UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

DISTRIBUTED COMPUTATION AND OPTIMIZATION OVER NETWORKS

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

DOCTOR OF PHILOSOPHY

By

JIE LU
Norman, Oklahoma
2011

DISTRIBUTED COMPUTATION AND OPTIMIZATION OVER NETWORKS

A DISSERTATION APPROVED FOR THE
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

BY

To my family

Acknowledgements

I would like to express deep gratitude to my advisor, Dr. Choon Yik Tang, for providing tremendous mentoring, help, and support throughout my Ph.D. program. I want to thank him for leading me into the world of research and consistently providing invaluable guidance in every stage of my research.

I am grateful to Dr. J.R. Cruz, Dr. Joseph P. Havlicek, Dr. Nikola Petrov, and Dr. Thordur Runolfsson for their interests in my research and for serving on my dissertation committee. I have greatly benefited from their insightful suggestions on my work.

I also wish to thank my parents, Zhiping Lu and Peifang Zhang, for their endless love, care, and support. Special thanks to my mother who came to Norman and stayed with me for the last six months.

Finally, financial support from the National Science Foundation is gratefully acknowledged.

Table of Contents

List of Tables

List of Figures

Abstract

DISTRIBUTED COMPUTATION AND OPTIMIZATION OVER NETWORKS

Jie Lu, Ph.D.
The University of Oklahoma, 2011

Supervisor: Choon Yik Tang

This dissertation is devoted to the development of efficient, robust, and scalable distributed algorithms, which enable agents in a large-scale, multi-hop network to cooperatively compute a global quantity, or solve an optimization problem, with only local interactions and without any centralized coordination. Algorithms of this nature are attracting growing interest from a number of scientific communities due to their broad application, for example, to autonomous agent coordination and control in mobile ad hoc networks, distributed signal processing and data fusion in wireless sensor networks, and studies of opinion dynamics in social networks.

In this dissertation, we address three fundamental problems in the area, namely: averaging, solving of positive definite linear equations, and unconstrained separable convex optimization. Based on a blend of tools and ideas from system, optimization, and graph theories, we construct a novel set of distributed algorithms—including continuous- and discrete-time, gossip and asynchronous—which solve these problems over undirected networks with arbitrary (and, in some cases, time-varying) topologies and agent memberships. We also analyze the properties of these algorithms, including their convergence rates and complexity characteristics, and compare them with existing schemes, showing analytically and numerically that our algorithms possess several appealing features.

The major contributions of this dissertation are as follows: first, we show that Lyapunov stability theory may be used to shape the behavior of asynchronous distributed algorithms. This finding allows us to introduce the notion of greedy, decentralized, feedback iteration control, leading to a class of *Controlled Hopwise* algorithms, which are highly bandwidth/energy efficient in wireless networks. The finding also creates a new paradigm in the design of asynchronous distributed algorithms, where iterations are opportunistically controlled, as opposed to being randomized.

Second, we show that the Bregman divergence of the Lagrangian of a separable convex optimization problem may be used to form a common Lyapunov function. This result enables us to derive a family of *Zero-Gradient-Sum* algorithms, which yield nonlinear networked dynamical systems on an invariant manifold, and which differ fundamentally from, and have pros and cons over, the existing subgradient algorithms. The derivation also shows that a gossip variant within the family generalizes the classic Pairwise Averaging, and the family itself is a natural generalization of several well-known algorithms for distributed consensus, to distributed convex optimization.

Finally, we provide a series of analysis of the properties of our algorithms (e.g., boundedness, asymptotic and exponential convergence, lower and upper bounds on convergence rates, scalability) on various networks (e.g., path, cycle, regular, complete, and general graphs), describing explicitly the dependency of such properties on network topologies, problem characteristics, and algorithm parameters, including the algebraic connectivity, Laplacian spectral radius, and function curvatures.

Chapter 1 Introduction

1.1 Motivation

Emerging technologies on intelligent devices have triggered the vision of many applications of large-scale networks, including target tracking by a mobile ad hoc network [15, 71], estimation of a physical phenomenon by a wireless sensor network [1, 18], demonstration of flocking/swarming by a team of mobile robots [7, 49], resource allocation in a computer network [14, 29], and study of social interactions and opinion dynamics in a social network [12, 33]. To realize these applications, nodes in such networks may have to operate autonomously in dynamic and infrastructure-less environments with severe bandwidth and energy constraints, communicate in multi-hop fashion over unreliable links, and accomplish tasks that require extensive processing of information, rapid decentralized decision making, and precise coordination of actions. Therefore, it is highly desirable that such networks possess the ability to perform in-network computation and optimization: efficiently compute a quantity or solve an optimization problem, where the data that determine the quantity or the problem are distributed across network and observed by nodes.

In principle, in-network computation and optimization may be accomplished via *flooding*, whereby every node floods the network with its observation, as well as a *centralized* scheme, whereby a central node uses an overlay tree to collect all the node observations, calculate the solution, and send it back to every node. These two methods, unfortunately, have serious limitations: flooding is extremely bandwidth and energy inefficient because it propagates redundant information across the network, ignoring the fact that the ultimate goal is to simply determine the desirable quantity or an optimizer. The centralized scheme, on the other hand, is vulnerable to node mobility, node membership changes, and single-point failures, making it necessary to frequently maintain the overlay tree and occasionally start over with a new central node, both of which are rather costly to implement.

The limitations of flooding as well as the centralized scheme have motivated the search for distributed algorithms where each node in a network communicates and shares information with its neighbors only. Clearly, such algorithms require neither flooding of node observations, nor construction of overlay trees and routing tables, to execute. Moreover, the decentralized nature of distributed algorithms makes them more robust to dynamic environments and unreliable links. Thus, the goal of this research is to develop robust, scalable, and efficient distributed algorithms for computation and optimization over networks.

1.2 Literature Review

The current literature provides a growing collection of distributed algorithms for in-network computation and optimization, which may be simply referred to as *distributed computation* and *distributed optimization*.

For distributed computation, one line of research is *distributed averaging*, i.e., computing the network-wide average of node observations. This problem arises in many applications. For example, by averaging their individual throughputs, an ad hoc network of computers can assess how well the network, as a whole, is performing, and by averaging their humidity measurements, a wireless network of sensing agents can cooperatively detect the occurrence of local, deviation-from-average anomalies. To date, a collection of distributed averaging schemes with continuous-time [16, 52, 69], discrete-time synchronous [20, 30, 31, 52, 53, 55, 56, 64, 69, 74, 75, 78], and discrete-time asynchronous [8, 11, 13, 20, 26, 36, 38, 39, 72] settings have been developed.

Another distributed computation problem is to determine the solution for a system of linear equations where each parameter is the sum of a set of node observations. Examples of its applications include finding the least-squares solution of a distributed sensor fusion problem [76, 77] and solving unconstrained quadratic programming problems over networks. The current literature offers several distributed algorithms for solving this problem, including the continuous-time algorithm in [66], as well as the discrete-time algorithms in [76, 77] which find the solution by computing the average of each parameter of the linear equations.

In addition, distributed algorithms for finding the maximum of node observations have been introduced in [13, 17, 39, 41, 68]. In [26, 30, 39, 41], the problem of decentralizedly computing the sum of node observations has been explored. Distributed algorithms for computing the power mean of node observations have also been reported in [2, 17, 39]. Furthermore, *distributed consensus*, a topic closely related to distributed computation, where nodes seek to achieve an arbitrary network-wide consensus on their individual opinions, has also been extensively studied; see [4, 34] for early treatments, [20, 21, 24, 25, 40, 52, 54, 63, 67, 70] for more recent work, and [50] for a survey.

Distributed optimization problems are generally more complicated compared to distributed computation, among which the most common problem may be *distributed convex optimization*, where the objective and constraint functions are all convex. A special case of distributed convex optimization is that the objective function is the sum of the convex functions observed by nodes, which has found diverse applications. For example, least-squares estimation [65], robust estimation [65], energy-based source localization [57], and clustering and density estimation [57] are all in the form of this special case. As another example, consider a social network, where each individual's level of dissatisfaction if the network takes a decision may be represented by a convex function, so that finding an optimal decision means minimizing the total dissatisfaction across the network, where everyone's voice is heard. To date, a family of discrete-time subgradient algorithms [27,28,32,42–47,57–62,65] for solving this problem have been reported in the literature. These algorithms may be classified into two groups. The first group is incremental [28,42–44,57–59,61,65], relying on the passing of an estimate on an optimizer of the convex optimization problem. Incremental subgradient algorithms can be further categorized into cyclic ones [42–44,57–59,61,65], which require the estimate to be passed along a Hamiltonian cycle that visits every node exactly once, and non-cyclic ones [28,42–44], which allow the estimate to be passed around the network randomly. The second group is non-incremental [27,32,45–47,60,62], which relies instead on combining subgradient updates with linear consensus iterations. All these subgradient algorithms need appropriate choices of stepsizes to let the estimate(s) move along the gradient of the observed functions and approach an optimizer of the problem.

1.3 Original Contributions

In this dissertation, a collection of distributed algorithms that solve three fundamental in-network computation and optimization problems, namely, averaging, solving of positive definite linear equations, and unconstrained separable convex optimization, are designed and analyzed.

The dissertation starts with addressing the problem of averaging numbers across a wireless network from an important, but largely neglected, viewpoint: *bandwidth/energy efficiency*. We show that existing distributed averaging schemes have several drawbacks and are inefficient, producing networked dynamical systems that evolve with wasteful communications. Motivated by this, we develop *Controlled Hopwise Averaging* (CHA), a distributed asynchronous algorithm that attempts to "make the most" out of each iteration by fully exploiting the broadcast nature of wireless medium and enabling control of when to initiate an iteration. We show that CHA admits a common quadratic Lyapunov function for analysis, derive bounds on its exponential convergence rate, and show that they outperform the convergence rate of Pairwise Averaging for some common graphs. We also introduce a new way to apply Lyapunov stability theory, using the Lyapunov function to perform greedy, decentralized, feedback iteration control. Through extensive simulation on random geometric graphs, we show that CHA is substantially more efficient than several existing schemes, requiring far fewer transmissions to complete an averaging task.

Next, a family of distributed asynchronous algorithms for solving symmetric positive definite systems of linear equations over agent and wireless networks are constructed. In particular, we develop *Subset Equalizing* (SE), a Lyapunov-based algorithm for solving the problem over agent networks with arbitrary asynchronous interactions and spontaneous membership dynamics, both of which may be exogenously driven and completely unpredictable. To analyze the behavior of SE, we introduce several notions of network connectivity, capable of handling such interactions and membership dynamics, and a time-varying quadratic Lyapunov-like function, defined on a state space with changing dimension. Based on them, we derive sufficient conditions for ensuring the boundedness, asymptotic convergence, and exponential convergence of SE, and show that these conditions are mild. Moreover, we study the interplay among wireless communications, distributed algorithms, and control in solving such quadratic optimization problems over multi-hop wireless networks with fixed topologies. Building on the results from SE, we develop and analyze *Pairwise*, *Groupwise*, *Random Hopwise*, and *Controlled Hopwise Equalizing* (PE, GE, RHE, and CHE), showing along the way how the broadcast nature of wireless transmissions may be fully utilized, how undesirable overlapping iterations may be avoided, and how iterations may be feedback controlled in a greedy, decentralized, Lyapunov-based fashion, leading to CHE, which yields provable exponential convergence and a quantifiable bound on the convergence rate. Through extensive simulation, we show that GE, RHE, and CHE are dramatically more efficient and scalable than two existing, average-consensus-based schemes, with CHE having the best performance.

Finally, we address the problem of distributed convex optimization from both discrete- and continuous-time standpoints. More specifically, with a few additional mild assumptions, we develop two gossip-style, non-gradient-based algorithms, referred to as *Pairwise Equalizing* (PE) and *Pairwise Bisectioning* (PB), for achieving unconstrained, separable, convex optimization over undirected networks with time-varying topologies, which are fundamentally different from the existing subgradient algorithms. We show that PE and PB are easy to implement, bypass limitations of the subgradient algorithms, and produce switched, nonlinear, networked dynamical systems that admit a common Lyapunov function based on the Bregman divergence [10] and asymptotically converge. Moreover, PE generalizes the well-known Pairwise Averaging and Randomized Gossip Algorithm and extends naturally to networks with both time-varying topologies and node memberships, while PB relaxes a requirement of PE, allowing nodes to never share their local functions. Furthermore, we introduce a new approach to the problem: control of distributed convex optimization, which extends the ideas of PE and the notion of greedy, decentralized, feedback iteration control for CHA using the Bregman-divergence-based Lyapunov function for PE and PB. The resulting distributed asynchronous algorithm, referred to as *Controlled Hopwise Equalizing* (CHE), is shown via extensive simulation to be significantly more bandwidth/energy efficient than several existing subgradient algorithms over wireless networks with fixed topologies, requiring far less communications to solve a convex optimization problem. In addition to the above discrete-time distributed algorithms, we derive a set of continuous-time distributed algorithms that solve the problem over undirected networks with fixed topologies. The algorithms are developed using a Lyapunov function candidate that exploits convexity, and are called *Zero-Gradient-Sum* (ZGS) algorithms as they yield nonlinear networked dynamical systems that evolve invariantly on a zero-gradient-sum manifold and converge asymptotically to the unknown optimizer. We also describe a systematic way to construct ZGS algorithms, show that a subset of them actually converge exponentially, and obtain lower and upper bounds on their convergence rates in terms of the network topologies, problem characteristics, and algorithm parameters, including the algebraic connectivity, Laplacian spectral radius, and function curvatures. The findings may be regarded as a natural generalization of several well-known algorithms and results for distributed consensus, to distributed convex optimization.

1.4 Dissertation Outline

The outline of this dissertation is as follows: Chapter 2 studies distributed averaging over networks, in which CHA is developed. Chapters 3–4 present distributed algorithms for solving positive definite linear equations over networks. In particular, Chapter 3 proposes SE for agent networks and Chapter 4 constructs a few distributed algorithms for wireless networks. Chapters 5–7 address the problem of distributed convex optimization over networks, where PE and PB are developed in Chapter 5, CHE is introduced in Chapter 6, and ZGS algorithms are constructed in Chapter 7. Finally, Chapter 8 concludes the dissertation and provides several possible future research directions. The proofs for Chapters 2–6 are included in Appendices A–E, respectively.

Chapter 2 Controlled Hopwise Averaging

2.1 Introduction

Distributed averaging is a fundamental problem in distributed computation that finds many applications in multi-agent systems, ad hoc networks, sensor networks and the likes. Due to its significance, the problem has been widely studied (see, e.g., [8, 11, 13, 16, 20, 26, 30, 31, 36, 38, 39, 52, 53, 55, 56, 64, 69, 72, 74, 75, 78]) for different network models (e.g., wired or wireless; undirected or directed links; fixed or time-varying topologies), with different communication assumptions (e.g., without delays, errors, and quantization or with), and in different time domains (e.g., continuous- or discrete-time; synchronous or asynchronous). The research efforts have led to a growing list of algorithms, including Pairwise Averaging [72], Randomized Gossip Algorithm [8], Accelerated Gossip Algorithm [11], Distributed Random Grouping [13], and Linear Prediction-Based Accelerated Averaging [56], to name just a few.

Although the current literature offers a rich collection of distributed averaging schemes along with in-depth analysis of their behaviors, their efficacy from a *bandwidth/energy efficiency* standpoint has not been examined. This chapter is devoted to studying the distributed averaging problem from this standpoint. Its contributions are as follows: we first show that the existing schemes—regardless of whether they are developed in continuous- or discrete-time, for synchronous or asynchronous models—have a few deficiencies and are inefficient, producing networked dynamical systems that evolve with wasteful communications. To address these issues, we develop *Random Hopwise Averaging* (RHA), an asynchronous distributed averaging algorithm with several positive features, including a novel one among the asynchronous schemes: an ability to fully exploit the broadcast nature of wireless medium, so that no overheard information is ever wastefully discarded. We show that RHA admits a common quadratic Lyapunov function, is almost surely asymptotically convergent, and eliminates all but one of the deficiencies facing the existing schemes.

To tackle the remaining deficiency, on lack of control, we introduce the concept of *feedback iteration control*, whereby individual nodes use feedback to control when to initiate an iteration. Although simple and intuitive, this concept, somewhat surprisingly, has not been explored in the literature on distributed averaging [8, 11, 13, 16, 20, 26, 30, 31, 36, 38, 39, 52, 53, 55, 56, 64, 69, 72, 74, 75, 78] and distributed consensus [4, 20, 21, 24, 25, 34, 40, 50, 52, 54, 63, 67, 70]. We show that RHA, along with the common quadratic Lyapunov function, exhibits features that enable a greedy, decentralized approach to feedback iteration control, which leads to bandwidth/energy-efficient iterations at zero feedback cost. Based on this approach, we present two modified versions of RHA: an ideal version referred to as *Ideal Controlled Hopwise Averaging* (ICHA), and a practical one referred to simply as *Controlled Hopwise Averaging* (CHA). We show that ICHA yields a networked dynamical system with state-dependent switching, derive deterministic bounds on its exponential convergence rate for general and specific graphs, and show that the bounds are better than the stochastic convergence rate of Pairwise Averaging [20, 72] for path, cycle, and complete graphs. We also show that CHA is able to closely mimic the behavior of ICHA, achieving the same bounds on its convergence rate. Finally, via extensive simulation on random geometric graphs, we demonstrate that CHA is substantially more bandwidth/energy efficient than Pairwise Averaging [72], Consensus Propagation [38], Algorithm A2 of [36], and Distributed Random Grouping [13], requiring far fewer transmissions to complete an averaging task. In particular, CHA is twice more efficient than the most efficient existing scheme when the network is sparsely connected.

The outline of this chapter is as follows: Section 2.2 formulates the distributed averaging problem. Section 2.3 describes the deficiencies of the existing schemes. Sections 2.4 and 2.5 develop RHA and CHA and characterize their convergence properties. In Section 2.6, their comparison with several existing schemes is carried out. Finally, Section 2.7 concludes the chapter. The proofs of the main results are included in Appendix A.

2.2 Problem Formulation

Consider a multi-hop wireless network consisting of $N \geq 2$ nodes, connected by L bidirectional links in a fixed topology. The network is modeled as a connected, undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, \ldots, N\}$ represents the set of N nodes (vertices) and $\mathcal{E} \subset \{\{i, j\} : i, j \in \mathcal{V}, i \neq j\}$ represents the set of L links (edges). Any two nodes $i, j \in \mathcal{V}$ are one-hop neighbors and can communicate if and only if $\{i, j\} \in \mathcal{E}$. The set of one-hop neighbors of each node $i \in \mathcal{V}$ is denoted as $\mathcal{N}_i = \{j \in \mathcal{V} : \{i, j\} \in \mathcal{E}\}$, and the communications are assumed to be delay- and error-free, with no quantization. Each node $i \in \mathcal{V}$ observes a scalar $y_i \in \mathbb{R}$, and all the N nodes wish to determine the network-wide average $x^* \in \mathbb{R}$ of their individual observations, given by

$$x^* = \frac{1}{N} \sum_{i \in \mathcal{V}} y_i. \tag{2.1}$$

Given the above model, the problem addressed in this chapter is how to construct a distributed averaging algorithm—continuous- or discrete-time, synchronous or otherwise—with which each node $i \in \mathcal{V}$ repeatedly communicates with its one-hop neighbors, iteratively updates its estimate $\hat{x}_i \in \mathbb{R}$ of the unknown average x^* in (2.1), and asymptotically drives $\hat{x}_i$ to x^*—all while consuming bandwidth and energy efficiently.

The bandwidth/energy efficiency of an algorithm is measured by *the number of real-number transmissions it needs to drive all the $\hat{x}_i$'s to a sufficiently small neighborhood of x^**, essentially completing the averaging task. This quantity is a natural measure of efficiency because the smaller it is, the lesser bandwidth is occupied, the lesser energy is expended for communications, and the faster an averaging task may be completed. These, in turn, imply more bandwidth and time for other tasks, smaller probability of collision, longer lifetime for battery-powered nodes, and possible earlier return to sleep mode, all of which are desirable. The quantity also allows algorithms with different numbers of real-number transmissions per iteration to be fairly compared. Although, in networking, every message inevitably contains overhead (e.g., transmitter/receiver IDs and message type), we exclude such overhead when measuring efficiency since it is not inherent to an algorithm, may be reduced by piggybacking messages, and becomes negligible when averaging long vectors.

2.3 Deficiencies of Existing Schemes

As was pointed out in Section 2.1, the current literature offers a variety of distributed averaging schemes for solving the problem formulated in Section 2.2. Unfortunately, as is explained below, they suffer from a number of deficiencies, especially a lack of bandwidth/energy efficiency, by producing networked dynamical systems that evolve with wasteful real-number transmissions.

The *continuous-time* algorithms in [16, 52, 69] have the following deficiency:

D1. *Costly discretization*: As immensely inefficient as flooding is, the continuous-time algorithms in [16, 52, 69] may be more so: flooding only requires N^2 real-number transmissions for all the N nodes to exactly determine the average x^* (since it takes N real-number transmissions for each node $i \in \mathcal{V}$ to flood the network with its y_i), whereas these algorithms may need far more than that to essentially complete an averaging task. For instance, the algorithm in [52] updates the estimates $\hat{x}_i$'s of x^* according to the differential equation

$$\frac{d\hat{x}_i(t)}{dt} = \sum_{j \in \mathcal{N}_i} (\hat{x}_j(t) - \hat{x}_i(t)), \quad \forall i \in \mathcal{V}. \tag{2.2}$$

To realize (2.2), each node $i \in \mathcal{V}$ has to continuously monitor the $\hat{x}_j(t)$ of every one-hop neighbor $j \in \mathcal{N}_i$. If this can be done without wireless communications (e.g., by direct sensing), then the bandwidth/energy efficiency issue is moot. If wireless communications must be employed, then (2.2) has to be discretized, either exactly via a zero-order hold, i.e.,

$$\hat{x}_i((k+1)T) = \sum_{j \in \mathcal{V}} h_{ij} \hat{x}_j(kT), \quad \forall i \in \mathcal{V}, \tag{2.3}$$

or approximately via numerical techniques such as the Euler forward difference method, i.e.,

$$\frac{\hat{x}_i((k+1)T) - \hat{x}_i(kT)}{T} = \sum_{j \in \mathcal{N}_i} (\hat{x}_j(kT) - \hat{x}_i(kT)), \quad \forall i \in \mathcal{V}, \tag{2.4}$$

where each $h_{ij} \in \mathbb{R}$ is the ij-entry of $e^{-\mathbf{L}T}$, $\mathbf{L} \in \mathbb{R}^{N \times N}$ is the Laplacian matrix of the graph $\mathcal{G}$ that governs the dynamics (2.2), and $T > 0$ is the sampling period. Regardless of (2.3) or (2.4), they may be far more costly to realize than flooding: with (2.3), N^2 real-number transmissions are already needed per iteration (since, in general, $h_{ij} \neq 0\ \forall i, j \in \mathcal{V}$, so that each node $i \in \mathcal{V}$ has to flood the network with its $\hat{x}_i(kT)$, for every k). In contrast, with (2.4), only N real-number transmissions are needed per iteration (since each node $i \in \mathcal{V}$ only has to wirelessly transmit its $\hat{x}_i(kT)$ once, to every one-hop neighbor $j \in \mathcal{N}_i$, for every k). However, the number of iterations, needed for all the $\hat{x}_i(kT)$'s to converge to an acceptable neighborhood of x^*, may be very large, since the sampling period T must be sufficiently small for (2.4) to be stable. If the number of iterations needed exceeds N—which is possible and likely so with a conservatively small T—then (2.4) would be worse than flooding[1].

The *discrete-time synchronous* algorithms in [20, 30, 31, 52, 53, 55, 56, 64, 69, 74, 75, 78] have the following deficiencies:

D2. *Clock synchronization*: The discrete-time synchronous algorithms in [20, 30, 31, 52, 53, 55, 56, 64, 69, 74, 75, 78] require all the N nodes to always have the same clock to operate. Although techniques for reducing clock syn-

[1]Flooding is, of course, more storage and bookkeeping intensive.

chronization errors are available, it is still desirable that this requirement can be removed.

D3. *Forced transmissions*: The algorithms in [20,31,52,53,55,56,64,69,74,75] update the estimates $\hat{x}_i$'s of x^* according to the difference equation

$$\hat{x}_i(k+1) = w_{ii}(k)\hat{x}_i(k) + \sum_{j \in \mathcal{N}_i} w_{ij}(k)\hat{x}_j(k), \quad \forall i \in \mathcal{V}, \tag{2.5}$$

where each $w_{ij}(k) \in \mathbb{R}$ is a weighting factor that is typically constant. The $w_{ij}(k)$'s may be specified in several ways, including choosing them to maximize the convergence rate [74] or minimize the mean-square deviation [75]. However, no matter how the $w_{ij}(k)$'s are chosen, these algorithms are bandwidth/energy inefficient because the underlying update rule (2.5) simply forces every node $i \in \mathcal{V}$ at each iteration k to transmit its $\hat{x}_i(k)$ to its one-hop neighbors, irrespective of whether such transmissions are worthy. It is possible, for example, that the $\hat{x}_i(k)$'s of a cluster of nearby nodes are almost equal, so that their $\hat{x}_i(k+1)$'s, being convex combinations of their $\hat{x}_i(k)$'s, are also almost equal, causing their transmissions to be unworthy. The fact that N real-number transmissions are needed per iteration also implies that (2.5) must drive all the $\hat{x}_i(k)$'s to an acceptable neighborhood of x^* within at most N iterations, in order to just outperform flooding.

D4. *Computing intermediate quantities*: The scheme in [53] uses two parallel runs of a consensus algorithm to obtain two consensus values and defines each $\hat{x}_i(k)$ as the ratio of these two values. While possible, this scheme is likely inefficient because it attempts to compute two *intermediate* quantities, as opposed to computing x^* directly.

