At the age of 4, he moved with his family to the city of Harbin, Heilongjiang Province, and grew up there. In 2010, he joined the Harbin Institute of Technology and received his bachelor's degree with first-class honors in Aerospace Engineering in 2014. Then he kept studying at the same university, under the guidance of Prof. Huijun Gao and Okyay Kaynak, and received his master's degree in Control Science and Engineering in 2016. In August 2016, he joined the School of Aeronautics and Astronautics, Purdue University as a Ph.D. student. He has played the leading role in several projects funded by Northrop Grumman Corp., AFRL and CRISP, covering the areas of autonomy and intelligence in networked multi-agent systems; Resilient cyber-physical systems; and the resources/task allocation in multi-robot swarms. He is the receiver of the *Outstanding Research Award, Koerner Scholarship and Bilsland Fellowship*, Purdue University. He also serves as the reviewer for a number of journals, including IEEE Transactions on Automatic Control, Control of Network Systems, Intelligent Transportation Systems, Automatica.