The *discrete-time asynchronous* algorithms in [8,11,13,20,26,36,38,39, 72] have the following deficiencies:

D5. *Wasted receptions*: Each iteration of Pairwise Averaging [72], Anti-Entropy Aggregation [26, 39], Randomized Gossip Algorithm [8], and Accelerated Gossip Algorithm [11] involves a pair of nodes transmitting to each other their state variables. Due to the broadcast nature of wireless medium, their transmissions are overheard by unintended nearby nodes, who would immediately discard this "free" information, instead of using it to possibly speed up convergence, enhancing bandwidth/energy efficiency. Hence, these algorithms result in wasted receptions. The same can be said about Consensus Propagation [38] and Algorithm A2 of [36], although they do not assume pairwise exchanges. It can also be said about Distributed Random Grouping [13], which only slightly exploits such broadcast nature: the leader of a group does, but the members, who contribute the majority of the transmissions, do not.

D6. *Overlapping iterations*: Pairwise Averaging [72], Anti-Entropy Aggregation [26, 39], Randomized Gossip Algorithm [8], Accelerated Gossip Algorithm [11], and Distributed Random Grouping [13] require sequential transmissions from multiple nodes to execute an iteration. This suggests that before an iteration completes, the nodes involved may be asked to participate in other iterations initiated by those unaware of the ongoing iteration. Thus, these algorithms are prone to overlapping iterations and, therefore, to deadlock situations [36]. It is noted that this practical issue is naturally avoided by Consensus Propagation [38] and explicitly handled by Algorithms A1 and A2 of [36].

D7. *Uncontrolled iterations*: The discrete-time asynchronous algorithms in [8, 11, 13, 26, 36, 38, 39, 72] do not let individual nodes use information available to them during runtime (e.g., history of the state variables they locally maintain) to control when to initiate an iteration and who to include in the iteration. Indeed, Pairwise Averaging [72], Anti-Entropy Aggregation [26,39], Accelerated Gossip Algorithm [11], Consensus Propagation [38], and Algorithm A2 of [36] focus mostly on how nodes would update their state variables during an iteration, saying little about how they could use such information to control the iterations. Randomized Gossip Algorithm [8] and Distributed Random Grouping [13], on the other hand, let nodes randomly initiate an iteration according to some probabilities. Although these probabilities may be optimized [8, 13], the optimization is carried out *a priori*, dependent only on the graph $\mathcal{G}$ and independent of the nodes' state variables during runtime. Consequently, wasteful iterations may occur, despite the optimality. For instance, suppose Randomized Gossip Algorithm [8] is utilized, and a pair of adjacent nodes $i, j \in \mathcal{V}$ have just finished gossiping with each other, so that $\hat{x}_i$ and $\hat{x}_j$ are equal. Since the optimal probabilities are generally nonzero, nodes i and j may gossip with each other again before any of them gossips with someone else, causing $\hat{x}_i$ and $\hat{x}_j$ to remain unchanged, wasting that particular gossip. Similarly, suppose Distributed Random Grouping [13] is employed, and a node $i \in \mathcal{V}$ has just finished leading an iteration, so that $\hat{x}_i$ and $\hat{x}_j$ $\forall j \in \mathcal{N}_i$ are equal. Due again to nonzero probabilities, node i may lead another iteration before any of its one- or two-hop neighbors leads an iteration, causing $\hat{x}_i$ and $\hat{x}_j$ $\forall j \in \mathcal{N}_i$ to stay the same, wasting that particular iteration. These examples suggest that not letting nodes control the iterations is detrimental to bandwidth/energy efficiency and, conceivably, letting them do so may cut down on wasteful iterations, improving efficiency.

D8. *Steady-state errors*: Consensus Propagation [38] ensures that all the $\hat{x}_i$'s asymptotically converge to the same steady-state value. However, this value is, in general, not equal to x^* (see Figure 2.3 of Section 2.6 for an illustration). Although the error can be made arbitrarily small, it comes at the expense of increasingly slow convergence [38], which is undesirable.

D9. *Lack of convergence guarantees*: Accelerated Gossip Algorithm [11], developed based on the power method in numerical analysis, is shown by simulation to have the potential of speeding up the convergence of Randomized Gossip Algorithm [8] by a factor of 10. Furthermore, whenever all the $\hat{x}_i$'s converge, they must converge to x^*. However, it was not established in [11] that they would always converge.

2.4 Random Hopwise Averaging

Deficiencies D1–D9 facing the existing distributed averaging schemes raise a question: *is it possible to develop an algorithm, which does not at all suffer from these deficiencies?* In this section, we construct an algorithm that simultaneously eliminates all but issue D7 with uncontrolled iterations. In the next section, we will modify the algorithm to address this issue.

To circumvent the costly discretization issue D1 facing the existing continuous-time algorithms and the clock synchronization and forced transmissions issues D2 and D3 facing the existing discrete-time synchronous algorithms, the algorithm we construct must be *asynchronous*, regardless of whether the nodes have access to the same global clock. To avoid issue D6 with overlapping iterations, each iteration of this algorithm must involve only a *single* node sending a *single* message to its one-hop neighbors, without needing them to reply. To tackle issue D5 with wasted receptions, all the neighbors, upon hearing the same message, have to "meaningfully" incorporate it into updating their state variables, rather than simply discarding it. To overcome issues D8 and D9 with steady-state errors and convergence guarantees, the algorithm must be asymptotically convergent to the correct average. Finally, to eliminate D4, it has to avoid computing intermediate quantities.

To develop an algorithm having the aforementioned properties, consider a networked dynamical system, defined on the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as follows: associated with each link $\{i,j\} \in \mathcal{E}$ are a parameter $c_{\{i,j\}} > 0$ and a state variable $x_{\{i,j\}} \in \mathbb{R}$ of the system. In addition, associated with each node $i \in \mathcal{V}$ is an output variable $\hat{x}_i \in \mathbb{R}$, which represents its estimate of the unknown average x^* in (2.1). Since the graph $\mathcal{G}$ has L links and N nodes, the system has L parameters $c_{\{i,j\}}$'s, L state variables $x_{\{i,j\}}$'s, and N output variables $\hat{x}_i$'s. To describe the system dynamics, let $x_{\{i,j\}}(0)$ and $\hat{x}_i(0)$ represent the initial values of $x_{\{i,j\}}$ and $\hat{x}_i$, and $x_{\{i,j\}}(k)$ and $\hat{x}_i(k)$ their values upon completing each iteration $k \in \mathbb{P}$, where $\mathbb{P}$ denotes the set of positive integers. With these notations, the state and output equations governing the system dynamics may be stated as

$$x_{\{i,j\}}(k) = \begin{cases} \dfrac{\sum_{\ell \in \mathcal{N}_{u(k)}} c_{\{u(k),\ell\}} x_{\{u(k),\ell\}}(k-1)}{\sum_{\ell \in \mathcal{N}_{u(k)}} c_{\{u(k),\ell\}}}, & \text{if } u(k) \in \{i,j\}, \\ x_{\{i,j\}}(k-1), & \text{otherwise}, \end{cases} \qquad \forall k \in \mathbb{P},\ \forall \{i,j\} \in \mathcal{E}, \tag{2.6}$$

$$\hat{x}_i(k) = \frac{\sum_{j \in \mathcal{N}_i} c_{\{i,j\}} x_{\{i,j\}}(k)}{\sum_{j \in \mathcal{N}_i} c_{\{i,j\}}}, \quad \forall k \in \mathbb{N},\ \forall i \in \mathcal{V}, \tag{2.7}$$

where $u(k) \in \mathcal{V}$ is a variable to be interpreted shortly and $\mathbb{N}$ denotes the set of nonnegative integers. Equation (2.7) says that the output variable associated with each node is a convex combination of the state variables associated with links incident to the node. Equation (2.6) says that at each iteration $k \in \mathbb{P}$, the state variables associated with links incident to node $u(k)$ are set equal to the same convex combination of their previous values. Equation (2.6) also implies that the system is a linear switched system, since (2.6) may be written as

$$\mathbf{x}(k) = \mathbf{A}_{u(k)}\mathbf{x}(k-1), \quad \forall k \in \mathbb{P}, \tag{2.8}$$

where $\mathbf{x}(k) \in \mathbb{R}^L$ is the state vector obtained by stacking the L $x_{\{i,j\}}(k)$'s, $\mathbf{A}_{u(k)} \in \mathbb{R}^{L\times L}$ is a time-varying matrix taking one of N possible values $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N$ depending on $u(k)$, and each $\mathbf{A}_i \in \mathbb{R}^{L\times L}$ is a row stochastic matrix whose entries depend on the $c_{\{i,j\}}$'s. Hence, the sequence $(u(k))_{k=1}^{\infty}$ fully dictates how the asynchronous iteration (2.6) takes place, or equivalently, how the system (2.8) switches. Throughout this section, we assume that $(u(k))_{k=1}^{\infty}$ is an independent and identically distributed random sequence with a uniform distribution, i.e.,

$$\mathrm{P}\{u(k) = i\} = \frac{1}{N}, \quad \forall k \in \mathbb{P},\ \forall i \in \mathcal{V}. \tag{2.9}$$

Remark 2.1. Clearly, alternatives to letting $(u(k))_{k=1}^{\infty}$ be random and equiprobable are possible, and perhaps beneficial. We will explore such alternatives in Section 2.5, when we discuss control. ■

For the system (2.6), (2.7), (2.9) to solve the distributed averaging problem, the $\hat{x}_i(k)$'s must asymptotically approach x^* of (2.1), i.e.,

$$\lim_{k\to\infty} \hat{x}_i(k) = x^*, \quad \forall i \in \mathcal{V}. \tag{2.10}$$

Due to (2.7), condition (2.10) is met if the $x_{\{i,j\}}(k)$'s satisfy

$$\lim_{k\to\infty} x_{\{i,j\}}(k) = x^*, \quad \forall \{i,j\} \in \mathcal{E}. \tag{2.11}$$

To ensure (2.11), the parameters $c_{\{i,j\}}$'s and initial states $x_{\{i,j\}}(0)$'s must satisfy a condition. To derive the condition, observe from (2.6) that no matter what $u(k)$ is, the expression $\sum_{\{i,j\}\in\mathcal{E}} c_{\{i,j\}} x_{\{i,j\}}(k)$ is conserved after every iteration $k \in \mathbb{P}$, i.e.,

$$\sum_{\{i,j\}\in\mathcal{E}} c_{\{i,j\}} x_{\{i,j\}}(k) = \sum_{\{i,j\}\in\mathcal{E}} c_{\{i,j\}} x_{\{i,j\}}(k-1), \quad \forall k \in \mathbb{P}. \tag{2.12}$$

Therefore, as it follows from (2.12) and (2.1), (2.11) holds only if the $c_{\{i,j\}}$'s and $x_{\{i,j\}}(0)$'s satisfy

$$\frac{\sum_{\{i,j\}\in\mathcal{E}} c_{\{i,j\}} x_{\{i,j\}}(0)}{\sum_{\{i,j\}\in\mathcal{E}} c_{\{i,j\}}} = \frac{\sum_{i\in\mathcal{V}} y_i}{N}. \tag{2.13}$$

To achieve (2.13), notice that the expressions $\sum_{\{i,j\}\in\mathcal{E}} c_{\{i,j\}}$ and $\sum_{\{i,j\}\in\mathcal{E}} c_{\{i,j\}} x_{\{i,j\}}(0)$ each has L terms, of which $|\mathcal{N}_i|$ terms are associated with links incident to node i, for every $i \in \mathcal{V}$, where $|\cdot|$ denotes the cardinality of a set. Hence, by letting each node $i \in \mathcal{V}$ evenly distribute the number 1 to the $|\mathcal{N}_i|$ terms in $\sum_{\{i,j\}\in\mathcal{E}} c_{\{i,j\}}$, i.e.,

$$c_{\{i,j\}} = \frac{1}{|\mathcal{N}_i|} + \frac{1}{|\mathcal{N}_j|}, \quad \forall \{i,j\} \in \mathcal{E}, \tag{2.14}$$

we get $\sum_{\{i,j\}\in\mathcal{E}} c_{\{i,j\}} = N$. Similarly, by letting each node $i \in \mathcal{V}$ evenly distribute its observation y_i to the $|\mathcal{N}_i|$ terms in $\sum_{\{i,j\}\in\mathcal{E}} c_{\{i,j\}} x_{\{i,j\}}(0)$, i.e.,

$$x_{\{i,j\}}(0) = \frac{\frac{y_i}{|\mathcal{N}_i|} + \frac{y_j}{|\mathcal{N}_j|}}{c_{\{i,j\}}}, \quad \forall \{i,j\} \in \mathcal{E}, \tag{2.15}$$

we get $\sum_{\{i,j\}\in\mathcal{E}} c_{\{i,j\}} x_{\{i,j\}}(0) = \sum_{i\in\mathcal{V}} y_i$. Thus, (2.14) and (2.15) together ensure (2.13), which is necessary for achieving (2.11).

Remark 2.2. Obviously, (2.14) and (2.15) are not the only way to select the $c_{\{i,j\}}$'s and $x_{\{i,j\}}(0)$'s. In fact, their selection may be posed as an optimization problem, analogous to the synchronous algorithms in [74, 75]. Nevertheless, (2.14) and (2.15) have the virtue of being simple and inexpensive to implement: for every link $\{i,j\} \in \mathcal{E}$, both $c_{\{i,j\}}$ and $x_{\{i,j\}}(0)$ depend only on local information $|\mathcal{N}_i|$, $|\mathcal{N}_j|$, y_i, and y_j that nodes i and j know, as opposed to on global information derived from the graph $\mathcal{G}$, which is typically difficult and costly to gather, but often the outcome of optimization. ■

The system (2.6), (2.7), (2.9) with parameters (2.14) and initial states (2.15) can be realized over the wireless network by having the nodes take the following actions: for every link $\{i,j\} \in \mathcal{E}$, nodes i and j each maintains a local copy of $x_{\{i,j\}}(k)$, denoted as $x_{ij}(k)$ and $x_{ji}(k)$, respectively, where they are meant to be always equal, so that the order of the subscripts is only used to indicate where they physically reside. Each node $i \in \mathcal{V}$, in addition to $x_{ij}(k)$ $\forall j \in \mathcal{N}_i$, also maintains $c_{\{i,j\}}$ $\forall j \in \mathcal{N}_i$ and $\hat{x}_i(k)$. To initialize the system, every node $i \in \mathcal{V}$ transmits $|\mathcal{N}_i|$ and y_i each once, to every one-hop neighbor $j \in \mathcal{N}_i$, so that upon completion, each node $i \in \mathcal{V}$ can calculate $c_{\{i,j\}}$ $\forall j \in \mathcal{N}_i$ from (2.14), $x_{ij}(0)$ $\forall j \in \mathcal{N}_i$ from (2.15), and $\hat{x}_i(0)$ from (2.7). To evolve the system, at each iteration $k \in \mathbb{P}$, a node $u(k) \in \mathcal{V}$ is selected randomly and equiprobably based on (2.9) to initiate the iteration. To describe the subsequent actions, note that (2.6) and (2.7) imply: (i) $\hat{x}_{u(k)}(k) = \hat{x}_{u(k)}(k-1)$; (ii) $x_{u(k)j}(k) = \hat{x}_{u(k)}(k)$ $\forall j \in \mathcal{N}_{u(k)}$; (iii) $x_{ju(k)}(k) = \hat{x}_{u(k)}(k)$ $\forall j \in \mathcal{N}_{u(k)}$; (iv) $x_{j\ell}(k) = x_{j\ell}(k-1)$ $\forall \ell \in \mathcal{N}_j - \{u(k)\}$ $\forall j \in \mathcal{N}_{u(k)}$; (v) $\hat{x}_j(k) = \frac{\sum_{\ell \in \mathcal{N}_j} c_{\{j,\ell\}} x_{j\ell}(k)}{\sum_{\ell \in \mathcal{N}_j} c_{\{j,\ell\}}}$ $\forall j \in \mathcal{N}_{u(k)}$; (vi) $x_{\ell m}(k) = x_{\ell m}(k-1)$ $\forall m \in \mathcal{N}_\ell$ $\forall \ell \in \mathcal{V} - (\{u(k)\} \cup \mathcal{N}_{u(k)})$; and (vii) $\hat{x}_\ell(k) = \hat{x}_\ell(k-1)$ $\forall \ell \in \mathcal{V} - (\{u(k)\} \cup \mathcal{N}_{u(k)})$. To execute (i) and (ii), node $u(k)$, upon being selected to initiate iteration k, sets $\hat{x}_{u(k)}(k)$ and $x_{u(k)j}(k)$ $\forall j \in \mathcal{N}_{u(k)}$ all to $\hat{x}_{u(k)}(k-1)$. To execute (iii), node $u(k)$ then transmits $\hat{x}_{u(k)}(k)$ once, to every one-hop neighbor $j \in \mathcal{N}_{u(k)}$, so that upon reception, each of them can set $x_{ju(k)}(k)$ to $\hat{x}_{u(k)}(k)$. Equations (iv) and (v) say that every neighbor

$j \in \mathcal{N}_{u(k)}$ experiences no change in the rest of its local copies and, hence, can compute $\hat{x}_j(k)$ from (v) upon finishing (iii). Finally, (vi) and (vii) say that the rest of the N nodes, i.e., excluding node $u(k)$ and its one-hop neighbors, experience no change in the variables they maintain.

The above node actions define a distributed averaging algorithm that runs iteratively and asynchronously on the wireless network. We refer to this algorithm as *Random Hopwise Averaging* (RHA), since every iteration is *randomly* initiated and involves state variables associated with links within one *hop* of each other. RHA may be expressed in a compact algorithmic form as follows:

Algorithm 2.1 (Random Hopwise Averaging)**.**
Initialization:

1. Each node $i \in \mathcal{V}$ transmits $|\mathcal{N}_i|$ and y_i to every node $j \in \mathcal{N}_i$.
2. Each node $i \in \mathcal{V}$ creates variables $x_{ij} \in \mathbb{R}\ \forall j \in \mathcal{N}_i$ and $\hat{x}_i \in \mathbb{R}$ and initializes them sequentially:
   $x_{ij} \leftarrow \frac{\frac{y_i}{|\mathcal{N}_i|}+\frac{y_j}{|\mathcal{N}_j|}}{c_{\{i,j\}}}, \quad \forall j \in \mathcal{N}_i,$
   $\hat{x}_i \leftarrow \frac{\sum_{j\in\mathcal{N}_i} c_{\{i,j\}} x_{ij}}{\sum_{j\in\mathcal{N}_i} c_{\{i,j\}}}.$

Operation: At each iteration:

3. A node, say, node i, is selected randomly and equiprobably out of the set $\mathcal{V}$ of N nodes.
4. Node i updates $x_{ij}\ \forall j \in \mathcal{N}_i$:
   $x_{ij} \leftarrow \hat{x}_i, \quad \forall j \in \mathcal{N}_i.$
5. Node i transmits $\hat{x}_i$ to every node $j \in \mathcal{N}_i$.
6. Each node $j \in \mathcal{N}_i$ updates x_{ji} and $\hat{x}_j$ sequentially:
   $x_{ji} \leftarrow \hat{x}_i,$

$\hat{x}_j \leftarrow \frac{\sum_{\ell \in \mathcal{N}_j} c_{\{j,\ell\}} x_{j\ell}}{\sum_{\ell \in \mathcal{N}_j} c_{\{j,\ell\}}}$. ■

Observe from Algorithm 2.1 that RHA requires an initialization overhead of $2N$ real-number transmissions to perform Step 1 (the $|\mathcal{N}_i|$'s are counted as real numbers, for simplicity). However, each iteration of RHA requires only transmission of a *single* message, consisting of exactly *one* real number, by the initiating node, in Step 5. Also notice that RHA fully exploits the broadcast nature of wireless medium, allowing everyone that hears the message to use it for revising their local variables, in Step 6. Therefore, RHA avoids issues D6 and D5 with overlapping iterations and wasted receptions. Furthermore, as RHA operates asynchronously and calculates the average directly, it circumvents issues D1–D4 with costly discretization, clock synchronization, forced transmissions, and computing intermediate quantities. To show that it overcomes issues D8 and D9 with steady-state errors and convergence guarantees, consider a quadratic Lyapunov function candidate $V : \mathbb{R}^L \to \mathbb{R}$, defined as

$$V(\mathbf{x}(k)) = \sum_{\{i,j\} \in \mathcal{E}} c_{\{i,j\}} (x_{\{i,j\}}(k) - x^*)^2. \tag{2.16}$$

Clearly, V in (2.16) is positive definite with respect to $(x^*, x^*, \ldots, x^*) \in \mathbb{R}^L$, and the condition

$$\lim_{k \to \infty} V(\mathbf{x}(k)) = 0 \tag{2.17}$$

implies (2.11) and thus (2.10). The following lemma shows that $V(\mathbf{x}(k))$ is always non-increasing and quantifies its changes:

Lemma 2.1. *Consider the wireless network modeled in Section 2.2 and the use of RHA described in Algorithm 2.1. Then, for any sequence $(u(k))_{k=1}^{\infty}$, the* *sequence* $(V(\mathbf{x}(k)))_{k=0}^{\infty}$ *is non-increasing and satisfies*

$$V(\mathbf{x}(k)) - V(\mathbf{x}(k-1)) = -\sum_{j\in\mathcal{N}_{u(k)}} c_{\{u(k),j\}}(x_{\{u(k),j\}}(k-1) - \hat{x}_{u(k)}(k-1))^2, \quad \forall k \in \mathbb{P}. \qquad (2.18)$$

Proof. From (2.16) and the bottom of (2.6),

$$V(\mathbf{x}(k)) - V(\mathbf{x}(k-1)) = -\sum_{j\in\mathcal{N}_{u(k)}} c_{\{u(k),j\}}(-x^2_{\{u(k),j\}}(k) + 2x_{\{u(k),j\}}(k)x^* + x^2_{\{u(k),j\}}(k-1) - 2x_{\{u(k),j\}}(k-1)x^*), \quad \forall k \in \mathbb{P}.$$

Due to the top of (2.6), the second term $-\sum_{j\in\mathcal{N}_{u(k)}} 2c_{\{u(k),j\}}x_{\{u(k),j\}}(k)x^*$ cancels the fourth term $\sum_{j\in\mathcal{N}_{u(k)}} 2c_{\{u(k),j\}}x_{\{u(k),j\}}(k-1)x^*$. Moreover, note from (2.6) and (2.7) that $x_{\{u(k),j\}}(k) = \hat{x}_{u(k)}(k-1)$ $\forall j \in \mathcal{N}_{u(k)}$. Hence, $V(\mathbf{x}(k)) - V(\mathbf{x}(k-1)) = -\sum_{j\in\mathcal{N}_{u(k)}} c_{\{u(k),j\}}(\hat{x}^2_{u(k)}(k-1) - 2\hat{x}_{u(k)}(k-1)x_{\{u(k),j\}}(k) + x^2_{\{u(k),j\}}(k-1))$ $\forall k \in \mathbb{P}$. Due again to the top of (2.6), the second term

$$\sum_{j\in\mathcal{N}_{u(k)}} 2c_{\{u(k),j\}}\hat{x}_{u(k)}(k-1)x_{\{u(k),j\}}(k) = \sum_{j\in\mathcal{N}_{u(k)}} 2c_{\{u(k),j\}}\hat{x}_{u(k)}(k-1)x_{\{u(k),j\}}(k-1).$$

Thus, (2.18) holds. Since the right-hand side of (2.18) is nonpositive, $(V(\mathbf{x}(k)))_{k=0}^{\infty}$ is non-increasing. □

Lemma 2.1 says that $V(\mathbf{x}(k)) \le V(\mathbf{x}(k-1))$ $\forall k \in \mathbb{P}$. Since $V(\mathbf{x}(k)) \ge 0$ $\forall \mathbf{x}(k) \in \mathbb{R}^L$, this implies that $\lim_{k\to\infty} V(\mathbf{x}(k))$ exists and is nonnegative. The following theorem asserts that this limit is almost surely zero, so that RHA is almost surely asymptotically convergent to x^*:

Theorem 2.1. *Consider the wireless network modeled in Section 2.2 and the use of RHA described in Algorithm 2.1. Then, with probability* 1, (2.17)*,* (2.11)*, and* (2.10) *hold.*

Proof. By associating the line graph of $\mathcal{G}$ with the graph in [20], RHA may be viewed as a special case of the algorithm (1) in [20]. Note from (2.6) and (2.14) that the diagonal entries of $\mathbf{A}_i$ $\forall i \in \mathcal{V}$ are positive, from (2.9) that $\mathrm{P}\{\mathbf{A}_{u(k)} = \mathbf{A}_i\} = \frac{1}{N}$ $\forall k \in \mathbb{P}$ $\forall i \in \mathcal{V}$, and from the connectedness of $\mathcal{G}$ that its line graph is connected. Thus, by Corollary 3.2 of [20], with probability 1, $\exists \tilde{x} \in \mathbb{R}$ such that $\lim_{k\to\infty} x_{\{i,j\}}(k) = \tilde{x}$ $\forall \{i,j\} \in \mathcal{E}$. Due to (2.1), (2.12), and (2.13), $\tilde{x} = x^*$, i.e., (2.11) holds almost surely. Because of (2.16) and (2.7), so do (2.17) and (2.10). $\square$

As it follows from Theorem 2.1 and the above, RHA solves the distributed averaging problem, while eliminating deficiencies D1–D9 facing the existing schemes except for D7, on lack of control. Lemma 2.1 above also says that V in (2.16) is a *common* quadratic Lyapunov function for the linear switched system (2.8). This V will be used next to introduce control and remove D7.

2.5 Controlled Hopwise Averaging

2.5.1 Motivation for Feedback Iteration Control

RHA operates by executing (2.6) or (2.8) according to $(u(k))_{k=1}^{\infty}$. Although, by Theorem 2.1, almost any $(u(k))_{k=1}^{\infty}$ can drive all the $\hat{x}_i(k)$'s in (2.7) to any neighborhood of x^*, certain sequences require fewer iterations (and, hence, fewer real-number transmissions) to do so than others, yielding better bandwidth/energy efficiency. To see this, consider the following proposition:

Proposition 2.1. *The matrices* $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N$ *in* (2.8) *are idempotent, i.e.,* $\mathbf{A}_i^2 = \mathbf{A}_i$ $\forall i \in \mathcal{V}$*. Moreover,* $\mathbf{A}_i$ *and* $\mathbf{A}_j$ *are commutative whenever* $\{i,j\} \notin \mathcal{E}$*, i.e.,* $\mathbf{A}_i\mathbf{A}_j = \mathbf{A}_j\mathbf{A}_i$ $\forall i,j \in \mathcal{V}$*,* $\{i,j\} \notin \mathcal{E}$*.*

Proof. Notice from (2.6) and (2.8) that for any $i \in \mathcal{V}$, if $\mathbf{x}(k) = \mathbf{A}_i\mathbf{x}(k-1)$, then $x_{\{i,j\}}(k)\ \forall j \in \mathcal{N}_i$ are set equal to the same convex combination of $x_{\{i,j\}}(k-1)$ $\forall j \in \mathcal{N}_i$, and $x_{\{p,q\}}(k) = x_{\{p,q\}}(k-1)\ \forall \{p,q\} \in \mathcal{E} - \cup_{j\in\mathcal{N}_i}\{\{i,j\}\}$. Thus, $\mathbf{A}_i\mathbf{x}(k) = \mathbf{x}(k)$, so that $\mathbf{A}_i^2 = \mathbf{A}_i$. Moreover, for any $i, j \in \mathcal{V}$ with $\{i,j\} \notin \mathcal{E}$, because $\{\{i,\ell\} : \ell \in \mathcal{N}_i\} \cap \{\{j,\ell\} : \ell \in \mathcal{N}_j\} = \emptyset$, $\mathbf{A}_i\mathbf{A}_j = \mathbf{A}_j\mathbf{A}_i$. □

The idempotence and partial commutativity of $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N$ from Proposition 2.1, together with the fact that the switched system (2.8) may be stated as $\mathbf{x}(k) = \mathbf{A}_{u(k)}\mathbf{A}_{u(k-1)}\cdots\mathbf{A}_{u(1)}\mathbf{x}(0)\ \forall k \in \mathbb{P}$, imply that for a given $(u(k))_{k=1}^{\infty}$, the event $\mathbf{x}(k) = \mathbf{x}(k-1)$ can occur for quite a few k's, each of which signifies a wasted iteration. Furthermore, if the event $\mathbf{x}(k) = \mathbf{x}(k-1)$ does occur for at least one k, then by deleting from $(u(k))_{k=1}^{\infty}$ some of its elements that correspond to the wasted iterations, we obtain a new sequence $(u'(k))_{k=1}^{\infty}$ that is more efficient. To illustrate these two points, consider, for instance, a 5-node cycle graph with $\mathcal{V} = \{1,2,3,4,5\}$ and $\mathcal{E} = \{\{1,2\},\{2,3\},\{3,4\},\{4,5\},\{5,1\}\}$. Notice that if $(u(k))_{k=1}^{\infty} = (1, \underline{1}, 3, 4, \underline{1}, 2, \underline{4}, 5, \underline{2}, \underline{5}, \ldots)$, then as many as 5 out of the first 10 iterations—namely, those underlined elements—are wasted. By deleting these underlined elements and keeping the rest intact, we obtain a new sequence $(u'(k))_{k=1}^{\infty} = (1,3,4,2,5,\ldots)$ that is 5 real-number transmissions more efficient than $(u(k))_{k=1}^{\infty}$.

The preceding analysis shows that RHA is prone to wasteful iterations, which is a primary reason why certain sequences are more efficient than others. RHA, however, makes no attempt to distinguish the sequences, as it lets every possible $(u(k))_{k=1}^{\infty}$ be equiprobable, via (2.9). In other words, it does not try to *control* how the asynchronous iterations occur and, thus, suffers from D7.

Remark 2.3. Wasteful iterations incurred by idempotent and partially commutative operations are not an attribute unique to RHA, but one that is shared by Pairwise Averaging [72], Anti-Entropy Aggregation [26, 39], Randomized Gossip Algorithm [8], and Distributed Random Grouping [13] (indeed, the examples provided in D7 against the latter two algorithms were created from this attribute). What is different is that in this chapter, we view the attribute as a limitation and find ways to overcome it, whereas in [8, 13, 26, 39, 72], the attribute was not viewed as such. ■

One way to control the iterations, alluded to in Remark 2.1, is to replace (2.9) with a general distribution $\mathrm{P}\{u(k) = i\} = p_i \ \forall k \in \mathbb{P} \ \forall i \in \mathcal{V}$ and then choose the p_i's to maximize efficiency, before any averaging task begins. This approach, however, has an inherent shortcoming: because the p_i's are optimized once-and-for-all, they are constant and do not adapt to $\mathbf{x}(k)$ during runtime. Hence, optimal or not, the p_i's almost surely would produce inefficient, wasteful $(u(k))_{k=1}^{\infty}$. The fact that the nodes do not adjust the p_i's based on information they pick up during runtime also suggests that this way of controlling the iterations may be considered *open loop*.

The aforementioned shortcoming of open-loop iteration control raises the question of whether it is possible to introduce some form of *closed-loop* iteration control as a means to generate efficient, non-wasteful $(u(k))_{k=1}^{\infty}$. Obviously, to carry out closed-loop iteration control, feedback is needed. Due to the distributed nature of the network, however, feedback may be expensive to acquire: if an algorithm demands that the feedback used by a node be a function of state variables maintained by other nodes, then additional communications are necessary to implement the feedback. Such communications can produce plenty of real-number transmissions, which must all count toward the total real-number transmissions, when evaluating the algorithm's bandwidth/energy efficiency. Thus, in the design of feedback algorithms, the cost of "closing the loop" cannot be overlooked.

In this section, we first describe an approach to closed-loop iteration control, which leads to highly efficient and surely non-wasteful $(u(k))_{k=1}^{\infty}$ at *zero* feedback cost. Based on this approach, we then present and analyze two modified versions of RHA: an ideal version and a practical one.

2.5.2 Approach to Feedback Iteration Control

Note that with RHA, $(u(k))_{k=1}^{\infty}$ is undefined at the moment an averaging task begins and is gradually defined, one element per iteration, as time elapses, i.e., when a node $i \in \mathcal{V}$ initiates an iteration $k \in \mathbb{P}$, the element $u(k)$ becomes defined and is given by $u(k) = i$. Thus, by controlling *when* to initiate an iteration, the nodes may jointly shape the value of $(u(k))_{k=1}^{\infty}$. With RHA, this opportunity to shape $(u(k))_{k=1}^{\infty}$ is not utilized, as the nodes simply randomly and equiprobably decide when to initiate an iteration. To exploit the opportunity, suppose henceforth that the nodes wish to control when to initiate an iteration using some form of *feedback*. The questions are:

Q1. What feedback to use, so that the corresponding feedback cost is minimal?

Q2. How to control, so that the resulting $(u(k))_{k=1}^{\infty}$ is highly efficient?

Q3. How to control, so that the resulting $(u(k))_{k=1}^{\infty}$ is surely non-wasteful?

To answer questions Q1–Q3, we first show that RHA, along with the common quadratic Lyapunov function V of (2.16), exhibits the following features:

F1. Although the nodes never know the value of V, every one of them at any time knows by how much the value would drop if it suddenly initiates an iteration.

F2. The faster $(u(k))_{k=1}^{\infty}$ makes the value of V drop to zero, the more efficient it is.

F3. If the value of V does not drop after an iteration, then the iteration is wasted, causing $(u(k))_{k=1}^{\infty}$ to be wasteful. The converse is also true.

The first part of feature F1 can be seen by noting that $V(\mathbf{x}(k))$ in (2.16) depends on $c_{\{i,j\}}$ $\forall\{i,j\} \in \mathcal{E}$, $x_{\{i,j\}}(k)$ $\forall\{i,j\} \in \mathcal{E}$, and x^*, whereas each node $i \in \mathcal{V}$ only knows $c_{\{i,j\}}$ $\forall j \in \mathcal{N}_i$ and $x_{\{i,j\}}(k)$ $\forall j \in \mathcal{N}_i$. To see the second part, suppose a node $i \in \mathcal{V}$ initiates an iteration $k \in \mathbb{P}$ at some time instant t, so that $u(k) = i$ by definition. Observe from Lemma 2.1 that whoever node $u(k)$ is, upon completing this iteration, the value of V would drop from $V(\mathbf{x}(k-1))$ to $V(\mathbf{x}(k))$ by an amount equal to the right-hand side of (2.18). To compactly represent this drop, for each $i \in \mathcal{V}$ let $\Delta V_i : \mathbb{R}^L \to \mathbb{R}$ be a positive semidefinite quadratic function, defined as

$$\Delta V_i(\mathbf{x}(k)) = \sum_{j \in \mathcal{N}_i} c_{\{i,j\}}(x_{\{i,j\}}(k) - \hat{x}_i(k))^2, \quad \forall k \in \mathbb{N}, \tag{2.19}$$

where $\hat{x}_i(k)$ is as in (2.7). Then, with (2.19), (2.18) may be written as

$$V(\mathbf{x}(k)) - V(\mathbf{x}(k-1)) = -\Delta V_{u(k)}(\mathbf{x}(k-1)), \quad \forall k \in \mathbb{P}, \tag{2.20}$$

where $\Delta V_{u(k)}(\mathbf{x}(k-1))$ in (2.20) represents the amount of drop, i.e.,

$$\Delta V_{u(k)}(\mathbf{x}(k-1)) = \sum_{j \in \mathcal{N}_{u(k)}} c_{\{u(k),j\}}(x_{\{u(k),j\}}(k-1) - \hat{x}_{u(k)}(k-1))^2, \quad \forall k \in \mathbb{P}. \tag{2.21}$$

Notice that $\Delta V_{u(k)}(\mathbf{x}(k-1))$ in (2.21) depends on parameters and variables maintained by node $u(k)$, whose values are known to node $u(k)$ prior to iteration

k at time t. Therefore, before initiating this iteration at time t, node $u(k)$ already knows that the value of V would drop by $\Delta V_{u(k)}(\mathbf{x}(k-1))$. Since t, k, and $u(k)$ are arbitrary, this means that every node $i \in \mathcal{V}$ at any time knows by how much the value of V would drop if it suddenly initiates an iteration (i.e., by $\Delta V_i(\mathbf{x}(\cdot))$). This establishes feature F1. To show feature F2, recall that: (i) $V(\mathbf{x}(k))$ in (2.16) is a measure of the deviation of the $x_{\{i,j\}}(k)$'s from x^*; (ii) the $\hat{x}_i(k)$'s in (2.7) are convex combinations of the $x_{\{i,j\}}(k)$'s; (iii) bandwidth/energy efficiency is measured by the number of real-number transmissions needed for all the $\hat{x}_i(k)$'s to converge to a given neighborhood of x^*; and (iv) RHA in Algorithm 2.1 has a fixed, one real-number transmission per iteration. Hence, the faster $(u(k))_{k=1}^{\infty}$ drives $V(\mathbf{x}(k))$ to zero, the faster it drives the $x_{\{i,j\}}(k)$'s and $\hat{x}_i(k)$'s to x^* (due to (i) and (ii)), and the more efficient it is (due to (iii) and (iv)). Finally, to show feature F3, suppose $V(\mathbf{x}(k)) = V(\mathbf{x}(k-1))$ after an iteration $k \in \mathbb{P}$. Then, it follows from (2.20) that $\Delta V_{u(k)}(\mathbf{x}(k-1)) = 0$, from (2.21) that $x_{\{u(k),j\}}(k-1)\ \forall j \in \mathcal{N}_{u(k)}$ are equal, and from (2.6) that $\mathbf{x}(k) = \mathbf{x}(k-1)$. Thus, iteration k is wasted. The converse is also true, as $\mathbf{x}(k) = \mathbf{x}(k-1)$ implies $V(\mathbf{x}(k)) = V(\mathbf{x}(k-1))$.

Having demonstrated features F1–F3, we now use them to answer questions Q1–Q3. Feature F1 suggests that every node $i \in \mathcal{V}$ may use $\Delta V_i(\mathbf{x}(\cdot))$, which it always knows, as feedback to control, on its own, when to initiate an iteration. As the feedbacks $\Delta V_i(\mathbf{x}(\cdot))$'s are locally available and the control decisions are made locally, the resulting feedback control architecture is fully decentralized, requiring zero communication cost to realize. Therefore, an answer to question Q1 is:

A1. Each node $i \in \mathcal{V}$ uses $\Delta V_i(\mathbf{x}(\cdot))$ as feedback to control when to initiate an iteration.

Feature F2 suggests that, to produce highly efficient $(u(k))_{k=1}^{\infty}$, the nodes may focus on making the value of V drop significantly after each iteration, especially initially. In other words, they may focus on letting every iteration be initiated by a node i with a relatively large $\Delta V_i(\mathbf{x}(\cdot))$. With architecture A1, this may be accomplished if nodes with larger $\Delta V_i(\mathbf{x}(\cdot))$'s would rush to initiate, while nodes with smaller $\Delta V_i(\mathbf{x}(\cdot))$'s would wait longer. Hence, an answer to question Q2 is:

A2. The larger $\Delta V_i(\mathbf{x}(\cdot))$ is, the sooner node i initiates an iteration (i.e., the smaller $\Delta V_i(\mathbf{x}(\cdot))$ is, the longer node i waits).

Finally, feature F3 suggests that, to generate surely non-wasteful $(u(k))_{k=1}^{\infty}$, the value of V must strictly decrease after each iteration. With architecture A1, this can be achieved if nodes with zero $\Delta V_i(\mathbf{x}(\cdot))$'s would refrain from initiating an iteration. Thus, an answer to question Q3 is:

A3. Whenever $\Delta V_i(\mathbf{x}(\cdot)) = 0$, node i refrains from initiating an iteration.

Answers A1–A3 describe a greedy, decentralized approach to feedback iteration control, where potential drops $\Delta V_i(\mathbf{x}(\cdot))$'s in the value of V are used to drive the asynchronous iterations. This approach may be viewed as a greedy approach because the nodes seek to make the value of V drop as much as possible at each iteration, without considering the future. Because the nodes also seek to fully exploit the broadcast nature of every wireless transmission (a feature inherited from Steps 5 and 6 of RHA), this approach strives to "make the most" out of each iteration. Note that although Lyapunov functions have been used to analyze distributed averaging and consensus algorithms (e.g., in the form of a disagreement function [52] or a set-valued convex hull [40]), their use for *controlling* such algorithms has not been reported. Therefore, this approach represents a new way to apply Lyapunov stability theory.

2.5.3 Ideal Version

In this subsection, we use the aforementioned approach to create an ideal, modified version of RHA, which possesses strong convergence properties that motivate a practical version.

The above approach wants the nodes to try to be greedy. Thus, it is of interest to analyze an ideal scenario where, instead of just trying, the nodes actually succeed at being greedy, ensuring that every iteration $k \in \mathbb{P}$ is initiated by a node $i \in \mathcal{V}$ with the maximum $\Delta V_i(\mathbf{x}(k-1))$, i.e.,

$$u(k) \in \arg\max_{i \in \mathcal{V}} \Delta V_i(\mathbf{x}(k-1)), \quad \forall k \in \mathbb{P}, \tag{2.22}$$

so that $V(\mathbf{x}(k-1))$ drops maximally to $V(\mathbf{x}(k))$ for every $k \in \mathbb{P}$. Notice that (2.22) does not always uniquely determine $u(k)$: when multiple nodes have the same maximum, $u(k)$ may be any of these nodes. Although $u(k)$ can be made unique (e.g., by letting $u(k)$ be the minimum of $\arg\max_{i\in\mathcal{V}} \Delta V_i(\mathbf{x}(k-1))$), in the analysis below we will allow for arbitrary $u(k)$ satisfying (2.22). Also note that in the rare case where $\Delta V_i(\mathbf{x}(k^*-1)) = 0$ $\forall i \in \mathcal{V}$ for some $k^* \in \mathbb{P}$, due to (2.1), (2.12), (2.13), (2.19), and the connectedness of the graph $\mathcal{G}$, we have $x_{\{i,j\}}(k^*-1) = x^*$ $\forall \{i,j\} \in \mathcal{E}$ and $\hat{x}_i(k^*-1) = x^*$ $\forall i \in \mathcal{V}$, thereby solving the problem in finite time. Furthermore, due to A3, all the nodes would refrain from initiating iteration k^* (and beyond), thereby terminating the algorithm in finite time and causing $x_{\{i,j\}}(k)$ $\forall \{i,j\} \in \mathcal{E}$, $\hat{x}_i(k)$ $\forall i \in \mathcal{V}$, $u(k)$, and $V(\mathbf{x}(k))$

to be undefined $\forall k \geq k^*$. In the analysis below, however, we will allow the algorithm to keep executing according to (2.22), so that $x_{\{i,j\}}(k)$ $\forall \{i,j\} \in \mathcal{E}$, $\hat{x}_i(k)$ $\forall i \in \mathcal{V}$, $u(k)$, and $V(\mathbf{x}(k))$ are defined $\forall k$.

Equation (2.22), together with (2.6), (2.7), (2.14), (2.15), and (2.19), defines a networked dynamical system that switches among N different dynamics, depending on where the state is in the state space, i.e., if $\mathbf{x}(k-1)$ is such that $\Delta V_i(\mathbf{x}(k-1)) > \Delta V_j(\mathbf{x}(k-1))$ $\forall j \in \mathcal{V} - \{i\}$, then $\mathbf{x}(k) = \mathbf{A}_i \mathbf{x}(k-1)$. This system may be expressed in the form of an algorithm—which we refer to as *Ideal Controlled Hopwise Averaging* (ICHA)—as follows:

Algorithm 2.2 (Ideal Controlled Hopwise Averaging)**.**

Initialization:

1. Each node $i \in \mathcal{V}$ transmits $|\mathcal{N}_i|$ and y_i to every node $j \in \mathcal{N}_i$.
2. Each node $i \in \mathcal{V}$ creates variables $x_{ij} \in \mathbb{R}$ $\forall j \in \mathcal{N}_i$, $\hat{x}_i \in \mathbb{R}$, and $\Delta V_i \in [0, \infty)$ and initializes them sequentially:
   $x_{ij} \leftarrow \frac{\frac{y_i}{|\mathcal{N}_i|} + \frac{y_j}{|\mathcal{N}_j|}}{c_{\{i,j\}}}, \quad \forall j \in \mathcal{N}_i,$
   $\hat{x}_i \leftarrow \frac{\sum_{j \in \mathcal{N}_i} c_{\{i,j\}} x_{ij}}{\sum_{j \in \mathcal{N}_i} c_{\{i,j\}}},$
   $\Delta V_i \leftarrow \sum_{j \in \mathcal{N}_i} c_{\{i,j\}} (x_{ij} - \hat{x}_i)^2.$

Operation: At each iteration:

3. Let $i \in \arg\max_{j \in \mathcal{V}} \Delta V_j$.
4. Node i updates x_{ij} $\forall j \in \mathcal{N}_i$ and ΔV_i sequentially:
   $x_{ij} \leftarrow \hat{x}_i, \quad \forall j \in \mathcal{N}_i,$
   $\Delta V_i \leftarrow 0.$
5. Node i transmits $\hat{x}_i$ to every node $j \in \mathcal{N}_i$.
6. Each node $j \in \mathcal{N}_i$ updates x_{ji}, $\hat{x}_j$, and ΔV_j sequentially:
   $x_{ji} \leftarrow \hat{x}_i,$

$\hat{x}_j \leftarrow \frac{\sum_{\ell\in\mathcal{N}_j} c_{\{j,\ell\}} x_{j\ell}}{\sum_{\ell\in\mathcal{N}_j} c_{\{j,\ell\}}}$,
$\Delta V_j \leftarrow \sum_{\ell\in\mathcal{N}_j} c_{\{j,\ell\}}(x_{j\ell} - \hat{x}_j)^2$. ■

Algorithm 2.2, or ICHA, is identical to RHA in Algorithm 2.1 except that each node i also maintains ΔV_i, in Steps 2, 4, and 6, and that each iteration is initiated by a node i experiencing the maximum ΔV_i, in Step 3. Note that "$\Delta V_i \leftarrow 0$" in Step 4 is equivalent to "$\Delta V_i \leftarrow \sum_{j\in\mathcal{N}_i} c_{\{i,j\}}(x_{ij} - \hat{x}_i)^2$" since x_{ij} $\forall j \in \mathcal{N}_i$ and $\hat{x}_i$ are equal at that point. The fact that ΔV_i goes from being the maximum to zero whenever node i initiates an iteration also suggests that it may be a while before ΔV_i becomes the maximum again, causing node i to initiate another iteration.

The convergence properties of ICHA on general networks are characterized in the following theorem, in which $\mathbf{1}_n \in \mathbb{R}^n$ and $\hat{\mathbf{x}}(k) \in \mathbb{R}^N$ denote, respectively, the vectors obtained by stacking n 1's and the N $\hat{x}_i(k)$'s:

Theorem 2.2. *Consider the wireless network modeled in Section 2.2 and the use of ICHA described in Algorithm 2.2. Then,*

$$V(\mathbf{x}(k)) \leq (1 - \tfrac{1}{\gamma})V(\mathbf{x}(k-1)), \quad \forall k \in \mathbb{P}, \tag{2.23}$$

$$\|\mathbf{x}(k) - x^*\mathbf{1}_L\| \leq \sqrt{\tfrac{V(\mathbf{x}(0))\max_{i\in\mathcal{V}}|\mathcal{N}_i|}{2}}(1 - \tfrac{1}{\gamma})^{k/2}, \quad \forall k \in \mathbb{N}, \tag{2.24}$$

$$\|\hat{\mathbf{x}}(k) - x^*\mathbf{1}_N\| \leq \sqrt{\tfrac{2V(\mathbf{x}(0))\max_{i\in\mathcal{V}}|\mathcal{N}_i|}{\min_{i\in\mathcal{V}}|\mathcal{N}_i| + \max_{i\in\mathcal{V}}|\mathcal{N}_i|}}(1 - \tfrac{1}{\gamma})^{k/2}, \quad \forall k \in \mathbb{N}, \tag{2.25}$$

where $\gamma \in [\frac{N}{2} + 1, N^3 - 2N^2 + \frac{N}{2} + 1]$ *is given by*

$$\gamma = \frac{N}{2} + \alpha + \frac{(N^2 - \beta)(3(N-1) - D)(D+1)}{2N}, \tag{2.26}$$

and where $\alpha = \max_{\{i,j\}\in\mathcal{E}} \frac{b_i + b_j}{c_{\{i,j\}}} \in [1, \frac{N^2-2N+2}{2}]$, $\beta = \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i\cup\{i\}} b_i b_j \in [N + \frac{L}{2}(1 + \frac{1}{N-1})^2, N^2]$, $b_i = \frac{1}{2}\sum_{j\in\mathcal{N}_i} c_{\{i,j\}}$ $\forall i \in \mathcal{V}$, *and* D *is the network diameter.*

Proof. See Appendix A.1. $\square$

Theorem 2.2 says that ICHA is exponentially convergent on any network, ensuring that $V(\mathbf{x}(k))$, $\|\mathbf{x}(k) - x^*\mathbf{1}_L\|$, and $\|\hat{\mathbf{x}}(k) - x^*\mathbf{1}_N\|$ all go to zero exponentially fast, at a rate that is no worse than $1 - \frac{1}{\gamma}$ or $(1 - \frac{1}{\gamma})^{1/2}$, so that γ in (2.26) represents a bound on the convergence rate. It also says that the bound γ is between $\Omega(N)$ and $O(N^3)$ and depends only on N, D, and the $|\mathcal{N}_i|$'s, making it easy to compute. The following corollary lists the bound γ for a number of common graphs:

Corollary 2.1. *The constant* γ *in* (2.26) *becomes:*

G1. $\gamma = N^3 - 4N^2 + \frac{9}{2}N + \frac{5}{4}$ *for a path graph with* $N \geq 5$*,*

G2. $\gamma = \frac{5}{8}N^3 - \frac{15}{8}N^2 - \frac{1}{8}N + \frac{31}{8}$ *if* N *is odd and* $\gamma = \frac{5}{8}N^3 - \frac{11}{8}N^2 - \frac{5}{2}N + \frac{13}{2}$ *if* N *is even for a cycle graph,*

G3. $\gamma = \frac{N}{2} + K + \frac{(N-K-1)(3(N-1)-D)(D+1)}{2}$ *for a* K*-regular graph with* $K \geq 2$*,*

G4. $\gamma = \frac{3}{2}N - 1$ *for a complete graph.*

Proof. For a path graph with $N \geq 5$, $\alpha = \frac{9}{4}$, $\beta = 3N - 1$, and $D = N - 1$. For a cycle graph, $\alpha = 2$, $\beta = 3N$, $D = \frac{N-1}{2}$ if N is odd, and $D = \frac{N}{2}$ if N is even. For a K-regular graph with $K \geq 2$, $\alpha = K$ and $\beta = N(K+1)$. For a complete graph, $\alpha = N - 1$ and $\beta = N^2$. Hence, G1–G4 hold. $\square$

Each bound γ in Corollary 2.1 is obtained by specializing (2.26) for arbitrary graphs to a specific one. Conceivably, tighter bounds may be obtained by working with each of these graphs individually, exploiting their particular structure. Theorem 2.3 below shows that this is indeed the case with path and cycle graphs (6 and 15 times tighter, respectively), besides providing additional bounds for regular and strongly regular graphs:

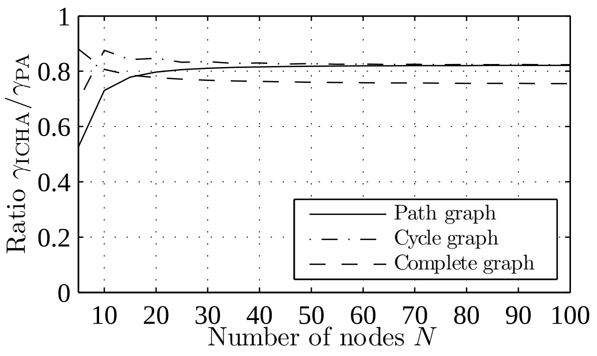


Figure 2.1: Comparison between the stochastic convergence rate $1-\frac{1}{\gamma_{\text{PA}}}$ of PA and the deterministic bound $1-\frac{1}{\gamma_{\text{ICHA}}}$ on convergence rate of ICHA for path, cycle, and complete graphs.

Theorem 2.3. *Consider the wireless network modeled in Section 2.2 and the use of ICHA described in Algorithm 2.2. Then,* (2.23)–(2.25) *hold with:*

S1. $\gamma = \frac{N^3}{6} - \frac{13}{6}N + 3$ *for a path graph with* $N \geq 4$,

S2. $\gamma = \frac{N^3}{24} + \frac{7}{12}N - 2 + \frac{11}{8N}$ *if* N *is odd and* $\gamma = \frac{N^3}{24} + \frac{5}{6}N - 3 + \frac{4}{N}$ *if* N *is even for a cycle graph,*

S3. $\gamma = \frac{N}{2} + K + \frac{KD(D+1)(N-K-1)}{2}$ *for a* K*-regular graph with* $K \geq 2$,

S4. $\gamma = \frac{N}{2} + K + \frac{K(\mu+2)(N-K-1)}{\mu}$ *for a* (N, K, λ, μ)*-strongly regular graph with* $\mu \geq 1$.

Proof. See Appendix A.2. □

Recently, [20] studied, among other things, the convergence rate of Pairwise Averaging (PA) [72]. The results in [20] are different from those above in three notable ways: first, the convergence rate of PA is defined in [20] as the decay rate of the *expected value* of a Lyapunov-like function $d(k)$. Although this stochastic measure captures the average behavior of PA, it offers little guarantee on the decay rate of each realization $(d(k))_{k=0}^{\infty}$. In contrast, the bounds

γ on convergence rate of ICHA above are deterministic, providing guarantees on the decay rate of $(V(\mathbf{x}(k)))_{k=0}^{\infty}$. Second, even if the first difference is disregarded, the bounds of ICHA are still roughly 20% better than the convergence rate of PA for a few common graphs. To justify this claim, let $1 - \frac{1}{\gamma_{\text{PA}}}$ denote the convergence rate of PA. Since PA requires two real-number transmissions per iteration while ICHA requires only one, to enable a fair comparison we introduce a two-iteration bound γ_{ICHA} for ICHA, defined as $\gamma_{\text{ICHA}} = \frac{\gamma^2}{2\gamma-1}$ so that $1 - \frac{1}{\gamma_{\text{ICHA}}} = (1 - \frac{1}{\gamma})^2$. Figure 2.1 plots the ratio $\frac{\gamma_{\text{ICHA}}}{\gamma_{\text{PA}}}$ versus N for path, cycle, and complete graphs, where γ_{PA} is computed according to [20], while γ_{ICHA} is computed using γ in S1, S2, and G4. Observe that for $N > 50$, γ_{ICHA} is 18% smaller than γ_{PA} for path and cycle graphs, and 25% so for complete graphs. The latter can also be shown analytically: since $\gamma_{\text{PA}} = N - 1$ and $\gamma_{\text{ICHA}} = \frac{(\frac{3}{2}N-1)^2}{2(\frac{3}{2}N-1)-1}$, $\lim_{N\to\infty} \frac{\gamma_{\text{ICHA}}}{\gamma_{\text{PA}}} = \frac{3}{4}$. This justifies the claim. Finally, unlike γ and γ_{ICHA}, γ_{PA} in general cannot be expressed in a form that explicitly reveals its dependence on the graph invariants. Indeed, it generally can only be computed by numerically finding the spectral radius of an invariant subspace of an N^2-by-N^2 matrix, which may be prohibitive for large N.

2.5.4 Practical Version

The strong convergence properties of ICHA suggest that its greedy behavior may be worthy of emulating. In this subsection, we derive a practical algorithm that closely mimics such behavior.

Reconsider the system (2.6), (2.7), (2.14), (2.15) and suppose this system evolves in a discrete event fashion, according to the following description: associated with the system is *time*, which is real-valued, nonnegative, and denoted as $t \in [0, \infty)$, where $t = 0$ represents the time instant at which the nodes have observed the y_i's but have yet to execute an iteration. In addition, associated with each node $i \in \mathcal{V}$ is an *event*, which is scheduled to occur at time $\tau_i \in (0, \infty]$ and is marked by node i initiating an iteration, where $\tau_i = \infty$ means the event will not occur. Each event time τ_i is a *variable*, which is initialized at time $t = 0$ to $\tau_i(0)$, is updated only at each iteration $k \in \mathbb{P}$ from $\tau_i(k-1)$ to $\tau_i(k)$, and is no less than t at any time t, so that no event is scheduled to occur in the past. Starting from $t = 0$, time advances to $t = \min_{i \in \mathcal{V}} \tau_i(0)$, at which an event, marked by node $u(1) \in \arg\min_{i \in \mathcal{V}} \tau_i(0)$ initiating iteration 1, occurs, during which $\tau_i(1)$ $\forall i \in \mathcal{V}$ are determined. Time then advances to $t = \min_{i \in \mathcal{V}} \tau_i(1)$, at which a subsequent event, marked by node $u(2) \in \arg\min_{i \in \mathcal{V}} \tau_i(1)$ initiating iteration 2, occurs, during which $\tau_i(2)$ $\forall i \in \mathcal{V}$ are determined. In the same way, time continues to advance toward infinity, while events continue to occur one after another, except if $\tau_i(k) = \infty$ $\forall i \in \mathcal{V}$ for some $k \in \mathbb{N}$, for which the system terminates.

Having described how the system evolves, we now specify how $\tau_i(k)$ $\forall k \in \mathbb{N}$ $\forall i \in \mathcal{V}$ are recursively determined. First, consider the time instant $t = 0$, at which $\tau_i(0)$ $\forall i \in \mathcal{V}$ need to be determined. To behave greedily, nodes with the maximum $\Delta V_i(\mathbf{x}(0))$'s should have the minimum $\tau_i(0)$'s. This may be accomplished by letting

$$\tau_i(0) = \Phi(\Delta V_i(\mathbf{x}(0))), \quad \forall i \in \mathcal{V}, \tag{2.27}$$

where $\Phi : [0, \infty) \to (0, \infty]$ is a continuous and strictly decreasing function satisfying $\lim_{v \to 0} \Phi(v) = \infty$ and $\Phi(0) = \infty$. Although, mathematically, (2.27) ensures that $V(\mathbf{x}(0))$ drops maximally to $V(\mathbf{x}(1))$, in reality it is possible that multiple nodes have the same minimum $\tau_i(0)$'s, leading to wireless collisions.

To address this issue, we insert a little randomness into (2.27), rewriting it as

$$\tau_i(0) = \Phi(\Delta V_i(\mathbf{x}(0))) + \varepsilon(\Delta V_i(\mathbf{x}(0))) \cdot \text{rand}(), \quad \forall i \in \mathcal{V}, \tag{2.28}$$

where $\varepsilon : [0, \infty) \to (0, \infty)$ is a continuous function meant to take on small positive values and each call to rand() returns a uniformly distributed random number in $(0, 1)$. With (2.28), with high probability iteration 1 is initiated by a node i with the maximum, or a near-maximum, $\Delta V_i(\mathbf{x}(0))$.

Next, pick any $k \in \mathbb{P}$ and consider the time instant $t = \min_{i\in\mathcal{V}} \tau_i(k-1)$, at which node $u(k) \in \arg\min_{i\in\mathcal{V}} \tau_i(k-1)$ initiates iteration k, during which $\tau_i(k)$ $\forall i \in \mathcal{V}$ need to be determined. Again, to be greedy, nodes with the maximum $\Delta V_i(\mathbf{x}(k))$'s should have the minimum $\tau_i(k)$'s. At first glance, this may be approximately accomplished following ideas from (2.28), i.e., by letting

$$\tau_i(k) = \Phi(\Delta V_i(\mathbf{x}(k))) + \varepsilon(\Delta V_i(\mathbf{x}(k))) \cdot \text{rand}(), \quad \forall i \in \mathcal{V}. \tag{2.29}$$

However, with (2.29), it is possible that $\tau_i(k)$ turns out to be smaller than t, causing an event to be scheduled in the past. Moreover, nodes who are two or more hops away from node $u(k)$ are unaware of the ongoing iteration k and, thus, are unable to perform an update. Fortunately, these issues may be overcome by slightly modifying (2.29) as follows:

$$\tau_i(k) = \begin{cases} \max\{\Phi(\Delta V_i(\mathbf{x}(k))), t\} + \varepsilon(\Delta V_i(\mathbf{x}(k))) \cdot \text{rand}(), & \text{if } i \in \mathcal{N}_{u(k)} \cup \{u(k)\}, \\ \tau_i(k-1), & \text{otherwise,} \end{cases} \quad \forall i \in \mathcal{V}. \tag{2.30}$$

Using (2.28) and (2.30) and by induction on $k' \in \mathbb{P}$, it can be shown that $\tau_i(k')$ satisfies

$$\max\{\Phi(\Delta V_i(\mathbf{x}(k'))), t'\} \le \tau_i(k') \le \max\{\Phi(\Delta V_i(\mathbf{x}(k'))), t'\} + \varepsilon(\Delta V_i(\mathbf{x}(k'))), \quad \forall k' \in \mathbb{P},\ \forall i \in \mathcal{V},$$

where $t' = \min_{j\in\mathcal{V}} \tau_j(k'-1)$. Hence, with (2.30), it is highly probable that iteration $k+1$ is initiated by a node i with the maximum or a near-maximum $\Delta V_i(\mathbf{x}(k))$. It follows that with (2.28) and (2.30), the nodes closely mimic the greedy behavior of ICHA. Note that (2.28) and (2.30) represent a *feedback iteration controller*, which uses architecture A1 and follows the spirit of A2 (since Φ is strictly decreasing and ε is small) and A3 (since $\Phi(0)=\infty$). Also, Φ and ε represent the *controller parameters*, which may be selected based on practical wireless networking considerations (e.g., all else being equal, $\Phi(v)=\frac{1}{v}$ and $\varepsilon(v)=0.001$ yield faster convergence time than $\Phi(v)=\frac{10}{v}$ and $\varepsilon(v)=0.01$ but higher collision probability).

The above description defines a discrete event system, which can be realized via a distributed asynchronous algorithm, referred to as *Controlled Hopwise Averaging* (CHA) and stated as follows:

Algorithm 2.3 (Controlled Hopwise Averaging)**.**

Initialization:

1. Let time $t=0$.
2. Each node $i\in\mathcal{V}$ transmits $|\mathcal{N}_i|$ and y_i to every node $j\in\mathcal{N}_i$.
3. Each node $i\in\mathcal{V}$ creates variables $x_{ij}\in\mathbb{R}\ \forall j\in\mathcal{N}_i$, $\hat{x}_i\in\mathbb{R}$, $\Delta V_i\in[0,\infty)$, and $\tau_i\in(0,\infty]$ and initializes them sequentially:
   $x_{ij} \leftarrow \frac{\frac{y_i}{|\mathcal{N}_i|}+\frac{y_j}{|\mathcal{N}_j|}}{c_{\{i,j\}}}, \quad \forall j\in\mathcal{N}_i,$
   $\hat{x}_i \leftarrow \frac{\sum_{j\in\mathcal{N}_i} c_{\{i,j\}}x_{ij}}{\sum_{j\in\mathcal{N}_i} c_{\{i,j\}}},$
   $\Delta V_i \leftarrow \sum_{j\in\mathcal{N}_i} c_{\{i,j\}}(x_{ij}-\hat{x}_i)^2,$
   $\tau_i \leftarrow \Phi(\Delta V_i)+\varepsilon(\Delta V_i)\cdot \mathrm{rand}().$

Operation: At each iteration:

4. Let $t=\min_{j\in\mathcal{V}}\tau_j$ and $i\in\arg\min_{j\in\mathcal{V}}\tau_j$.

5. Node i updates $x_{ij}\ \forall j \in \mathcal{N}_i$, ΔV_i, and τ_i sequentially:

   $x_{ij} \leftarrow \hat{x}_i, \quad \forall j \in \mathcal{N}_i,$

   $\Delta V_i \leftarrow 0,$

   $\tau_i \leftarrow \infty.$

6. Node i transmits $\hat{x}_i$ to every node $j \in \mathcal{N}_i$.
7. Each node $j \in \mathcal{N}_i$ updates x_{ji}, $\hat{x}_j$, ΔV_j, and τ_j sequentially:

   $x_{ji} \leftarrow \hat{x}_i,$

   $\hat{x}_j \leftarrow \frac{\sum_{\ell \in \mathcal{N}_j} c_{\{j,\ell\}} x_{j\ell}}{\sum_{\ell \in \mathcal{N}_j} c_{\{j,\ell\}}},$

   $\Delta V_j \leftarrow \sum_{\ell \in \mathcal{N}_j} c_{\{j,\ell\}} (x_{j\ell} - \hat{x}_j)^2,$

   $\tau_j \leftarrow \max\{\Phi(\Delta V_j), t\} + \varepsilon(\Delta V_j) \cdot \text{rand}().$ ■

Algorithm 2.3, or CHA, is similar to ICHA in Algorithm 2.2 except that each node i maintains an additional variable τ_i, in Steps 3, 5, and 7, and that each iteration is initiated, in a discrete event fashion, by a node i having the minimum τ_i, in Step 4. Note that "$\tau_i \leftarrow \infty$" in Step 5 is due to "$\Delta V_i \leftarrow 0$" and to $\Phi(0) = \infty$. Moreover, every step of CHA is implementable in a fully decentralized manner, making it a practical algorithm.

To analyze the behavior of CHA, recall that ε is meant to take on small positive values, creating just a little randomness so that the probability of wireless collisions is zero. For the purpose of analysis, we turn this feature off (i.e., set $\varepsilon(v) = 0\ \forall v \in [0, \infty)$) and let the symbol "$\in$" in Step 4 take care of the randomness (i.e., randomly pick an element i from the set $\arg\min_{j \in \mathcal{V}} \tau_j$ whenever it has multiple elements). We also allow Φ to be arbitrary (but satisfy the conditions stated when it was introduced). With this setup, the following convergence properties of CHA can be established:

Theorem 2.4. *Theorems 2.2 and 2.3, intended for ICHA described in Algorithm 2.2, hold verbatim for CHA described in Algorithm 2.3 with any* Φ *and with* ε *satisfying* $\varepsilon(v) = 0\ \forall v \in [0, \infty)$. *In addition,* $\lim_{k\to\infty} t(k) = \infty$ *and* $V(\mathbf{x}(k)) \leq (\gamma - 1)\Phi^{-1}(t(k))\ \forall k \in \mathbb{P}$*, where* $t(0) = 0$ *and* $t(k)$ *is the time instant at which iteration* k *occurs.*

Proof. See Appendix A.3. □

Theorem 2.4 characterizes the convergence of CHA in two senses: *iteration* and *time*. Iteration-wise, it says that CHA converges exponentially and shares the same bounds γ on convergence rate as ICHA, regardless of Φ. This result suggests that CHA does closely emulate ICHA. Time-wise, the theorem says that CHA converges asymptotically and perhaps exponentially, depending on Φ. For example, $\Phi(v) = \frac{1}{v}$ does not guarantee exponential convergence in time (since $\Phi^{-1}(v) = \frac{1}{v}$), but $\Phi(v) = W(\frac{1}{v})$, where W is the Lambert W function, does (since $\Phi^{-1}(v) = \frac{1}{v}e^{-v}$). Therefore, the controller parameter Φ may be used to shape the temporal convergence of CHA.

Remark 2.4. CHA has a limitation: it assumes no clock offsets among the nodes. Note, however, that although such offsets would cause CHA to deviate from its designed behavior, they would not render it "inoperable," i.e., $V(\mathbf{x}(k))$ would still strictly decrease after every iteration k, and the conservation (2.12) would still hold, so that the $x_{\{i,j\}}(k)$'s and $\hat{x}_i(k)$'s would still approach x^*.

2.6 Performance Comparison

In this section, we compare the performance of RHA and CHA with that of Pairwise Averaging (PA) [72], Consensus Propagation (CP) [38], Algorithm A2 (A2) of [36], and Distributed Random Grouping (DRG) [13] via extensive simulation on multi-hop wireless networks modeled by random geometric graphs. For completeness, PA, CP, A2, and DRG are stated below, in which $\mathcal{E}' = \{(i,j) \in \mathcal{V} \times \mathcal{V} : \{i,j\} \in \mathcal{E}\}$ denotes the set of $2L$ directed links:

Algorithm 2.4 (Pairwise Averaging [72])**.**

Initialization:

1. Each node $i \in \mathcal{V}$ creates a variable $\hat{x}_i \in \mathbb{R}$ and initializes it: $\hat{x}_i \leftarrow y_i$.

Operation: At each iteration:

2. A link, say, link $\{i,j\}$, is selected randomly and equiprobably out of the set $\mathcal{E}$ of L links. Node i transmits $\hat{x}_i$ to node j. Node j updates $\hat{x}_j$: $\hat{x}_j \leftarrow \frac{\hat{x}_i + \hat{x}_j}{2}$. Node j transmits $\hat{x}_j$ to node i. Node i updates $\hat{x}_i$: $\hat{x}_i \leftarrow \hat{x}_j$. ■

Algorithm 2.5 (Consensus Propagation [38])**.**

Initialization:

1. Each node $i \in \mathcal{V}$ creates variables $K_{ji} \geq 0\ \forall j \in \mathcal{N}_i$, $\mu_{ji} \in \mathbb{R}\ \forall j \in \mathcal{N}_i$, and $\hat{x}_i \in \mathbb{R}$ and initializes them sequentially: $K_{ji} \leftarrow 0\ \forall j \in \mathcal{N}_i$, $\mu_{ji} \leftarrow 0\ \forall j \in \mathcal{N}_i$, $\hat{x}_i \leftarrow y_i$.

Operation: At each iteration:

2. A directed link, say, link (i,j), is selected randomly and equiprobably out of the set $\mathcal{E}'$ of $2L$ directed links. Node i transmits $F_{ij} \triangleq \frac{1+\sum_{\ell \in \mathcal{N}_i, \ell \neq j} K_{\ell i}}{1+\frac{1}{\beta}(1+\sum_{\ell \in \mathcal{N}_i, \ell \neq j} K_{\ell i})}$ and $G_{ij} \triangleq \frac{y_i + \sum_{\ell \in \mathcal{N}_i, \ell \neq j} K_{\ell i}\mu_{\ell i}}{1+\sum_{\ell \in \mathcal{N}_i, \ell \neq j} K_{\ell i}}$ to node j. Node j updates K_{ij}, μ_{ij}, and $\hat{x}_j$ sequentially: $K_{ij} \leftarrow F_{ij}$, $\mu_{ij} \leftarrow G_{ij}$, $\hat{x}_j \leftarrow \frac{y_j + \sum_{\ell \in \mathcal{N}_j} K_{\ell j}\mu_{\ell j}}{1+\sum_{\ell \in \mathcal{N}_j} K_{\ell j}}$. ■

Algorithm 2.6 (Algorithm A2 [36])**.**

Initialization:

1. Each node $i \in \mathcal{V}$ creates variables $\delta_{ij} \in \mathbb{R}\ \forall j \in \mathcal{N}_i$ and $\hat{x}_i \in \mathbb{R}$ and initializes them sequentially: $\delta_{ij} \leftarrow 0\ \forall j \in \mathcal{N}_i$, $\hat{x}_i \leftarrow y_i$.

Operation: At each iteration:

2. A directed link, say, link (i, j), is selected randomly and equiprobably out of the set $\mathcal{E}'$ of $2L$ directed links. Node i transmits $\hat{x}_i$ to node j. Node j updates δ_{ji}: $\delta_{ji} \leftarrow \delta_{ji} + \phi(\hat{x}_i - \hat{x}_j)$. Node j transmits $\phi(\hat{x}_i - \hat{x}_j)$ to node i. Node i updates δ_{ij}: $\delta_{ij} \leftarrow \delta_{ij} - \phi(\hat{x}_i - \hat{x}_j)$. Each node $\ell \in \mathcal{V}$ updates $\hat{x}_\ell$: $\hat{x}_\ell \leftarrow \hat{x}_\ell + \frac{\gamma}{|\mathcal{N}_\ell|+1}((\sum_{m \in \mathcal{N}_\ell} \delta_{\ell m}) + y_\ell - \hat{x}_\ell)$. ■

Algorithm 2.7 (Distributed Random Grouping [13])**.**

Initialization:

1. Each node $i \in \mathcal{V}$ creates a variable $\hat{x}_i \in \mathbb{R}$ and initializes it: $\hat{x}_i \leftarrow y_i$.

Operation: At each iteration:

2. A node, say, node i, is selected randomly and equiprobably out of the set $\mathcal{V}$ of N nodes. Node i transmits a message to every node $j \in \mathcal{N}_i$, requesting their $\hat{x}_j$'s. Each node $j \in \mathcal{N}_i$ transmits $\hat{x}_j$ to node i. Node i updates $\hat{x}_i$: $\hat{x}_i \leftarrow \frac{\sum_{j \in \{i\} \cup \mathcal{N}_i} \hat{x}_j}{|\mathcal{N}_i|+1}$. Node i transmits $\hat{x}_i$ to every node $j \in \mathcal{N}_i$. Each node $j \in \mathcal{N}_i$ updates $\hat{x}_j$: $\hat{x}_j \leftarrow \hat{x}_i$. ■

Note that RHA and CHA require $2N$ real-number transmissions as initialization overhead, whereas PA, CP, A2, and DRG require none. However, PA, CP, and A2 require two real-number transmissions per iteration and DRG requires $|\mathcal{N}_i| + 1$ (where i is the node that leads an iteration), whereas RHA and CHA require only one. Also note that CP has a parameter $\beta \in (0, \infty]$ and

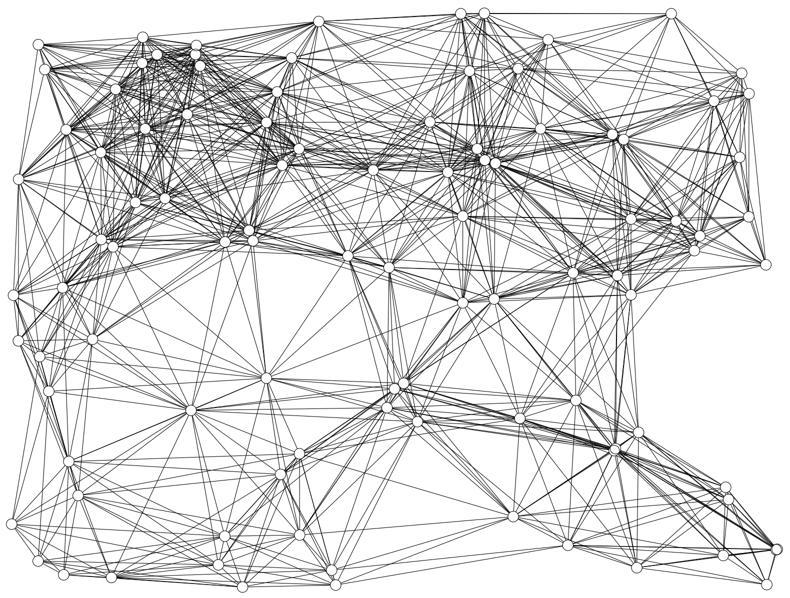


Figure 2.2: A 100-node, 1000-link multi-hop wireless network.

A2 has two parameters $\gamma \in (0, 1)$ and $\phi \in (0, \frac{1}{2})$. Moreover, PA and DRG are assumed to be free of overlapping iterations, i.e., deficiency D6.

To compare the performance of these algorithms, two sets of simulation are carried out. The first set corresponds to a single scenario of a multi-hop wireless network with $N = 100$ nodes, where each node i observes $y_i \in (0, 1)$ and has, on average, $\frac{2L}{N} = 20$ one-hop neighbors, as shown in Figure 2.2. The second set corresponds to multi-hop wireless networks modeled by random geometric graphs, with the number of nodes varying from $N = 100$ to $N = 500$, and the average number of neighbors varying from $\frac{2L}{N} = 10$ to $\frac{2L}{N} = 60$. For each N and $\frac{2L}{N}$, we generate 50 scenarios. For each scenario, we randomly and uniformly place N nodes in the unit square $(0, 1) \times (0, 1)$, gradually increase the one-hop radius until there are L links (or $\frac{2L}{N}$ neighbors on average), randomly and uniformly generate the y_i's in $(0, 1)$, and repeat this process if the resulting network is not connected. We then simulate PA, CP, A2, DRG, RHA, and

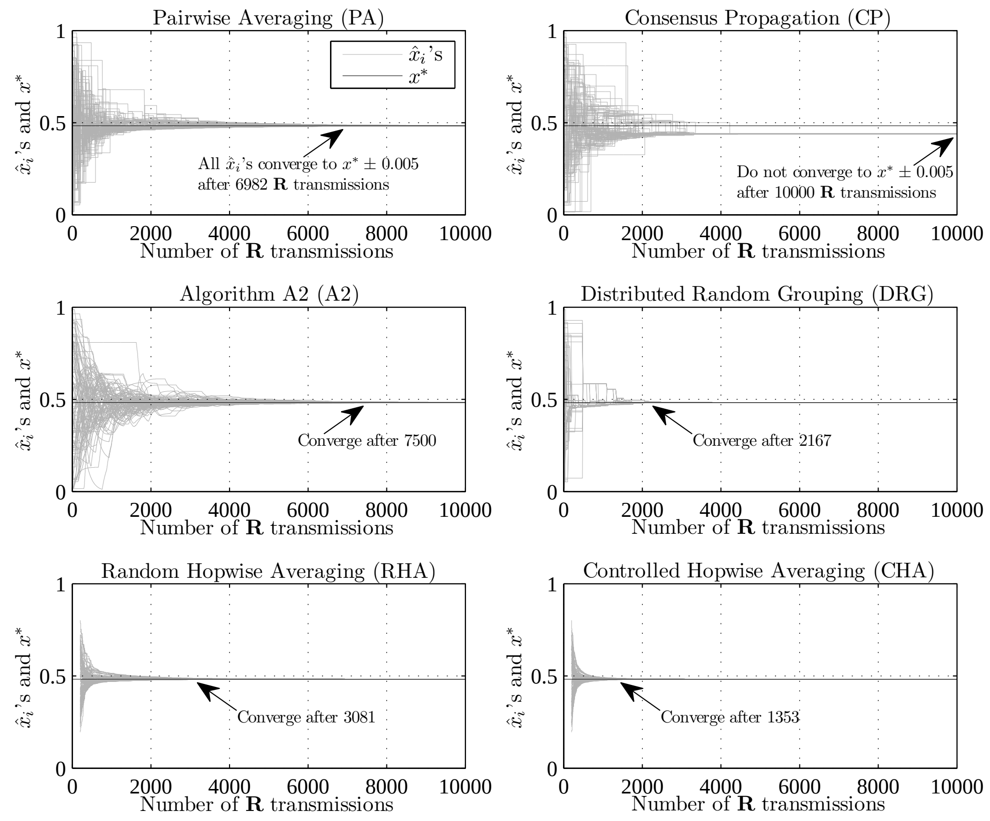


Figure 2.3: Convergence of the estimates $\hat{x}_i(k)$'s to the unknown average x^* under PA, CP, A2, DRG, RHA, and CHA for the network in Figure 2.2.

CHA until $3N^2$ real-number transmissions have occurred (i.e., three times of what flooding needs), record the number of real-number transmissions needed to converge (including initialization overhead, if any), and assume that this number is $3N^2$ if an algorithm fails to converge after $3N^2$. For both sets of simulation, we let the convergence criterion be $|\hat{x}_i - x^*| \leq 0.005 \ \forall i \in \mathcal{V}$ and the parameters be $\beta = 10^6$ for CP (obtained after some tuning), $\gamma = 0.3$ and $\phi = 0.49$ for A2 (ditto), and $\Phi(v) = \frac{1}{v}$ and $\varepsilon(v) = 0.001$ for CHA.

Results from the first set of simulation are shown in Figure 2.3. Observe that PA and A2 have roughly the same performance, requiring approximately $7,000$ real-number transmissions to converge. In contrast, CP fails to converge after $10,000$ transmissions, although it does achieve a consensus. On the other hand, DRG is found to be quite efficient, needing only approximately $2,100$ transmissions for convergence. Note that RHA outperforms PA, CP, and A2, but not DRG, while CHA is the most efficient, requiring only roughly $1,300$ transmissions to converge.

Results from the second set of simulation are shown in Figure 2.4, where the number of real-number transmissions needed to converge, averaged over 50 scenarios, is plotted as a function of the number of nodes N and the average number of neighbors $\frac{2L}{N}$. Also included in the figure, as a baseline for comparison, is the performance of flooding (i.e., N^2). Observe that regardless of N and $\frac{2L}{N}$, CP has the worst bandwidth/energy efficiency, followed by PA and A2. In addition, DRG, RHA, and CHA are all fairly efficient, with CHA again having the best efficiency. In particular, CHA is at least 20% more efficient than DRG, and around 50% more so when the network is sparsely connected, at $\frac{2L}{N} = 10$. Notice that the performance of DRG is achieved under the assumption that overlapping iterations cannot occur, a condition that CHA does not require. Finally, the significant difference in efficiency between RHA and CHA demonstrates the benefit of incorporating greedy, decentralized, feedback iteration control.

2.7 Conclusion

In this chapter, we have shown that the existing distributed averaging schemes have a few drawbacks, which hurt their bandwidth/energy efficiency. Motivated by this, we have devised RHA, an asynchronous algorithm that exploits the broadcast nature of wireless medium, achieves almost sure asymptotic convergence, and overcomes all but one of the drawbacks. To deal with the remaining drawback, on lack of control, we have introduced a new way to apply Lyapunov stability theory, namely, the concept of greedy, decentralized, feedback iteration control. Based on this concept, we have developed ICHA and CHA, established bounds on their exponential convergence rates, and shown that CHA is practical and capable of closely mimicking the behavior of ICHA. Finally, we have shown via extensive simulation that CHA is substantially more bandwidth/energy efficient than several existing schemes.

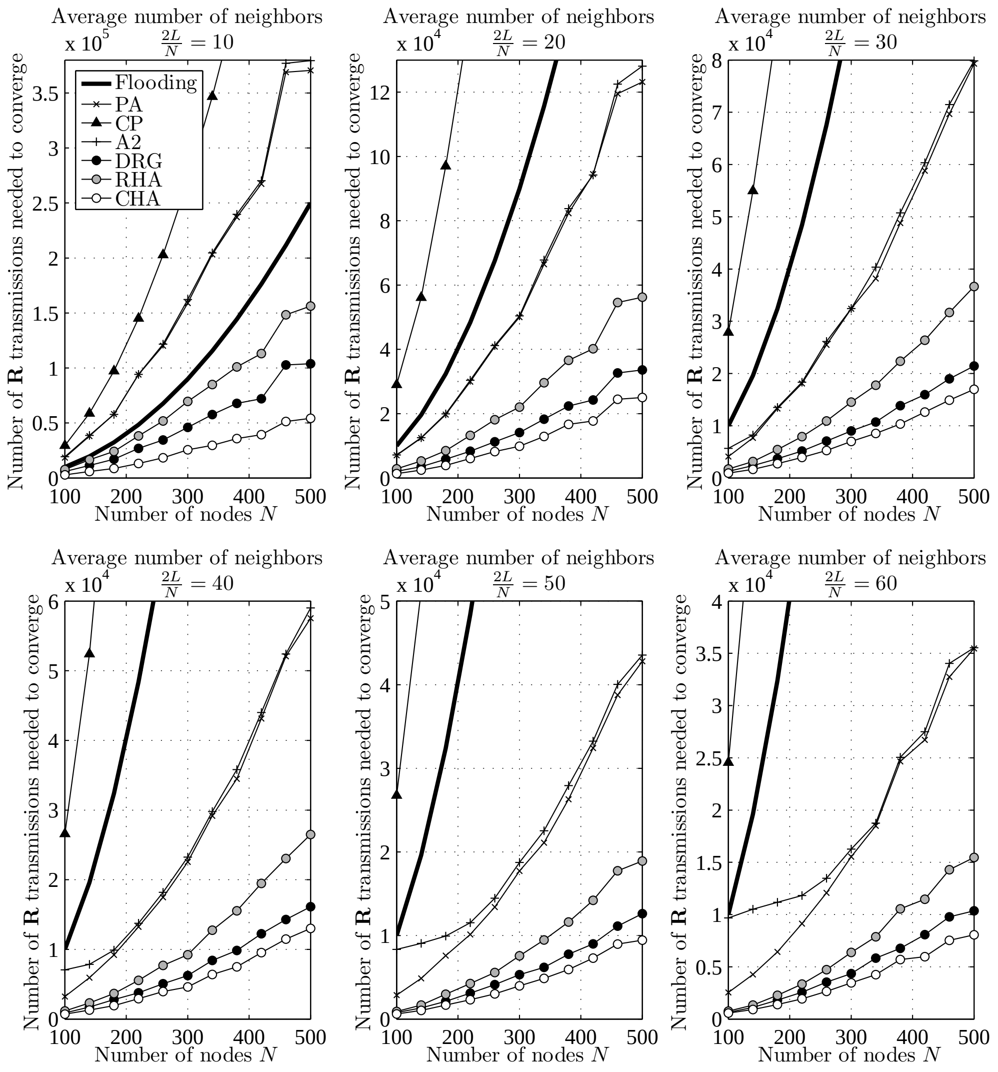


Figure 2.4: Bandwidth/energy efficiency of flooding, PA, CP, A2, DRG, RHA, and CHA on random geometric networks with varying number of nodes N and average number of neighbors $\frac{2L}{N}$.

Chapter 3 Subset Equalizing for Solving Positive Definite Linear Equations over Agent Networks

3.1 Introduction

Solving a system of linear equations $Ax = b$ is a fundamental problem with numerous applications spanning various areas of science and engineering. In this and the next chapters, we address the problem of solving an important special case of such equations over networks, whereby each agent/node i observes a symmetric positive definite matrix $A_i \in \mathbb{R}^{n\times n}$ and a vector $b_i \in \mathbb{R}^n$, and *all* of them wish to find the solution $x \in \mathbb{R}^n$ to

$$\Big(\sum_{i=1}^{N} A_i\Big)x = \sum_{i=1}^{N} b_i. \tag{3.1}$$

Since each agent/node i knows only its own A_i and b_i, none of them has sufficient information to individually solve (3.1). As a result, they must collaborate, and how to make them collaborate—robustly, scalably, and efficiently—is the focus of this chapter.

The need to solve (3.1) arises in many applications of multi-agent systems, mobile ad hoc networks, and wireless sensor networks. For instance, the least-squares solution of a distributed sensor fusion problem may be cast into the form of (3.1) [76, 77]. As another example, suppose each agent i in a multi-agent system uses a quadratic function $f_i(x) = (x - c_i)^T A_i(x - c_i)$ to represent the penalty it perceives if all the agents reach a consensus on taking action x. Then, finding the optimal action $x^\star$, which minimizes the network-wide sum of the penalties, is equivalent to solving (3.1). Finally, the widely studied distributed averaging problem [8, 11, 13, 16, 26, 30, 31, 36, 38, 39, 52, 53, 64, 69, 72, 74, 75, 78] is a special case of (3.1) with $n = 1$ and $A_i = 1\ \forall i = 1, 2, \ldots, N$.

The current literature offers a number of distributed algorithms for solving (3.1), including the continuous-time algorithm in [66], as well as the discrete-time, average-consensus-based algorithms in [76, 77]. These algorithms, however, have several limitations:

L1. *Clock synchronization*: The existing algorithms [66, 76, 77] require all the agents/nodes to always have the same clock to operate. Although techniques for reducing clock synchronization errors are available, it is often desirable that this requirement can be removed.

L2. *Static network memberships*: The existing algorithms [66, 76, 77] were developed under the assumption that agents/nodes do not join or leave the network during runtime, even though dynamic network memberships are very common, due, for example, to agent redeployment, agent mobility, sensor battery depletion/recharge, and other kinds of failures/repairs. In fact, the same can be said about the existing distributed averaging algorithms [8, 11, 13, 16, 26, 30, 31, 36, 38, 39, 52, 53, 64, 69, 72, 74, 75, 78].

L3. *Bandwidth/energy inefficient*: To implement the continuous-time algorithm in [66] with wireless communications, agents/nodes likely have to transmit many more messages than are needed. The discrete-time synchronous algorithms in [76, 77] are also inefficient, since they are based on applying an existing distributed averaging scheme to *every scalar entry* of $\sum_{i=1}^{N} A_i$ and

$\sum_{i=1}^{N} b_i$, so that each agent/node, after running the scheme for some time, has access to (the average of) these entries and, hence, may individually solve (3.1) for x. While possible, this averaging-based approach is highly inefficient because all the agents/nodes want to know is x, but are consuming bandwidth/energy to determine the intermediate quantities $\sum_{i=1}^{N} A_i$ and $\sum_{i=1}^{N} b_i$, which they really do not need to know.

This chapter, along with the next one, is devoted to the development of a family of distributed asynchronous algorithms for solving symmetric positive definite systems of linear equations over networks, that circumvent limitations L1–L3. The contributions of this chapter are as follows:

1. We present, in Section 3.2, a general agent network model, where agents are allowed to: (i) arbitrarily and asynchronously interact with one another, (ii) spontaneously join and leave the network infinitely many times, (iii) have their actions exgoneously driven and be completely unpredictable in advance, (iv) have no knowledge about the network beyond their own existence, (v) lose all their memories upon leaving the network, and (vi) not have globally unique identifiers.
2. We develop, in Section 3.3, *Subset Equalizing* (SE), an algorithm that attempts to solve (3.1) over the general agent network. SE is constructed based on decentralized, asynchronous, incremental minimization of a time-varying, quadratic Lyapunov-like function, defined on a state space with changing dimension.
3. We introduce, in Section 3.4, several notions of network connectivity, including instantaneous connectivity, connectivity, and uniform connectivity. These notions are capable of handling the general agent network

model, allowing the behavior of SE to be analyzed.

4. We derive, in Section 3.5, sufficient conditions for ensuring the boundedness, asymptotic convergence, and exponential convergence of SE, and show that these conditions are mild.
5. We illustrate, in Section 3.6, the effectiveness of SE through an example, using it to perform unconstrained quadratic optimization over a volatile multi-agent system.

The outline of this chapter is as follows: Section 3.2 describes the agent network model and formulates the problem. Section 3.3 details the development of SE. Section 3.4 introduces the notions of network connectivity. Section 3.5 characterizes the boundedness and convergence of SE. In Section 3.6, the effectiveness of SE is illustrated through an example. Finally, the conclusion of this chapter is given in Section 3.7. All proofs are included in the Appendix. Throughout the chapter, let $\mathbb{N}$, $\mathbb{P}$, $\mathbb{S}_{+}^{n}$, and $|\cdot|$ denote, respectively, the sets of nonnegative integers, positive integers, $n \times n$ symmetric positive definite matrices over $\mathbb{R}$, and the cardinality of a set.

3.2 Problem Formulation

Consider a nonempty, finite set of M agents, taking actions at each time $k \in \mathbb{N}$, according to the following model:

A1. At time $k = 0$, a nonempty subset $\mathcal{F}$ of the M agents form a network and become *members* of the network.

A2. Upon forming, each member $i \in \mathcal{F}$ observes a matrix $P_i \in \mathbb{S}_{+}^{n}$ and a vector $q_i \in \mathbb{R}^n$.

A3. The rest of the M agents become *non-members* of the network and make no observations.

A4. At each time $k \in \mathbb{P}$, three disjoint subsets of the M agents, namely—a possibly empty subset $\mathcal{J}(k)$ of the non-members, a nonempty subset $\mathcal{I}(k)$ of the members, and a possibly empty, proper subset $\mathcal{L}(k)$ of the members—take actions A5–A7 below.

A5. The set $\mathcal{J}(k)$ of non-members join the network and become members.

A6. Upon joining, the set $\mathcal{J}(k) \cup \mathcal{I}(k) \cup \mathcal{L}(k)$ of members interact, sharing information with each other and acknowledging their joining (i.e., $\mathcal{J}(k)$), staying (i.e., $\mathcal{I}(k)$), and leaving (i.e., $\mathcal{L}(k)$).

A7. Upon interacting, the set $\mathcal{L}(k)$ of members leave the network and become non-members.

A8. The rest of the M agents (i.e., complement of $\mathcal{J}(k) \cup \mathcal{I}(k) \cup \mathcal{L}(k)$) take no actions.

Assumptions A1–A8 above define a general agent network model, where: (i) initially, an arbitrary subset of the agents form the network (A1) and make one-time observations (A2), but the rest of them do not (A3); (ii) at each subsequent time, arbitrary subsets of the agents (A4) spontaneously join the network (A5), interact with one another (A6), and leave the network (A7); and (iii) agents take actions asynchronously (A8). With this model, M is the maximum number of members the network may have, and each agent at any given time is either a member or non-member, but may change membership infinitely often. Labeling the M agents as $1, 2, \ldots, M$ and letting $\mathcal{M}(k) \subset \{1, 2, \ldots, M\}$ denote the set of members upon completing the actions at time

$k \in \mathbb{N}$, the membership dynamics may be expressed as

$$\begin{aligned} \mathcal{M}(0) &= \mathcal{F}, \\ \mathcal{M}(k) &= (\mathcal{M}(k-1) \cup \mathcal{J}(k)) - \mathcal{L}(k), \quad \forall k \in \mathbb{P}, \end{aligned} \tag{3.2}$$

where, since $\mathcal{F} \neq \emptyset$ and $\mathcal{L}(k) \subsetneq \mathcal{M}(k-1)$ $\forall k \in \mathbb{P}$, the network always has at least one member, i.e., $\mathcal{M}(k) \neq \emptyset$ $\forall k \in \mathbb{N}$. Moreover, since $\mathcal{J}(k)$ and $\mathcal{L}(k)$ may be empty for some $k \in \mathbb{P}$ but $\mathcal{I}(k) \neq \emptyset$ $\forall k \in \mathbb{P}$, while there may not always be membership changes, there are always member interactions, among the agents in

$$\mathcal{J}(k) \cup \mathcal{I}(k) \cup \mathcal{L}(k), \quad \forall k \in \mathbb{P}. \tag{3.3}$$

Since the membership dynamics (3.2) and the member interactions (3.3) are completely characterized by $\mathcal{F}$, $\mathcal{J}(k)$, $\mathcal{I}(k)$, and $\mathcal{L}(k)$ $\forall k \in \mathbb{P}$, the network may be viewed as being driven by a sequence $\mathcal{A}$ of agent actions, where

$$\mathcal{A} = (\mathcal{F}, \mathcal{J}(1), \mathcal{I}(1), \mathcal{L}(1), \mathcal{J}(2), \mathcal{I}(2), \mathcal{L}(2), \ldots). \tag{3.4}$$

The problem addressed in this chapter may be stated as follows: Given the agent network modeled by A1–A8, construct a distributed asynchronous algorithm of iterative nature, which allows the ever-changing members of the network to asymptotically compute the solution $z \in \mathbb{R}^n$ of the following symmetric positive definite system of linear equations, defined by the one-time observations P_i and q_i $\forall i \in \mathcal{M}(0)$ of the initial members:

$$\Big(\sum_{i \in \mathcal{M}(0)} P_i\Big) z = \sum_{i \in \mathcal{M}(0)} q_i. \tag{3.5}$$

For versatility reasons, the algorithm should exhibit the following properties:

P1. It should allow the sequence $\mathcal{A}$ of agent actions to be dictated by an exogenous source, for which the agents have no control over, since, for example, in a sensor network, $\mathcal{J}(k)$, $\mathcal{I}(k)$, and $\mathcal{L}(k)$ may be governed by sensor reseeding, mobility, and failures, all of which may be forced exogenously.

P2. It should allow the agents to not know the values of M, k, $\mathcal{F}$, $\mathcal{J}(k)$, $\mathcal{I}(k)$, $\mathcal{L}(k)$, and $\mathcal{M}(k)$ $\forall k \in \mathbb{P}$, since in many practical situations they are not available, or at least not known ahead of time.

P3. It should allow the agents to lose all their memories upon leaving the network, since the departure may be due to, for instance, agent failures.

P4. It should allow the agents to not have globally unique identifiers, since in some applications they are not assigned one.

Due to property P1 and the fact that $\mathcal{A}$ dictates all but how members share information whenever they interact in A6, construction of an algorithm that solves this problem amounts to specifying how information is shared and processed whenever members interact. Moreover, due to property P4, sharing of information via flooding is prohibited.

3.3 Subset Equalizing

In this section, we develop an algorithm having properties P1–P4 by designing a networked dynamical system, which evolves asynchronously whenever subsets of the members interact and share information. We show that ideas from Lyapunov stability theory and tools from optimization may be utilized to shape the evolution of the networked dynamical system.

Suppose each agent $i \in \{1, 2, \ldots, M\}$ maintains state variables $z_i(k) \in \mathbb{R}^n \cup \{\#\}$ and $Q_i(k) \in \mathbb{S}_+^n \cup \{\#\}$, where $z_i(k)$ represents agent i's estimate of the unknown solution z of (3.5) upon completing the actions at time $k \in \mathbb{N}$, and $Q_i(k)$ represents an additional state variable which will be used shortly to define a Lyapunov-like function. The symbol # denotes "undefined" and is the value both $z_i(k)$ and $Q_i(k)$ assume whenever agent i is a non-member of the network, i.e.,

$$z_i(k) = \#, \quad \forall k \in \mathbb{N},\ \forall i \in \{1, 2, \ldots, M\} - \mathcal{M}(k), \tag{3.6}$$

$$Q_i(k) = \#, \quad \forall k \in \mathbb{N},\ \forall i \in \{1, 2, \ldots, M\} - \mathcal{M}(k). \tag{3.7}$$

This symbol is introduced to ensure that the algorithm exhibits property P3. The initial conditions $z_i(0)$ and $Q_i(0)$ will be specified shortly.

To define the evolution of $z_i(k)$ and $Q_i(k)$, consider the following time-varying quadratic Lyapunov-like function:

$$\begin{aligned} &V(z_1(k), z_2(k), \ldots, z_M(k), Q_1(k), Q_2(k), \ldots, Q_M(k)) \\ &\quad = \sum_{i \in \mathcal{M}(k)} (z_i(k) - z)^T Q_i(k)(z_i(k) - z). \end{aligned} \tag{3.8}$$

For convenience, we will write this function as $V(k)$, omitting its arguments. Note that whenever agent i is a non-member, i.e., $i \in \{1, 2, \ldots, M\} - \mathcal{M}(k)$, during which $z_i(k) = \#$ and $Q_i(k) = \#$, its state variables do not appear in (3.8), so that $V(k)$ is always well-defined. Also note that $V(k)$ has an ever-changing number of terms, akin to a function defined on a state space with ever-changing dimension. We refer to $V(k)$ as a Lyapunov-*like* function because strictly speaking it does not satisfy the definition of a true Lyapunov function candidate, although we intend it to mimic such a role.

The Lyapunov-like function $V(k)$ satisfies $V(k) \geq 0\ \forall k \in \mathbb{N}$, since $Q_i(k) \in \mathbb{S}_+^n \cup \{\#\}$. Moreover, $V(k) = 0$ if and only if $z_i(k) = z\ \forall i \in \mathcal{M}(k)$. However, $\lim_{k\to\infty} V(k) = 0$ does not imply $\lim_{k\to\infty} z_i(k) = z\ \forall i \in \mathcal{M}(k)$ because $Q_i(k)$ may be "losing" its positive definiteness as $k \to \infty$. In fact,

"$\lim_{k\to\infty} z_i(k) = z$" may not even be well-defined because $z_i(k)$ may be alternating between $z_i(k) \in \mathbb{R}^n$ and $z_i(k) = \#$ as $k \to \infty$. Nevertheless, if there exists $\alpha > 0$ such that $Q_i(k) - \alpha I \in \mathbb{S}_+^n$ $\forall k \in \mathbb{N}$ $\forall i \in \mathcal{M}(k)$, then $\lim_{k\to\infty} V(k) = 0$ does imply that $z_i(k)$, whenever not equal to $\#$, goes to z as $k \to \infty$.

As it follows from the above, $V(k)$ does not carry the same implication as a true Lyapunov function candidate. However, it is still valuable to the M agents, who otherwise have little idea on how they should evolve their $z_i(k)$'s and $Q_i(k)$'s. Indeed, this $V(k)$ offers a structure that enables decentralized, asynchronous, incremental minimization of $V(k)$ without any agent ever knowing its value, as is shown below.

Suppose the agent network is executing A6 at some given time $k \in \mathbb{P}$, i.e., the set $\mathcal{J}(k) \cup \mathcal{I}(k) \cup \mathcal{L}(k)$ of members are interacting, sharing with each other information on

$$z_i(k-1) \text{ and } Q_i(k-1), \quad \forall i \in \mathcal{I}(k) \cup \mathcal{L}(k) \tag{3.9}$$

and hoping to use this information to jointly determine

$$z_i(k) \text{ and } Q_i(k), \quad \forall i \in \mathcal{J}(k) \cup \mathcal{I}(k), \tag{3.10}$$

so that $V(k)$ defined in (3.8) would be less than $V(k-1)$ or, better yet, be minimized, while the rest of the members stay idle, i.e.,

$$z_i(k) = z_i(k-1), \quad \forall i \in \mathcal{M}(k) - (\mathcal{J}(k) \cup \mathcal{I}(k)), \tag{3.11}$$

$$Q_i(k) = Q_i(k-1), \quad \forall i \in \mathcal{M}(k) - (\mathcal{J}(k) \cup \mathcal{I}(k)). \tag{3.12}$$

Since all that the agents in $\mathcal{J}(k) \cup \mathcal{I}(k) \cup \mathcal{L}(k)$ know is (3.9) and since they do not know z but z appears in $V(k)$, they do not have sufficient information to minimize $V(k)$. To circumvent this issue, notice that $V(k)-V(k-1)$ may be expressed as

$$\begin{aligned}V(k)-V(k-1) &= \sum_{\substack{i\in\mathcal{J}(k)\\ \cup\mathcal{I}(k)}} z_i(k)^T Q_i(k) z_i(k) - \sum_{\substack{i\in\mathcal{I}(k)\\ \cup\mathcal{L}(k)}} z_i(k-1)^T Q_i(k-1) z_i(k-1)\\ &- 2z^T\Big[\sum_{\substack{i\in\mathcal{J}(k)\\ \cup\mathcal{I}(k)}} Q_i(k)z_i(k) - \sum_{\substack{i\in\mathcal{I}(k)\\ \cup\mathcal{L}(k)}} Q_i(k-1)z_i(k-1)\Big]\\ &+ z^T\Big[\sum_{\substack{i\in\mathcal{J}(k)\\ \cup\mathcal{I}(k)}} Q_i(k) - \sum_{\substack{i\in\mathcal{I}(k)\\ \cup\mathcal{L}(k)}} Q_i(k-1)\Big]z. \qquad (3.13)\end{aligned}$$

Note that every variable appearing inside the two pairs of brackets in (3.13) appears also in either (3.9) or (3.10). Also note that the unknown z appears only right by the brackets. Thus, if the members determine (3.10) in terms of (3.9) so that the terms inside the brackets in (3.13) vanish, i.e.,

$$\sum_{i\in\mathcal{I}(k)\cup\mathcal{L}(k)} Q_i(k-1)z_i(k-1) = \sum_{i\in\mathcal{J}(k)\cup\mathcal{I}(k)} Q_i(k)z_i(k), \quad \forall k\in\mathbb{P}, \qquad (3.14)$$

$$\sum_{i\in\mathcal{I}(k)\cup\mathcal{L}(k)} Q_i(k-1) = \sum_{i\in\mathcal{J}(k)\cup\mathcal{I}(k)} Q_i(k), \quad \forall k\in\mathbb{P}, \qquad (3.15)$$

then the effect of z would be eliminated. Since the second summation in (3.13) is a constant and since $V(k-1)$ is also a constant, now minimizing $V(k)$ is equivalent to minimizing the first summation in (3.13) subject to (3.14) and (3.15). Also, if we freeze the value of $Q_i(k)$ $\forall i \in \mathcal{J}(k)\cup\mathcal{I}(k)$, then the problem becomes an equality-constrained, convex optimization over $z_i(k)$ $\forall i \in \mathcal{J}(k)\cup\mathcal{I}(k)$. By forming the Lagrangian of this convex optimization problem, setting its gradient to zero, and solving for $z_i(k)$ $\forall i \in \mathcal{J}(k)\cup\mathcal{I}(k)$ and the Lagrange multipliers, we know that $z_i(k)$ $\forall i \in \mathcal{J}(k)\cup\mathcal{I}(k)$ must be identical. This, along with (3.14) and (3.15), analytically solves the problem, resulting in the unique optimizer

$$z_i(k) = \Big(\sum_{j\in\mathcal{I}(k)\cup\mathcal{L}(k)} Q_j(k-1)\Big)^{-1}\Big(\sum_{j\in\mathcal{I}(k)\cup\mathcal{L}(k)} Q_j(k-1)z_j(k-1)\Big),$$
$$\forall i \in \mathcal{J}(k)\cup\mathcal{I}(k). \qquad (3.16)$$

Thus, the optimal action, which minimizes $V(k)$ under conditions (3.14) and (3.15), is an *equalizing action*, whereby the state variables $z_i(k)$'s of the set $\mathcal{J}(k)\cup\mathcal{I}(k)$ of members are equalized. Furthermore, to guarantee (3.15), $Q_i(k)$ $\forall i \in \mathcal{J}(k)\cup\mathcal{I}(k)$ may be updated according to

$$Q_i(k) = Q_i(k-1), \qquad (3.17)$$

when there are no membership changes, i.e., $\mathcal{J}(k) = \emptyset$ and $\mathcal{L}(k) = \emptyset$, and according to

$$Q_i(k) = \frac{1}{|\mathcal{J}(k)\cup\mathcal{I}(k)|}\sum_{j\in\mathcal{I}(k)\cup\mathcal{L}(k)} Q_i(k-1) \qquad (3.18)$$

when there are membership changes. Note that (3.20) and (3.21) are not the only way $Q_i(k)$ may be updated, but perhaps the simplest.

The following lemma shows that with the above, at each time $k \in \mathbb{P}$, the minimized $V(k)$ cannot exceed $V(k-1)$:

Lemma 3.1. *Consider the agent network modeled by A1–A8 and suppose* (3.11)*,* (3.12)*,* (3.16)*,* (3.17)*, and* (3.18) *hold. Then, for any $\mathcal{A}$, the sequence* $(V(k))_k = 0^\infty$ *is non-increasing.*

Proof. See Appendix B.1. □

Expressions (3.6), (3.7), (3.11), (3.12), (3.16), (3.17), and (3.18) collectively define a distributed asynchronous iterative algorithm, leading to a linear networked dynamical system that evolves as follows: for each $k \in \mathbb{P}$,

$$z_i(k) = \begin{cases} \Big(\sum\limits_{j\in\mathcal{I}(k)\cup\mathcal{L}(k)} Q_j(k-1)\Big)^{-1}\Big(\sum\limits_{j\in\mathcal{I}(k)\cup\mathcal{L}(k)} Q_j(k-1)z_j(k-1)\Big), & \text{if } i\in\mathcal{J}(k)\cup\mathcal{I}(k), \\ \#, & \text{if } i\in\mathcal{L}(k), \\ z_i(k-1), & \text{otherwise.} \end{cases} \tag{3.19}$$

If $\mathcal{M}(k) = \mathcal{M}(k-1)$, then

$$Q_i(k) = Q_i(k-1), \quad \forall i \in \{1, 2, \dots, M\}. \tag{3.20}$$

Otherwise,

$$Q_i(k) = \begin{cases} \frac{1}{|\mathcal{J}(k)\cup\mathcal{I}(k)|}\sum_{j\in\mathcal{I}(k)\cup\mathcal{L}(k)} Q_i(k-1), & \text{if } i \in \mathcal{J}(k)\cup\mathcal{I}(k), \\ \#, & \text{if } i \in \mathcal{L}(k), \\ Q_i(k-1), & \text{otherwise.} \end{cases} \tag{3.21}$$

In addition, let the initial conditions be

$$z_i(0) = \begin{cases} P_i^{-1}q_i, & \text{if } i \in \mathcal{M}(0), \\ \#, & \text{otherwise,} \end{cases} \tag{3.22}$$

$$Q_i(0) = \begin{cases} P_i, & \text{if } i \in \mathcal{M}(0), \\ \#, & \text{otherwise.} \end{cases} \tag{3.23}$$

Notice that the initial conditions (3.22) and (3.23), along with (3.19), (3.20), and (3.21), ensure

$$\sum_{i\in\mathcal{M}(k)} Q_i(k)z_i(k) = \sum_{i\in\mathcal{M}(0)} q_i, \quad \forall k \in \mathbb{N}, \tag{3.24}$$

$$\sum_{i\in\mathcal{M}(k)} Q_i(k) = \sum_{i\in\mathcal{M}(0)} P_i, \quad \forall k \in \mathbb{N}. \tag{3.25}$$

Hence, once $z_i(k)$ $\forall i \in \mathcal{M}(k)$ achieve consensus, the consensus must be z due to (3.5).

Since at each time $k \in \mathbb{P}$, the algorithm involves an *equalizing* action taken by a *subset* $\mathcal{J}(k)\cup\mathcal{I}(k)\cup\mathcal{L}(k)$ of the agents, we refer to this algorithm as *Subset Equalizing* (SE). A complete description of SE is as follows:

Algorithm 3.1 (Subset Equalizing)**.**

Initialization: At time $k = 0$:

1. Each agent $i \in \{1, 2, \ldots, M\}$ creates variables $z_i(k) \in \mathbb{R}^n \cup \{\#\}$ and $Q_i(k) \in \mathbb{S}_+^n \cup \{\#\}$ and initializes them according to (3.22) and (3.23).

Operation: At each time $k \in \mathbb{P}$:

2. Each agent $i \in \{1, 2, \ldots, M\}$ updates $z_i(k)$ according to (3.19).
3. If $\mathcal{J}(k) = \emptyset$ and $\mathcal{L}(k) = \emptyset$, then each agent $i \in \{1, 2, \ldots, M\}$ updates $Q_i(k)$ according to (3.20). Otherwise, each agent $i \in \{1, 2, \ldots, M\}$ updates $Q_i(k)$ according to (3.21). ■

3.4 Network Connectivity

With SE, every time a subset of the M agents interact, they update their state variables $z_i(k)$'s and $Q_i(k)$'s in such a manner that the value of the Lyapunov-like function $V(k)$ is non-increasing. However, the agents themselves, however, cannot guarantee that $V(k)$ would go to zero. In fact, it is not difficult to imagine a sequence $\mathcal{A}$ of agent actions where $V(k)$ is bounded away from zero (e.g., a network where two groups of agents never interact with one another). Hence, in order to characterize the convergence behavior of SE, it is necessary to introduce notions of network connectivity, which can handle the general agent network modeled by A1–A8.

To define such notions, suppose a sequence $\mathcal{A}$ of agent actions and a time $k \in \mathbb{N}$ are given. For each time $\ell \geq k$, let us associate with each agent $i \in \{1, 2, \ldots, M\}$ a set $\mathcal{C}_i(k, \ell) \subset \mathcal{M}(\ell)$. The set $\mathcal{C}_i(k, \ell)$ is initialized to

$$\mathcal{C}_i(k, k) = \begin{cases} \{i\}, & \text{if } i \in \mathcal{M}(k), \\ \emptyset, & \text{otherwise}, \end{cases} \tag{3.26}$$

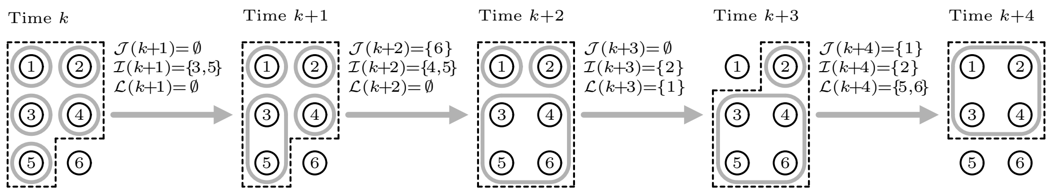

Time k
$\mathcal{J}(k{+}1)=\emptyset$
$\mathcal{I}(k{+}1)=\{3,5\}$
$\mathcal{L}(k{+}1)=\emptyset$
Time $k{+}1$
$\mathcal{J}(k{+}2)=\{6\}$
$\mathcal{I}(k{+}2)=\{4,5\}$
$\mathcal{L}(k{+}2)=\emptyset$
Time $k{+}2$
$\mathcal{J}(k{+}3)=\emptyset$
$\mathcal{I}(k{+}3)=\{2\}$
$\mathcal{L}(k{+}3)=\{1\}$
Time $k{+}3$
$\mathcal{J}(k{+}4)=\{1\}$
$\mathcal{I}(k{+}4)=\{2\}$
$\mathcal{L}(k{+}4)=\{5,6\}$
Time $k{+}4$
1
2
3
4
5
6

Figure 3.1: An example showing that an agent network is connected at time k.

and defined recursively for each $\ell \geq k+1$ as

$$\mathcal{C}_i(k,\ell) = \begin{cases} \Big(\cup_{\substack{j\in\mathcal{I}(\ell)\\ \cup\mathcal{L}(\ell)}}\mathcal{C}_j(k, & \ell-1)\cup\mathcal{J}(\ell)\Big) - \mathcal{L}(\ell), \\ & \text{if } i \in \Big(\cup_{\substack{j\in\mathcal{I}(\ell)\\ \cup\mathcal{L}(\ell)}}\mathcal{C}_j(k,\ell-1)\cup\mathcal{J}(\ell)\Big) - \mathcal{L}(\ell), \\ \emptyset, & \text{if } i \in \mathcal{L}(\ell), \\ \mathcal{C}_i(k,\ell-1), & \text{otherwise.} \end{cases} \tag{3.27}$$

Equation (3.26) suggests that at time $\ell = k$, each member i's $C_i(k,\ell)$ contains only itself, whereas each non-member i's $C_i(k,\ell)$ is empty. Equation (3.27) suggests that at each subsequent time $\ell \geq k+1$, the $C_i(k,\ell)$ of every agent i that remains a member either stays the same or adds those members it has directly or indirectly interacted with at time ℓ and deduct those members that leaves the network at time ℓ. In addition, if a member i leaves the network, its $C_i(k,\ell)$ would be reset to empty. Figure 3.1 illustrates how the set $\mathcal{C}_i(k,\ell)$ of each agent i changes over time in response to the sequence $\mathcal{A}$ of agent actions. The dark dashed line in this figure separates members from non-members of the network. For example, agent 6 is not a member at time $k+1$. The gray solid line represents $\mathcal{C}_i(k,\ell)$ of each agent i in the following sense: If two agents i and j are enclosed by the same gray solid line, then $\mathcal{C}_i(k,\ell) = \mathcal{C}_j(k,\ell)$. For example, at time $k+2$, $C_1(k,k+2) = \{1\}$, $C_2(k,k+2) = \{2\}$, $C_3(k,k+2) = C_4(k,k+2) = C_5(k,k+2) = C_6(k,k+2) = \{3,4,5,6\}$.

Having defined and illustrated the set $\mathcal{C}_i(k,\ell)$, we now state the following definitions. Suppose $\mathcal{A}$ is given. For each $k \in \mathbb{N}$, let $D_k = \{\ell \geq k : \mathcal{C}_i(k,\ell) =$

$\mathcal{M}(\ell), \forall i \in \mathcal{M}(\ell)\}$, so that each element in D_k represents the time at which the number of distinct, nonempty $\mathcal{C}_i(k,\ell)$'s is 1. Also let $h : \mathbb{N} \to \mathbb{N} \cup \{\infty\}$ be defined as $h(k) = \inf D_k - k$ and let $h^* = \sup_{k\in\mathbb{N}} h(k)$. For example, we have $h(k) = 4$ in Figure 3.1, which intuitively means that the network needs four time instants to become "connected". Based on this function h, the notion of network connectivity in this chapter may be defined as follows:

Definition 3.1. The agent network modeled by A1–A8 is said to be *connected under $\mathcal{A}$ at time $k \in \mathbb{N}$* if $h(k) < \infty$. It is said to be *connected under $\mathcal{A}$* if $h(k) < \infty \ \forall k \in \mathbb{N}$, and *uniformly connected under $\mathcal{A}$* if $h^* < \infty$.

Based on the above definitions, the agent network illustrated in Figure 3.1 is connected under $\mathcal{A}$ at time k because $h(k) = 4 < \infty$. To further illustrate the notions of network connectivity, consider the following examples.

Example 3.1. Consider the agent network modeled by A1–A8 and the use of SE described in Algorithm 3.1. Let $M = 3$. Suppose $\mathcal{A}$ is such that: $\mathcal{F} = \{1, 2\}$ and for any $\ell \in \mathbb{N}$, $(\mathcal{J}(k), \mathcal{I}(k), \mathcal{L}(k)) = (\{3\}, \{1\}, \emptyset)$ if $k = 1 + 6\ell$, $(\mathcal{J}(k), \mathcal{I}(k), \mathcal{L}(k)) = (\emptyset, \{3\}, \{1\})$ if $k = 2 + 6\ell$, $(\mathcal{J}(k), \mathcal{I}(k), \mathcal{L}(k)) = (\{1\}, \{2\}, \emptyset)$ if $k = 3 + 6\ell$, $(\mathcal{J}(k), \mathcal{I}(k), \mathcal{L}(k)) = (\emptyset, \{1\}, \{2\})$ if $k = 4 + 6\ell$, $(\mathcal{J}(k), \mathcal{I}(k), \mathcal{L}(k)) = (\{2\}, \{3\}, \emptyset)$ if $k = 5 + 6\ell$, $(\mathcal{J}(k), \mathcal{I}(k), \mathcal{L}(k)) = (\emptyset, \{2\}, \{3\})$ if $k = 6 + 6\ell$. Then, for any $k \in \mathbb{N}$, $h(k) = \infty$. Therefore, the agent network is not connected under $\mathcal{A}$ at any time $k \in \mathbb{N}$. □

Example 3.1 illustrates a scenario where two pieces of information, (P_1, q_1) and (P_2, q_2), are passed around the agents as they join and leave the network, but *never* get a change to "mix". Indeed, applying Definition 3.1 shows that the network is not connected under $\mathcal{A}$ at any time $k \in \mathbb{N}$. Example 3.2 below illustrates a very similar scenario, but the pieces of information,

(P_1, q_1) and (P_2, q_2), are allowed to be "mixed". Indeed, applying Definition 3.1 shows that the network is uniformly connected under $\mathcal{A}$.

Example 3.2. Consider the agent network modeled by A1–A8 and the use of SE described in Algorithm 3.1. Let $M = 3$. Suppose $\mathcal{A}$ is such that: $\mathcal{F} = \{1, 2\}$ and for any $\ell \in \mathbb{N}$, $(\mathcal{J}(k), \mathcal{I}(k), \mathcal{L}(k)) = (\{3\}, \{1\}, \emptyset)$ if $k = 1 + 6\ell$, $(\mathcal{J}(k), \mathcal{I}(k), \mathcal{L}(k)) = (\emptyset, \{2\}, \{1\})$ if $k = 2 + 6\ell$, $(\mathcal{J}(k), \mathcal{I}(k), \mathcal{L}(k)) = (\{1\}, \{2\}, \emptyset)$ if $k = 3 + 6\ell$, $(\mathcal{J}(k), \mathcal{I}(k), \mathcal{L}(k)) = (\emptyset, \{3\}, \{2\})$ if $k = 4 + 6\ell$, $(\mathcal{J}(k), \mathcal{I}(k), \mathcal{L}(k)) = (\{2\}, \{3\}, \emptyset)$ if $k = 5 + 6\ell$, $(\mathcal{J}(k), \mathcal{I}(k), \mathcal{L}(k)) = (\emptyset, \{1\}, \{3\})$ if $k = 6 + 6\ell$. Then, for any $k \in \mathbb{N}$, $h(k) = 2$ if k is even and $h(k) = 3$ if k is odd, implying that $h^* = 3 < \infty$. Therefore, the agent network is uniformly connected under $\mathcal{A}$. $\square$

Example 3.3 illustrates a scenario where the network is connected, but not uniformly so, under $\mathcal{A}$, because the interactions between agents 1 and 2 become less and less frequent, as if the network is "losing" its connectivity.

Example 3.3. Consider the agent network modeled by A1–A8 and the use of SE described in Algorithm 3.1. Let $M = 3$. Suppose $\mathcal{A}$ is such that: $\mathcal{F} = \{1, 2, 3\}$ and for any $\ell \in \mathbb{P}$, $(\mathcal{J}(k), \mathcal{I}(k), \mathcal{L}(k)) = (\emptyset, \{1, 2\}, \emptyset)$ if $k = \ell(\ell + 1)/2$, $(\mathcal{J}(k), \mathcal{I}(k), \mathcal{L}(k)) = (\emptyset, \{2, 3\}, \emptyset)$ otherwise. Then, $\forall \ell \in \mathbb{P}$, $\forall k \in [\ell(\ell + 1)/2, (\ell + 1)(\ell + 2)/2)$, $h(k) \leq \ell + 1 < \infty$. In particular, $h(\ell(\ell + 1)/2) = \ell + 1$, implying that $h^* = \infty$. Therefore, the agent network is connected under $\mathcal{A}$ but not uniformly connected under $\mathcal{A}$. $\square$

3.5 Boundedness and Convergence

In this section, we present mild sufficient conditions for the boundedness, asymptotic convergence, and exponential convergence of SE. To present the results, let $\beta > 0$ denote the spectral radius of $\sum_{i\in\mathcal{M}(0)} P_i$. In addition, consider the following definition and proposition:

Definition 3.2. Consider the agent network modeled by A1–A8 and the use of SE described in Algorithm 3.1. The sequence $\{Q_i(k)\}_{k\in\mathbb{N},i\in\mathcal{M}(k)}$ is said to be *uniformly positive definite under $\mathcal{A}$* if $\exists\alpha > 0$ such that $\forall k \in \mathbb{N}$, $\forall i \in \mathcal{M}(k)$, $Q_i(k) - \alpha I \in \mathbb{S}^n_+$.

Proposition 3.1. *Whether or not the sequence $\{Q_i(k)\}_{k\in\mathbb{N},i\in\mathcal{M}(k)}$ is uniformly positive definite under $\mathcal{A}$ is independent of the observations $P_i \in \mathbb{S}^n_+$, $q_i \in \mathbb{R}^n$, $\forall i \in \mathcal{F}$.*

Proof. See Appendix B.2. □

Based on Definition 3.2, we state below our first main result on the boundedness of SE:

Theorem 3.1. *Consider the agent network modeled by A1–A8 and the use of SE described in Algorithm 3.1. Let $\mathcal{A}$ be given. Then, $Q_i(k)$ is bounded as follows:*

$$Q_i(k) \leq \beta I, \quad \forall k \in \mathbb{N},\ \forall i \in \mathcal{M}(k). \tag{3.28}$$

If, in addition, the sequence $\{Q_i(k)\}_{k\in\mathbb{N},i\in\mathcal{M}(k)}$ is uniformly positive definite under $\mathcal{A}$, then $z_i(k)$ is bounded as follows:

$$\|z_i(k) - z\|^2 \leq \frac{V(0)}{\alpha}, \quad \forall k \in \mathbb{N},\ \forall i \in \mathcal{M}(k), \tag{3.29}$$

where α is any positive number satisfying $Q_i(k) - \alpha I \in \mathbb{S}^n_+$ $\forall k \in \mathbb{N}$ $\forall i \in \mathcal{M}(k)$.

Proof. See Appendix B.3. □

Theorem 3.1 asserts that all the $Q_i(k)$'s, whenever not equal to #, are always bounded from above, regardless of the sequence $\mathcal{A}$ of agent actions. In addition, if they turn out to be bounded from below, i.e., the sequence $\{Q_i(k)\}_{k\in\mathbb{N},i\in\mathcal{M}(k)}$ is uniformly positive definite under $\mathcal{A}$, then all the $z_i(k)$'s, whenever not equal to #, are guaranteed to stay within a ball of radius $\sqrt{V(0)/\alpha}$ centered at the solution z. However, such an α may not exist, as the following example shows:

Example 3.4. Consider the agent network modeled by A1–A8 and the use of SE described in Algorithm 3.1. Let $M = 3$, $\mathcal{F} = \{1, 2\}$, $(\mathcal{J}(k), \mathcal{I}(k), \mathcal{L}(k)) = (\{3\}, \{1\}, \emptyset)$ if k is odd, and $(\mathcal{J}(k), \mathcal{I}(k), \mathcal{L}(k)) = (\emptyset, \{2\}, \{3\})$ if k is even, thereby defining $\mathcal{A}$ via (3.4). With this $\mathcal{A}$, agent 3 repeatedly joins the network, interacts with agent 1 upon joining, leaves the network subsequently, and interacts with agent 2 prior to leaving, so that the agent network is connected under $\mathcal{A}$, by Definition 3.1. Suppose $P_1 = P_2 = 1$, $q_1 = 1$, and $q_2 = 2$, so that $z = 1.5$ from (3.5). Then, it is straightforward to show that $\forall k \in \mathbb{N}$, $Q_1(k) = (\frac{1}{2})^{\lceil \frac{k}{2} \rceil}$, $Q_2(k) = 2 - (\frac{1}{2})^{\lfloor \frac{k}{2} \rfloor}$, $Q_3(k) = (\frac{1}{2})^{\lceil \frac{k}{2} \rceil}$ if k is odd, $Q_3(k) = \#$ if k is even, $z_1(k) = 1$, $z_2(k) = (3 - (\frac{1}{2})^{\lfloor \frac{k}{2} \rfloor})/(2 - (\frac{1}{2})^{\lfloor \frac{k}{2} \rfloor})$, $z_3(k) = 1$ if k is odd, and $z_3(k) = \#$ if k is even. It follows that $\lim_{k\to\infty} Q_1(k) = 0$, $\lim_{k\to\infty} Q_2(k) = 2$, $\lim_{k\to\infty} z_1(k) = 1$, and $\lim_{k\to\infty} z_2(k) = 1.5$. $\square$

In the above example, the $\alpha > 0$ can not be found with which $\forall k \in \mathbb{N}$, $Q_1(k) - \alpha I \in \mathbb{S}^n_+$, implying that the sequence $\{Q_i(k)\}_{k\in\mathbb{N},i\in\mathcal{M}(k)}$ is not uniformly positive definite under $\mathcal{A}$. However, $Q_i(k)$ and $z_i(k)$ are bounded. Therefore, the condition that $\{Q_i(k)\}_{k\in\mathbb{N},i\in\mathcal{M}(k)}$ is uniformly positive definite under $\mathcal{A}$ in Theorem 3.1 is sufficient, but not necessary, for the boundedness of SE.

In general, given $\mathcal{A}$, it is not easy to check whether the resulting sequence

$\{Q_i(k)\}_{k\in\mathbb{N},i\in\mathcal{M}(k)}$ is uniformly positive definite under $\mathcal{A}$. However, if $\mathcal{A}$ happens to be such that every agent joins and leaves the network only *finitely* many times—a rather mild condition—then the uniform positive definiteness of the $\{Q_i(k)\}_{k\in\mathbb{N},i\in\mathcal{M}(k)}$ can be immediately verified. The following definition and corollary formalize this observation:

Definition 3.3. Consider the agent network modeled by A1–A8. The membership dynamics (3.2) are said to be *ultimately static under $\mathcal{A}$* if $\exists k \in \mathbb{N}$ such that $\forall \ell > k$, $\mathcal{M}(\ell) = \mathcal{M}(k)$, i.e., $\mathcal{J}(\ell) = \emptyset$ and $\mathcal{L}(\ell) = \emptyset$.

Corollary 3.1. *Let $\mathcal{A}$ be given. If the membership dynamics* (3.2) *are ultimately static under $\mathcal{A}$, then $Q_i(k)$ and $z_i(k)$ are bounded as in* (3.28) *and* (3.29) *for some $\alpha > 0$.*

Proof. See Appendix B.4. □

In Theorem 3.1 and Corollary 3.1, the network is not assumed to be connected since such an assumption is not needed for boundedness of SE. For convergence, however, this assumption is crucial. The following lemma, which makes use of this assumption, is an important step towards establishing both the asymptotic and exponential convergence of SE:

Lemma 3.2. *Consider the agent network modeled by A1–A8 and the use of SE described in Algorithm 3.1. Let $\mathcal{A}$ be given. Suppose the agent network is connected under $\mathcal{A}$ at some time $k \in \mathbb{N}$. Then,*

$$V(k+h(k)) \leq \frac{(\frac{4\beta}{\alpha})^{M-1} \cdot M \cdot M!}{(\frac{4\beta}{\alpha})^{M-1} \cdot M \cdot M! + 1} V(k), \tag{3.30}$$

where α is any positive number satisfying $Q_i(\ell) - \alpha I \in \mathbb{S}_+^n$ $\forall \ell \in [k, k+h(k)]$ $\forall i \in \mathcal{M}(\ell)$.

Proof. See Appendix B.5. □

Note that in Lemma 3.2, the integer $k + h(k)$ is guaranteed to be finite due to Definition 3.1. In addition, it says that $V(k + h(k))$ is guaranteed to be strictly less than $V(k)$. However, even if $V(k)$ decreases asymptotically to zero, it does not imply that SE would converge. Indeed, with network connectivity alone, SE may not converge: In Example 3.4, we have $h(k) = 2$ if k is even and $h(k) = 3$ if k is odd. Hence, the agent network is connected under $\mathcal{A}$, while $z_1(k)$ fails to converge to z, due to the fact that $Q_1(k)$ keeps "losing" its positive definiteness.

Example 3.4 suggests that network connectivity *and* uniform positive definiteness of $\{Q_i(k)\}_{k\in\mathbb{N},i\in\mathcal{M}(k)}$ may be all that are required to establish the asymptotic convergence of SE. The following theorem shows that these two conditions are indeed sufficient:

Theorem 3.2. *Consider the agent network modeled by A1–A8 and the use of SE described in Algorithm 3.1. Let $\mathcal{A}$ be given. Suppose the agent network is connected under $\mathcal{A}$ and the sequence $\{Q_i(k)\}_{k\in\mathbb{N},i\in\mathcal{M}(k)}$ is uniformly positive definite under $\mathcal{A}$. Then, $z_i(k)$ asymptotically converges to z, i.e.,*

$$\forall\varepsilon > 0, \exists k \in \mathbb{N} \text{ } s.t. \text{ } \forall\ell \geq k, \forall j \in \mathcal{M}(\ell), \ \|z_j(\ell) - z\| < \varepsilon. \tag{3.31}$$

Proof. See Appendix B.6. □

Note that in Theorem 3.2, we write (3.31) instead of $\lim_{k\to\infty} z_i(k) = z$ because the former excludes cases where $z_i(k) = \#$, while the latter does not.

The following corollary is an immediate consequence of Theorem 3.2, just like Corollary 3.1 is for Theorem 3.1:

Corollary 3.2. *Let $\mathcal{A}$ be given. If the agent network is connected under $\mathcal{A}$ and the membership dynamics* (3.2) *are ultimately static under $\mathcal{A}$, then* (3.31) *holds.*

Proof. See Appendix B.7. □

Our final result provides sufficient conditions on the exponential convergence of SE, in terms of h^*:

Theorem 3.3. *Consider the agent network modeled by A1–A8 and the use of SE described in Algorithm 3.1. Let $\mathcal{A}$ be given. Suppose the agent network is uniformly connected under $\mathcal{A}$ and the sequence $\{Q_i(k)\}_{k\in\mathbb{N},i\in\mathcal{M}(k)}$ is uniformly positive definite under $\mathcal{A}$. Then, $z_i(k)$ exponentially converges to z in the following sense:*

$$\|z_j(\ell h^*) - z\|^2 \leq \frac{V(0)}{\alpha}\Big(\frac{(\frac{4\beta}{\alpha})^{M-1}\cdot M\cdot M!}{(\frac{4\beta}{\alpha})^{M-1}\cdot M\cdot M!+1}\Big)^{\ell}, \quad \forall \ell\in\mathbb{N},\ \forall j\in\mathcal{M}(\ell h^*), \tag{3.32}$$

where α is any positive number satisfying $Q_i(k)-\alpha I\in\mathbb{S}_+^n$ $\forall k\in\mathbb{N}$ $\forall i\in\mathcal{M}(k)$.

Proof. See Appendix B.8. □

Similar to the derivation of Corollaries 3.1 and 3.2, we derive the following corollary from Theorem 3.3:

Corollary 3.3. *Let $\mathcal{A}$ be given. Suppose the agent network is uniformly connected under $\mathcal{A}$ and the membership dynamics* (3.2) *are ultimately static under $\mathcal{A}$. Then,* (3.32) *holds for some $\alpha > 0$.*

Proof. See Appendix B.9. □

3.6 Illustrative Example

In this section, we illustrate the use of SE via an example. Consider an agent network consisting of $M = 100$ agents. Initially at time $k = 0$, a subset $\mathcal{F} = \{1, 2, \ldots, 50\}$ of 50 agents form a network and become its members, with each member i observing a randomly generated matrix $P_i \in \mathbb{S}_+^4$ and a vector $q_i \in \mathbb{R}^4$. The 50 agents wish to collaboratively solve an unconstrained quadratic program defined by the P_i's and q_i's. However, at each subsequent time $k = 1, 2, \ldots, 1000$, some of the members leave the network, while some of the non-members begin to join.

Figure 3.2 shows the simulation results for the agent network. The top portion of Figure 3.2 shows that the number of members fluctuates significantly between 35 to 55. The middle portion shows the actions taken over time by two selected agents, agent 1 and agent 51. Observe that both the agents join and leave the network as well as participate in interactions several times over the course of the simulation. Finally, the bottom portion represents, as a function of time, the norm of the difference between the unknown solution z and the solution estimate $z_i(k)$. Observe that, despite the volatility of the network membership dynamics, the differences between z between $z_i(k)$'s go to zero.

3.7 Conclusion

In this chapter, we have developed SE, a distributed asynchronous algorithm for solving symmetric positive definite systems of linear equations over networks of agents with arbitrary asynchronous interactions and spontaneous membership changes. To construct SE and analyze its behavior, we have introduced a time-varying quadratic Lyapunov-like function and several notions

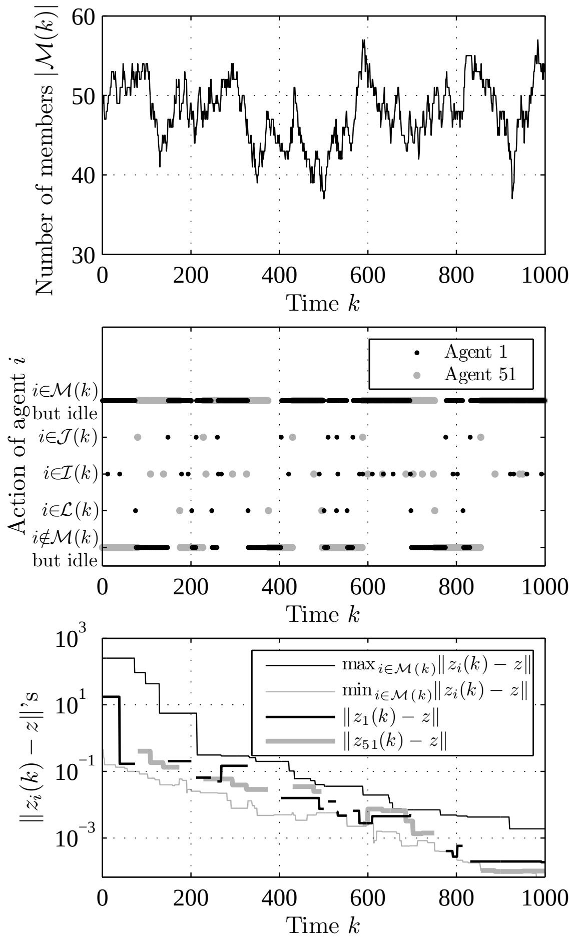


Figure 3.2: Simulation result illustrating the use of SE to perform unconstrained quadratic optimization over a volatile multi-agent system.

of network connectivity. Based on them, we have established necessary and/or sufficient conditions for its boundedness and convergence. Finally, the effectiveness of SE has been illustrated through a volatile multi-agent system.

Chapter 4 Distributed Algorithms for Solving Positive Definite Linear Equations over Wireless Networks

4.1 Introduction

In Chapter 3, we address the problem of solving symmetric positive definite systems of linear equations over networks, focusing on networks of agents with arbitrary asynchronous interactions and spontaneous membership dynamics. We develop *Subset Equalizing* (SE), a Lyapunov-based algorithm that overcomes some of the limitations facing the existing algorithms [66,76,77], and analyze its behavior, deriving conditions for establishing its boundedness and convergence.

In this chapter, we address the problem of solving such equations over multi-hop wireless networks with fixed topologies, focusing on the interplay among wireless communications, distributed algorithms, and control. Specifically, we address the following questions:

Q1. SE is developed without imposing any communication constraints among the agents (i.e., the agents have infinite bandwidth to exchange as much information as they choose to during each interaction), nor assuming that they are wirelessly connected. Thus, how does knowing that they are subject to such communication constraints and are wirelessly connected help the algorithm design, and how does ignoring these issues hurt the algorithm performance?

Q2. SE assumes that a new action involving a set of agents does not begin until the current one ends. For instance, it assumes that overlapping interactions cannot occur. However, wireless networks are inherently distributed systems, for which such issues may arise, especially when they are not accounted for. Hence, how could such issues be minimized through algorithm design?

Q3. Feedback control has been successfully utilized to control networks, including, but not limited to, power control for cellular systems [22, 37] and congestion control for networks [3, 35]. Therefore, is it possible to introduce some form of control in the algorithm design to improve its performance?

Although the current literature offers a large collection of distributed consensus [4, 21, 25, 50, 52, 63] and distributed averaging [8, 11, 13, 16, 26, 36, 38, 39, 53, 72, 74, 75] algorithms, few publications have considered questions Q1–Q3. Indeed, to the best of our knowledge, only question Q2 was addressed in [13, 36].

In this chapter, we show that there are significant benefits for *respecting wireless communications*, and that it is possible to introduce *feedback iteration control*, in the design of distributed algorithms. Building on the results from Chapter 3, we develop and analyze *Pairwise*, *Groupwise*, *Random Hopwise*, and *Controlled Hopwise Equalizing* (PE, GE, RHE, and CHE), providing along the way constructive answers to questions Q1–Q3. Specifically, we show how the broadcast nature of wireless transmissions may be fully utilized to enhance bandwidth/energy efficiency, how randomized gossip algorithms leave significant room for performance improvement, how undesirable overlapping iterations may be avoided, and how iterations may be feedback controlled in a greedy, decentralized, Lyapunov-based fashion. In addition, we show that CHE yields a networked dynamical system with state-dependent switching, establish its exponential convergence, characterize its bound on convergence rate, and derive a guaranteed bound on finite termination accuracy. Finally, through extensive simulation on random geometric graphs, we show that GE, RHE, and CHE are dramatically (six to ten times) more bandwidth/energy efficient and scalable than two existing, average-consensus-based schemes in [76, 77], with CHE having the best performance. This result suggests that there are tremendous benefits in exploiting the positive definite structure of the problem, compared to viewing the problem simply as an average-consensus problem.

The outline of this chapter is as follows: Section 4.2 describes the wireless network model and formulates the problem. Sections 4.3–4.5 introduce and analyze PE, GE, and RHE, respectively. Building upon RHE, Section 4.6 presents CHE, for which iterations are not randomized, but feedback controlled. In Section 4.7, PE, GE, RHE, and CHE are compared with MDW, MW, and flooding via simulation. Finally, the conclusion of this chapter is given in Section 4.8. All proofs are included in the Appendix. Throughout the chapter, let $\mathbb{N}$, $\mathbb{P}$, $\mathbb{S}_+^n$, and $|\cdot|$ denote, respectively, the sets of nonnegative integers, positive integers, $n \times n$ symmetric positive definite matrices over $\mathbb{R}$, and the cardinality of a set.

4.2 Problem Formulation

Consider a multi-hop wireless network consisting of $N \geq 2$ nodes, connected by L bidirectional links in a fixed topology. The network is modeled as a connected, undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, \ldots, N\}$ represents the set of N nodes (vertices) and $\mathcal{E} \subset \{\{i, j\} : i, j \in \mathcal{V}, i \neq j\}$ represents the set of L links (edges). Any two nodes $i, j \in \mathcal{V}$ are one-hop neighbors and can communicate if and only if $\{i, j\} \in \mathcal{E}$, and the set of one-hop neighbors of each node $i \in \mathcal{V}$ is denoted as $\mathcal{N}_i = \{j \in \mathcal{V} : \{i, j\} \in \mathcal{E}\}$. Each node $i \in \mathcal{V}$ observes a matrix $A_i \in \mathbb{S}_+^n$ and a vector $b_i \in \mathbb{R}^n$, and all the N nodes wish to determine the solution $x \in \mathbb{R}^n$ of the following symmetric positive definite system of linear equations:

$$\Big(\sum_{i\in\mathcal{V}} A_i\Big)x = \sum_{i\in\mathcal{V}} b_i. \tag{4.1}$$

Given the above model, the problem addressed in this chapter is how to construct a discrete-time asynchronous algorithm, with which each node $i \in \mathcal{V}$ repeatedly communicates with its one-hop neighbors, iteratively updates its estimate $\hat{x}_i \in \mathbb{R}$ of the unknown solution x in (4.1), and asymptotically drives $\hat{x}_i$ to x, while consuming bandwidth and energy efficiently.

The bandwidth/energy efficiency of an algorithm is measured by *the number of real-number transmissions it needs to drive all the $\hat{x}_i$'s to a sufficiently small neighborhood of x*, essentially solving (4.1). This quantity is a natural measure of efficiency because the smaller it is, the lesser bandwidth is occupied, the lesser energy is expended for communications, and the faster (4.1) may be solved. These, in turn, imply more bandwidth and time for other tasks, smaller probability of collision, longer lifetime for battery-powered nodes, and possible earlier return to sleep mode, all of which are desirable. The quantity also allows algorithms with different numbers of real-number transmissions per iteration to be fairly compared. Although, in networking, every message inevitably contains overhead (e.g., transmitter/receiver IDs and message type), we exclude such overhead when measuring efficiency since it is not inherent to an algorithm, may be reduced by piggybacking messages, and becomes negligible when the problem size n is large.

Similar to most existing work on distributed averaging [8, 11, 13, 16, 26, 30,31,36,38,39,52,53,64,69,72,74,75,78] and distributed consensus [4,21,24,25, 34, 40, 50, 52, 63, 67], we assume ideal internode communications, so that every message from each node $i \in \mathcal{V}$ is not subject to quantization, takes negligible time to transmit and propagate, and is received with negligible error. Moreover, since the nodes are wirelessly connected, the same message is received by every one-hop neighbor $j \in \mathcal{N}_i$, irrespective of the intended recipient(s).

4.3 Pairwise Equalizing

In this section, we present a simple algorithm for solving the problem formulated in Section 4.2. The algorithm is developed by first establishing the correspondence between the wireless network model of Section 4.2 and the agent network model from Chapter 3, followed by specializing the *Subset Equalizing* (SE) algorithm from Chapter 3 to this problem.

Recall that the agent network model from Chapter 3 operates as follows: At time $k = 0$, a subset $\mathcal{F}$ of the M agents form a network and make observations P_i and q_i $\forall i \in \mathcal{F}$. In addition, at each subsequent time $k \in \mathbb{P}$, a set $\mathcal{J}(k)$ of non-members join the network and interact with two sets $\mathcal{I}(k)$ and $\mathcal{L}(k)$ of existing members, after which the set $\mathcal{L}(k)$ of existing members leave the network. Thus, a simple way to associate this agent network model with the wireless network model described in Section 4.2 is to think of the set $\mathcal{V}$ of N nodes as the set of M agents, and the set $\mathcal{E}$ of L links as a set of physical means, through which the agents interact. Since $\mathcal{V}$ does not change over time, this association leads to the following correspondence between the two models: $M = N$, $\mathcal{F} = \mathcal{V}$, $\mathcal{J}(k) = \emptyset$, $\mathcal{L}(k) = \emptyset\ \forall k \in \mathbb{P}$, $P_i = A_i$, $q_i = b_i$, and $z = x$. With this correspondence, SE becomes

$$z_i(k) = \begin{cases} \Big(\sum\limits_{j \in \mathcal{I}(k)} A_j\Big)^{-1} \Big(\sum\limits_{j \in \mathcal{I}(k)} A_j z_j(k-1)\Big), & \text{if } i \in \mathcal{I}(k), \\ z_i(k-1), & \text{otherwise.} \end{cases} \tag{4.2}$$

$$Q_i(k) = A_i, \quad \forall i \in \mathcal{V}, \tag{4.3}$$

with initialization given by

$$z_i(0) = A_i^{-1} b_i, \tag{4.4}$$

$$Q_i(0) = A_i. \tag{4.5}$$

To specialize SE to the problem formulated in Section 4.2, the set $\mathcal{I}(k)$ in (4.2), characterizing the set of nodes that interact, must be specified at each time $k \in \mathbb{P}$. Given that nodes can only communicate directly with their one-hop neighbors, a natural choice is to let $\mathcal{I}(k) \in \mathcal{E}\ \forall k \in \mathbb{P}$. That is, at each time $k \in \mathbb{P}$, let a pair of one-hop neighbors i and j "gossip" with each other, performing a pairwise equalizing action according to (4.2). Since the set $\mathcal{E}$ consists of L links, there are L possibilities for $\mathcal{I}(k)$. The algorithm SE with $\mathcal{I}(k)$ defined as such leads to our first algorithm of this chapter, referred to as *Pairwise Equalizing* (PE) and stated as follows:

Algorithm 4.1 (Pairwise Equalizing)**.**

Initialization:

1. Each node $i \in \mathcal{V}$ transmits A_i to every node $j \in \mathcal{N}_i$.
2. Each node $i \in \mathcal{V}$ creates a variable $\hat{x}_i \in \mathbb{R}^n$ and initializes it: $\hat{x}_i \leftarrow A_i^{-1} b_i$.

Operation: At each iteration:

3. A link, say, link $\{i, j\}$, is selected randomly and equiprobably out of the set $\mathcal{E}$ of L links.
4. Node i transmits $\hat{x}_i$ to node j.
5. Node j updates $\hat{x}_j$: $\hat{x}_j \leftarrow (A_i + A_j)^{-1}(A_i\hat{x}_i + A_j\hat{x}_j)$.
6. Node j transmits $\hat{x}_j$ to node i.
7. Node i updates $\hat{x}_i$: $\hat{x}_i \leftarrow \hat{x}_j$. ■

Algorithm 4.1 consists of two stages: *initialization*, which is executed once, and *operation*, which is executed iteratively. Step 1 of the algorithm is needed to enable each node i to learn, once and for all, about the A_j of every one-hop neighbor $j \in \mathcal{N}_i$, so that it may carry out Step 5 later without having to query its one-hop neighbors for the same A_j's. Since the A_j's are symmetric and since the nodes are wirelessly connected, the number of real-number transmissions needed to realize Step 1 is $N\frac{n(n+1)}{2}$. In Step 2, each node i creates and maintains, in its local memory, a variable $\hat{x}_i$ representing its individual estimate of the unknown solution x. At each iteration k, Steps 3–7 are executed. In Step 3, a pair $\mathcal{I}(k)$ of one-hop neighbors is randomly and equiprobably selected to "gossip" with each other. Notice that other ways of selecting this pair are possible. For instance, the *Randomized Gossip Algorithm* [8] for distributed averaging first selects a node i and subsequently selects a one-hop neighbor j of node i to form the required pair, where both the selections are random but not necessarily equiprobable. Finally, Steps 4–7 define the pairwise equalizing action, through which $\hat{x}_i$ and $\hat{x}_j$ are equalized. Note that the number of real-number transmissions needed for these steps is $2n$.

PE defined in Algorithm 4.1 represents a generalization of three existing distributed averaging algorithms, namely, *Pairwise Averaging* [72], *Randomized*

Gossip Algorithm [8], and *Anti-Entropy Aggregation* [26, 39], in the sense that if $n = 1$ and $A_i = 1\ \forall i \in \mathcal{V}$, PE reduces to them.

Since PE is a specialization of SE with interactions limited to only randomly chosen pairs of adjacent nodes, one may expect its convergence behavior to be somewhat similar to that of SE. Indeed, the following theorem asserts that PE is almost surely asymptotically convergent:

Theorem 4.1. *Consider the wireless network modeled in Section 4.2 and the use of PE described in Algorithm 4.1. Then, with probability* 1, $\lim_{k\to\infty} \hat{x}_i(k) = x$, $\forall i \in \mathcal{V}$.

Proof. See Appendix C.1. □

4.4 Groupwise Equalizing

Although PE provides a provably convergent means to solve (4.1) over a multi-hop wireless network, it may have slow convergence and poor bandwidth/energy efficiency because it admits only *two* nodes at each iteration. Conceivably, admitting *more* nodes at a time can potentially speed up convergence and improve efficiency, since this allows more node estimates to be equalized at once. Hence, an alternative way of specializing SE, worthy of exploring, is to let each $\mathcal{I}(k)$ be a set consisting of more than two elements, whenever possible.

A simple way of ensuring that, at each iteration k, the set $\mathcal{I}(k)$ is larger with more elements is the following: At each iteration $k \in \mathbb{P}$, a node, say, node i, spontaneously forms a group with itself serving as the group leader. Upon forming, node i invites every one-hop neighbor $j \in \mathcal{N}_i$ to be group members and collaboratively perform a groupwise equalizing action, so that $\mathcal{I}(k) = \{i\} \cup \mathcal{N}_i$. Upon equalizing, the group is immediately disbanded. This alternative way of specializing SE leads to our second algorithm of this chapter, referred to as *Groupwise Equalizing* (GE):

Algorithm 4.2 (Groupwise Equalizing)**.**

Initialization:

1. Each node $i \in \mathcal{V}$ transmits A_i to every node $j \in \mathcal{N}_i$.
2. Each node $i \in \mathcal{V}$ creates a variable $\hat{x}_i \in \mathbb{R}^n$ and initializes it: $\hat{x}_i \leftarrow A_i^{-1} b_i$.

Operation: At each iteration:

3. A node, say, node i, is selected randomly and equiprobably out of the set $\mathcal{V}$ of N nodes.
4. Node i transmits a message to every node $j \in \mathcal{N}_i$, requesting their $\hat{x}_j$'s.
5. Each node $j \in \mathcal{N}_i$ transmits $\hat{x}_j$ to node i.
6. Node i updates $\hat{x}_i$:
   $\hat{x}_i \leftarrow (\sum_{j \in \{i\} \cup \mathcal{N}_i} A_j)^{-1} \sum_{j \in \{i\} \cup \mathcal{N}_i} A_j \hat{x}_j$.
7. Node i transmits $\hat{x}_i$ to every node $j \in \mathcal{N}_i$.
8. Each node $j \in \mathcal{N}_i$ updates $\hat{x}_j$: $\hat{x}_j \leftarrow \hat{x}_i$. ■

In Algorithm 4.2, Steps 1 and 2 are identical to those of PE, with Step 1 being needed to enable the nodes to carry out Step 6 later. Step 3 corresponds to the randomized selection of a group leader at each iteration. Steps 4–8 represent the groupwise equalizing action initiated by the group leader at each iteration, through which $\hat{x}_i$ and $\hat{x}_j$ $\forall j \in \mathcal{N}_i$ are equalized. Note that each iteration of GE requires $(|\mathcal{N}_i| + 1)n$ transmissions to realize, since $|\mathcal{N}_i|n$ are needed for Step 5 and n are needed for Step 7. Also note that the message broadcast by the group leader in Step 4 requires no real-number transmission since the message requires only a few bits to represent.

Similar to PE, GE defined in Algorithm 4.2 is a generalization of an existing distributed averaging algorithm, *Distributed Random Grouping* [13], reducing to the latter when $n = 1$ and $A_i = 1\ \forall i \in \mathcal{V}$. In addition, like PE, GE is also a specialization of SE with interactions limited to randomly chosen groups of nearby nodes. The following theorem characterizes the almost sure convergence of GE:

Theorem 4.2. *Consider the wireless network modeled in Section 4.2 and the use of GE described in Algorithm 4.2. Then, with probability* 1, $\lim_{k\to\infty} \hat{x}_i(k) = x$, $\forall i \in \mathcal{V}$.

Proof. See Appendix C.2. □

4.5 Random Hopwise Equalizing

Although PE and GE both solve the problem formulated in Section 4.2, they suffer from two drawbacks. First, both PE and GE do not fully exploit the broadcast nature of wireless medium. Specifically, when nodes i and j perform Steps 4 and 6 of PE, one-hop neighbors that overhear the two transmissions would simply discard the messages, leading to wasted receptions. The same can be said about GE: when each node $j \in \mathcal{N}_i$ sends its $\hat{x}_j$ to the group leader node i in Step 5, unintended one-hop neighbors of node j would also discard the overheard transmissions. Thus, it is of interest to investigate how the overheard information may be exploited and to what extent may such exploitation speed up convergence. Second, both PE and GE require multiple transmissions to complete an iteration. Specifically, PE requires two transmissions, in Steps 4 and 6, while GE requires $|\mathcal{N}_i| + 2$ transmissions, in Steps 4, 5, and 7, per iteration. Due to the distributed nature of the network, it is possible that before an iteration is completed, another iteration is initiated by a nearby node who is unaware of the ongoing iteration, thereby creating undesirable situations of overlapping iterations. Although this practical issue has been explicitly handled in [13, 36] in the context of distributed averaging, it is of interest to examine whether one can avoid this issue altogether, by limiting the number of transmissions to exactly one per iteration.

In this section, we present an algorithm that is capable of overcoming the two aforementioned drawbacks of PE and GE. The key idea here is to associate the agent network model of Chapter 3 with the wireless network model of this chapter in perhaps a little counterintuitive manner: think of the set of M agents *not* as the set $\mathcal{V}$ of N nodes, but instead as the set $\mathcal{E}$ of L fictitious, wireless links, and think of the set $\mathcal{V}$ of N nodes as physical means through which the agents interact. Since, physically, a link does not exist, an agent ℓ associated with a link $\{i, j\} \in \mathcal{E}$ is shared by both nodes i and j. With this agent-link association, the P_i's and q_i's in Chapter 3 may no longer be treated as the A_i's and b_i's, but rather as follows: if agent ℓ is associated with link $\{i, j\}$, then P_ℓ and q_ℓ are associated with $A_{\{i,j\}}$ and $b_{\{i,j\}}$, respectively, where they are defined as

$$A_{\{i,j\}} = \frac{1}{|\mathcal{N}_i|} A_i + \frac{1}{|\mathcal{N}_j|} A_j, \tag{4.6}$$

$$b_{\{i,j\}} = \frac{1}{|\mathcal{N}_i|} b_i + \frac{1}{|\mathcal{N}_j|} b_j, \quad \forall \{i, j\} \in \mathcal{E}. \tag{4.7}$$

Moreover, agent ℓ's estimate, z_ℓ, is associated with $x_{\{i,j\}}$, a new state variable that is conceptually shared by nodes i and j and locally maintained as x_{ij}

and x_{ji}, respectively, where both x_{ij} and x_{ji} are meant to be always equal upon completion of an iteration. Each node i, in addition to maintaining x_{ij}, $\forall j \in \mathcal{N}_i$, also maintains an estimate $\hat{x}_i$ of x, defined as

$$\hat{x}_i(k) = \Big(\sum_{j\in\mathcal{N}_i} A_{\{i,j\}}\Big)^{-1}\Big(\sum_{j\in\mathcal{N}_i} A_{\{i,j\}} x_{ij}(k)\Big),\ \forall k \in \mathbb{N}. \tag{4.8}$$

Finally, the sequence $\{\mathcal{I}(k)\}_{k=1}^{\infty}$ of interactions among subsets of agents is associated with the sequence of interactions among sets of links emanating from the same node. That is, each $\mathcal{I}(k)$ takes one of the following N possible values: $\{\{1,j\} \in \mathcal{E}\}, \{\{2,j\} \in \mathcal{E}\}, \ldots, \{\{N,j\} \in \mathcal{E}\}$. This agent-link association leads to *Random Hopwise Equalizing* (RHE), defined as follows:

Algorithm 4.3 (Random Hopwise Equalizing)**.**

Initialization:

1. Each node $i \in \mathcal{V}$ transmits $\frac{1}{|\mathcal{N}_i|}A_i$ and $\frac{1}{|\mathcal{N}_i|}b_i$ to every node $j \in \mathcal{N}_i$.
2. Each node $i \in \mathcal{V}$ creates variables $x_{ij} \in \mathbb{R}^n$ $\forall j \in \mathcal{N}_i$ and $\hat{x}_i \in \mathbb{R}^n$ and initializes them sequentially:
   $x_{ij} \leftarrow A_{\{i,j\}}^{-1}(\frac{1}{|\mathcal{N}_i|}b_i + \frac{1}{|\mathcal{N}_j|}b_j), \quad \forall j \in \mathcal{N}_i,$
   $\hat{x}_i \leftarrow (\sum_{j\in\mathcal{N}_i} A_{\{i,j\}})^{-1} \sum_{j\in\mathcal{N}_i} A_{\{i,j\}} x_{ij}.$

Operation: At each iteration:

3. A node, say, node i, is selected randomly and equiprobably out of the set $\mathcal{V}$ of N nodes.
4. Node i updates x_{ij} $\forall j \in \mathcal{N}_i$: $x_{ij} \leftarrow \hat{x}_i, \quad \forall j \in \mathcal{N}_i$.
5. Node i transmits $\hat{x}_i$ to every node $j \in \mathcal{N}_i$.
6. Each node $j \in \mathcal{N}_i$ updates x_{ji} and $\hat{x}_j$ sequentially:
   $x_{ji} \leftarrow \hat{x}_i,$
   $\hat{x}_j \leftarrow (\sum_{\ell\in\mathcal{N}_j} A_{\{j,\ell\}})^{-1} \sum_{\ell\in\mathcal{N}_j} A_{\{j,\ell\}} x_{j\ell}.$ ■

In Algorithm 4.3, Step 1, which requires $N\frac{n(n+1)}{2}+n)$ real-number transmissions to realize, is needed to enable the nodes to carry out Steps 2 and 6 later. In Step 2, each pair of one-hop neighbors i and j creates their local copies of x_{ij} and x_{ji}, along with their individual estimates $\hat{x}_i$ and $\hat{x}_j$ of the unknown solution x. Step 4 represents the hopwise equalizing action, where x_{ij} $\forall j \in \mathcal{N}_i$ are equalized to $\hat{x}_i$ at each iteration. With RHE, note that all the overheard information in Step 5 is fully utilized in Step 6. Also note that each iteration requires only a single transmission of n real-numbers, in Step 5, to complete. Hence, RHE circumvents the drawbacks of PE and GE by eliminating wasted receptions and avoiding overlapping iterations altogether.

Similar to both PE and GE, the almost sure convergence of RHE may be established as follows:

Theorem 4.3. *Consider the wireless network modeled in Section 4.2 and the use of RHE described in Algorithm 4.3. Then, with probability* 1, $\lim_{k\to\infty} x_{\{i,j\}}(k) = x$, $\forall \{i,j\} \in \mathcal{E}$ *and* $\lim_{k\to\infty} \hat{x}_i(k) = x$, $\forall i \in \mathcal{V}$.

Proof. See Appendix C.3. □

4.6 Controlled Hopwise Equalizing

The algorithms presented thus far, PE, GE, and RHE, all rely on randomized selection of links or nodes to initiate iterations, leading to stochastic networked dynamical systems. While this is simple, it offers significant rooms for improvement: the rate of convergence of an algorithm may be substantially increased if individual nodes are allowed to utilize locally maintained state variables as feedback to opportunistically control when to initiate iterations. In this subsection, using a control-theoretic approach, we show that the iterations can be feedback controlled with a suitable modification of RHE. We will begin with the assumption that a "genie" exists for controlling which node should initiate the next iteration, and will later relax this assumption.

4.6.1 Ideal Version

In Chapter 3, a time-varying quadratic, Lyapunov-like function is used to construct and analyze SE. In this chapter, this function reduces to

$$V(\mathbf{x}(k)) = \sum_{\{i,j\}\in\mathcal{E}} \Big(x_{\{i,j\}}(k) - x\Big)^T A_{\{i,j\}} \Big(x_{\{i,j\}}(k) - x\Big). \tag{4.9}$$

where $\mathbf{x}(k) \in \mathbb{R}^{Ln}$ is the state vector obtained by stacking the state variables $x_{\{i,j\}}(k)$, $\forall\{i,j\} \in \mathcal{E}$.

With RHE, whenever a node i initiates an iteration k, the value of the Lyapunov function $V(\mathbf{x}(k))$ would drop by an amount equal to

$$\begin{aligned} \Delta V_i(\mathbf{x}(k-1)) &= V(\mathbf{x}(k-1)) - V(\mathbf{x}(k)) \\ = \sum_{j\in\mathcal{N}_i} &\Big(x_{\{i,j\}}(k-1) - \hat{x}_i(k-1)\Big)^T A_{\{i,j\}} \Big(x_{\{i,j\}}(k-1) - \hat{x}_i(k-1)\Big). \end{aligned} \tag{4.10}$$

Note that the state variables appearing on the right-hand side of (4.10), i.e., quantifying the drop, are locally maintained by node i. Therefore, node i knows that if it spontaneously decides to initiate iteration k, the value of $V(\mathbf{x}(k))$, *whatever it may be*, would drop by an amount which it knows. This implies that every node i in the network at any given time knows by how much the value of $V(\mathbf{x}(k))$ would drop if it elects to become the node that initiates the next iteration. Hence, if there is a genie in the network that knows all the potential drops $\Delta V_i(\mathbf{x}(k-1))$, $\forall i \in \mathcal{V}$, the genie may choose to behave *greedily* and *always let the node that causes the largest drop in the value of* $V(\mathbf{x}(k))$ *initiate* *the next iteration.* The resulting algorithm, referred to as *Ideal Controlled Hopwise Equalizing* (ICHE), is defined below:

Algorithm 4.4 (Ideal Controlled Hopwise Equalizing)**.**

Initialization:

1. Each node $i \in \mathcal{V}$ transmits $\frac{1}{|\mathcal{N}_i|}A_i$ and $\frac{1}{|\mathcal{N}_i|}b_i$ to every node $j \in \mathcal{N}_i$.
2. Each node $i \in \mathcal{V}$ creates variables $x_{ij} \in \mathbb{R}^n$ $\forall j \in \mathcal{N}_i$, $\hat{x}_i \in \mathbb{R}^n$, and $\Delta V_i \in [0, \infty)$ and initializes them sequentially:

   $x_{ij} \leftarrow A_{\{i,j\}}^{-1}(\frac{1}{|\mathcal{N}_i|}b_i + \frac{1}{|\mathcal{N}_j|}b_j), \quad \forall j \in \mathcal{N}_i,$

   $\hat{x}_i \leftarrow (\sum_{j\in\mathcal{N}_i} A_{\{i,j\}})^{-1} \sum_{j\in\mathcal{N}_i} A_{\{i,j\}} x_{ij},$

   $\Delta V_i \leftarrow \sum_{j\in\mathcal{N}_i} (x_{ij} - \hat{x}_i)^T A_{\{i,j\}} (x_{ij} - \hat{x}_i).$

Operation: At each iteration:

3. Let $i \in \arg\max_{j\in\mathcal{V}} \Delta V_j$.
4. Node i updates x_{ij} $\forall j \in \mathcal{N}_i$ and ΔV_i sequentially:

   $x_{ij} \leftarrow \hat{x}_i, \quad \forall j \in \mathcal{N}_i,$

   $\Delta V_i \leftarrow 0.$
5. Node i transmits $\hat{x}_i$ to every node $j \in \mathcal{N}_i$.
6. Each node $j \in \mathcal{N}_i$ updates x_{ji}, $\hat{x}_j$, and ΔV_j sequentially:

   $x_{ji} \leftarrow \hat{x}_i,$

   $\hat{x}_j \leftarrow (\sum_{\ell\in\mathcal{N}_j} A_{\{j,\ell\}})^{-1} \sum_{\ell\in\mathcal{N}_j} A_{\{j,\ell\}} x_{j\ell},$

   $\Delta V_j \leftarrow \sum_{\ell\in\mathcal{N}_j} (x_{j\ell} - \hat{x}_j)^T A_{\{j,\ell\}} (x_{j\ell} - \hat{x}_j).$ ■

Note that ICHE is very similar to RHE, except that each node i maintains an additional ΔV_i, which it passes to the genie for deciding which node should initiate the next iteration. The genie's decision then manifests itself in Step 3, where the node with the largest ΔV_i will be selected to initiate the next iteration.

Because the selection of nodes to initiate iterations is not randomized but depends on the state variables, ICHE produces a state-dependent, switched dynamical system.

$$x_{\{i,j\}}(k) = \begin{cases} (\sum_{\ell \in \mathcal{N}_{u(k)}} A_{\{u(k),\ell\}})^{-1} \sum_{\ell \in \mathcal{N}_{u(k)}} A_{\{u(k),\ell\}} x_{\{u(k),\ell\}}(k-1), & \text{if } u(k) \in \{i,j\}, \\ x_{\{i,j\}}(k-1), & \text{otherwise}, \end{cases} \tag{4.11}$$

$$u(k) = \arg\max_{j \in \mathcal{V}} \Delta V_j(\mathbf{x}(k-1)), \tag{4.12}$$

where $u(k)$ represents the node with the largest $\Delta V_j(\mathbf{x}(k-1))$. The following theorem shows that ICHE achieves asymptotic convergence:

Theorem 4.4. *Consider the wireless network modeled in Section 4.2 and the use of ICHE described in Algorithm 4.4. Then,* $\lim_{k\to\infty} x_{\{i,j\}}(k) = x$, $\forall \{i,j\} \in \mathcal{E}$ *and* $\lim_{k\to\infty} \hat{x}_i(k) = x$, $\forall i \in \mathcal{V}$.

Proof. See Appendix C.4. □

A bound on the exponential convergence rate of ICHE can be calculated using the following theorem:

Theorem 4.5. *Consider the wireless network modeled in Section 4.2 and the use of ICHE described in Algorithm 4.4. Suppose* $L \geq 2$. *Then,*

$$V(\mathbf{x}(k)) \leq \rho V(\mathbf{x}(k-1)), \quad \forall k \in \mathbb{P}, \tag{4.13}$$

and $\exists \mathbf{x}(k-1) \in \mathbb{R}^{Ln}$ *such that*

$$V(\mathbf{x}(k)) = \rho V(\mathbf{x}(k-1)),$$

where $\rho \in [0,1)$ is such that $\frac{1}{1-\rho}$ is the optimal value of the following convex maximization problem:

$$\begin{array}{lll} \text{maximize}_{\mathbf{z}\in\mathbb{R}^{Ln}} & V(\mathbf{z}) & \\ \text{subject to} & \Delta V_i(\mathbf{z}) \leq 1, \quad \forall i \in \mathcal{V} & \\ & \sum_{\{i,j\}\in\mathcal{E}} A_{\{i,j\}} z_{\{i,j\}} = 0. & \end{array} \tag{4.14}$$

Proof. See Appendix C.5. □

4.6.2 Practical Version

Obviously, ICHE is not implementable since it assumes the presence of a genie. Fortunately, it is possible to closely mimic the greedy behavior of ICHE with a practical, decentralized controller. To describe this controller, the notion of time $t \geq 0$ is needed. Suppose each node i maintains a variable $\tau_i > 0$ representing the time-to-initiate for node i, so that when time $t = \tau_i$, node i initiates the next iteration. Suppose also that τ_i is the following function of ΔV_i:

$$\tau_i(k-1) = \max\{\Phi(\Delta V_i(\mathbf{x}(k-1))), t\} + \varepsilon(\Delta V_i(\mathbf{x}(k-1))) \cdot \text{rand}(), \tag{4.15}$$

where $\Phi : [0,\infty) \to (0,\infty]$ is a continuous, strictly decreasing function satisfying $\lim_{v\to 0} \Phi(v) = \infty$ and $\Phi(0) = \infty$, $\varepsilon : [0,\infty) \to (0,\infty)$ is a continuous function meant to take on a small positive value, and rand() is a number returned by a call to a uniformly distributed pseudo-random number generator on $[0,1]$. Since Φ is strictly decreasing, $\Delta V_i(\mathbf{x}(k-1))$ is inversely proportional to $\tau_i(k-1)$. Hence, with (4.15), the node with the largest $\Delta V_i(\mathbf{x}(k-1))$'s would have the smallest τ_i's and, thus, would likely become the node that initiates iteration k. The resulting algorithm, referred to simply as *Controlled Hopwise Equalizing* (CHE), is defined below:

Algorithm 4.5 (Controlled Hopwise Equalizing)**.**

Initialization:

1. Let time $t = 0$.
2. Each node $i \in \mathcal{V}$ transmits $\frac{1}{|\mathcal{N}_i|}A_i$ and $\frac{1}{|\mathcal{N}_i|}b_i$ to every node $j \in \mathcal{N}_i$.
3. Each node $i \in \mathcal{V}$ creates variables $x_{ij} \in \mathbb{R}^n$ $\forall j \in \mathcal{N}_i$, $\hat{x}_i \in \mathbb{R}^n$, $\Delta V_i \in [0, \infty)$, and $\tau_i \in (0, \infty]$ and initializes them sequentially:

   $x_{ij} \leftarrow A_{\{i,j\}}^{-1}(\frac{1}{|\mathcal{N}_i|}b_i + \frac{1}{|\mathcal{N}_j|}b_j), \quad \forall j \in \mathcal{N}_i,$

   $\hat{x}_i \leftarrow (\sum_{j\in\mathcal{N}_i} A_{\{i,j\}})^{-1} \sum_{j\in\mathcal{N}_i} A_{\{i,j\}} x_{ij},$

   $\Delta V_i \leftarrow \sum_{j\in\mathcal{N}_i} (x_{ij} - \hat{x}_i)^T A_{\{i,j\}} (x_{ij} - \hat{x}_i),$

   $\tau_i \leftarrow \max\{\Phi(\Delta V_i), t\} + \varepsilon(\Delta V_i) \cdot \mathrm{rand}().$

Operation: At each iteration:

4. Let $i \in \arg\min_{j\in\mathcal{V}} \tau_j$ and $t = \tau_i$.
5. Node i updates x_{ij} $\forall j \in \mathcal{N}_i$, ΔV_i, and τ_i sequentially:

   $x_{ij} \leftarrow \hat{x}_i, \quad \forall j \in \mathcal{N}_i,$

   $\Delta V_i \leftarrow 0,$

   $\tau_i \leftarrow \infty.$
6. Node i transmits $\hat{x}_i$ to every node $j \in \mathcal{N}_i$.
7. Each node $j \in \mathcal{N}_i$ updates x_{ji}, $\hat{x}_j$, ΔV_j, and τ_j sequentially:

   $x_{ji} \leftarrow \hat{x}_i,$

   $\hat{x}_j \leftarrow (\sum_{\ell\in\mathcal{N}_j} A_{\{j,\ell\}})^{-1} \sum_{\ell\in\mathcal{N}_j} A_{\{j,\ell\}} x_{j\ell},$

   $\Delta V_j \leftarrow \sum_{\ell\in\mathcal{N}_j} (x_{j\ell} - \hat{x}_j)^T A_{\{j,\ell\}} (x_{j\ell} - \hat{x}_j),$

   $\tau_j \leftarrow \max\{\Phi(\Delta V_j), t\} + \varepsilon(\Delta V_j) \cdot \mathrm{rand}().$ ■

In some applications, nodes may want to terminate the algorithm execution as soon as a desired level of accuracy is achieved. The following theorem provides a guaranteed bound on the termination accuracy, assuming that the termination criterion is $\Delta V_i(K) \leq \gamma$, $\forall i \in \mathcal{V}$, for some $\gamma > 0$. To describe the result, for each $\{i,j\} \in \mathcal{E}$, let α_{ij}, β_{ij} be, respectively, the smallest and largest eigenvalue of $A_{\{i,j\}}$. For each $i \in \mathcal{V}$, let α'_i, β'_i be, respectively, the smallest and largest eigenvalue of $\sum_{j \in \mathcal{N}_i} A_{\{i,j\}}$. Then, let

$$\alpha = \min_{\{i,j\} \in \mathcal{E}} \{\alpha_{ij}\}, \quad \beta = \max_{\{i,j\} \in \mathcal{E}} \{\beta_{ij}\}, \tag{4.16}$$

$$\alpha' = \min_{i \in \mathcal{V}} \{\alpha'_i\}, \quad \beta' = \max_{i \in \mathcal{V}} \{\beta'_i\}. \tag{4.17}$$

Theorem 4.6. *Consider the wireless network modeled in Section 4.2 and the use of CHE described in Algorithm 4.5. Let K be such that $\Delta V_i(K) \leq \gamma$, $\forall i \in \mathcal{V}$. Then,*

$$\sum_{i \in \mathcal{V}} \|\hat{x}_i(K) - x\|^2 \leq \frac{4\beta'}{\alpha\alpha'}(N-2)^2(N-1)\gamma. \tag{4.18}$$

Proof. See Appendix C.6. □

4.7 Performance Comparison

In this section, we compare PE, GE, RHE, and CHE with the two schemes proposed in [76, 77], namely, *Maximum-Degree Weights* (MDW) and *Metropolis Weights* (MW) as well as with *flooding*.

4.7.1 Method of Comparison

To compare the aforementioned algorithms, we simulate them on wireless networks modeled by random geometric graphs, with nodes trying to solve randomly generated, symmetric positive definite systems of linear equations (4.1). To generate a network with N nodes and an average of $\frac{2L}{N}$ one-hop neighbors per node, we randomly and equiprobably place each of the N nodes on the unit square $[0,1] \times [0,1]$ and gradually increase the one-hop transmission radius from zero, until the average number of neighbors becomes $\frac{2L}{N}$. If the resulting network is not connected, it is discarded and the above process is repeated. To generate a positive definite system of linear equations of size n, we factor each $A_i \in \mathbb{R}^{n\times n}$ as $A_i = X_i^T X_i$ and let both $X_i \in \mathbb{R}^{n\times n}$ and $b_i \in \mathbb{R}^n$ have normally distributed random entries with zero mean and unit variance. Therefore, each simulation is defined by three parameters: the number of nodes N, the average number of neighbors $\frac{2L}{N}$, and the problem size n. To understand how each of these parameters affects performance, we carry out three sets of simulations, each corresponds to varying one of the three parameters and keeping the other two constant. For each algorithm and each simulation run, we record the number of real-number transmissions needed for the algorithm to converge, where the convergence criterion is $\max_{i\in\mathcal{V}} \|\hat{x}_i - x\| < 0.005$. For CHE, we let $\Phi(\Delta V_i) = \frac{1}{\Delta V_i}$ and $\varepsilon(\Delta V_i) = 0.001$.

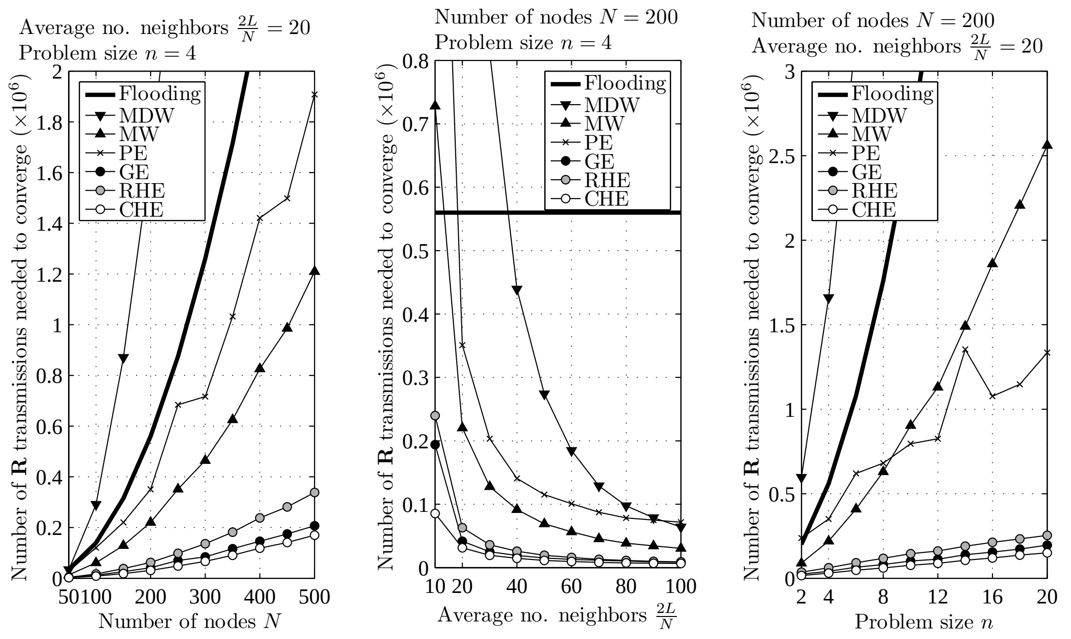
4.7.2 Results of Comparison

Figure 4.1(a) shows the first set of simulation results, where we let the number of nodes N vary from 50 to 500, while fixing the average number of neighbors at $\frac{2L}{N} = 20$ and problem size at $n = 4$. From this figure, we observe that regardless of the number of nodes N, MDW has the worst bandwidth/energy efficiency, requiring a very large number of real-number transmissions to converge—even more so than the worst possible scheme that is flooding. In addition, we see that GE, RHE, and CHE are dramatically more efficient than PE and MW, with CHE being uniformly the most efficient. GE, RHE, and CHE also exhibit much better scaling with respect to N compared with PE and MW. Finally, we observe that CHE is about twice more efficient than RHE. This represents the amount of improvement that can be achieved by letting nodes greedily control, as opposed to randomly decide, when to initiate an iteration.

Figure 4.1(b) shows the second set of results, where we let the average number of neighbors $\frac{2L}{N}$ vary from 10 to 100, while fixing the number of nodes at $N = 200$ and problem size at $n = 4$. Observe from the figure that, generally, the sparser the network is, the worse the algorithms perform, except for flooding, the performance of which is independent of network density. Also observe that the trend seen in the Figure 4.1(a) holds here: MDW is the least efficient, followed by PE and MW. As before, GE, RHE, and CHE are quite efficient, with CHE again having the best efficiency. In particular, when the network is sparse with, say, $\frac{2L}{N} = 10$, CHE is twice more efficient than GE. However, when the network is dense, say, with $\frac{2L}{N} \geq 70$, RE, GE, and CHE have indistinguishable performances.

Figure 4.1(c) shows the third set of results, where we let the problem size n vary from 2 to 20, while fixing the number of nodes at $N = 200$ and average number of neighbors at $\frac{2L}{N} = 20$. Observe from the figure that the trend seen in the previous two figures also holds here. Also observe that GE, RHE, and CHE again exhibit significantly better scaling with respect to n compared with PE and MW since their real-number transmissions seem to be proportional with n.

To summarize, Figure 4.1 suggests that GE, RHE, and CHE are highly bandwidth/energy efficient, compared with MDW and MW. The figure also suggests that it is advantageous to be able to fully exploit the broadcast nature of wireless communications and decide when to initiate an iteration via



(a) Varying number of nodes N

(b) Varying average number of neighbors $\frac{2L}{N}$

(c) Varying problem size n

Figure 4.1: Performance comparison on multi-hop wireless networks with varying number of nodes N, varying average number of neighbors $\frac{2L}{N}$, and varying problem size n.

decentralized feedback control.

4.8 Conclusion

In this chapter, we have introduced and analyzed a collection of algorithms for solving symmetric positive definite systems of linear equations over multi-hop wireless networks with fixed topologies. We have demonstrated that it is possible, and highly beneficial, to bring together wireless communications, distributed algorithms, and control. We have also established various convergence properties of the proposed algorithms. Finally, we have shown through extensive simulation on random geometric graphs that GE, RHE, and CHE are dramatically more efficient and scalable than MDW, MW, and flooding.

Chapter 5 Gossip Algorithms for Distributed Convex Optimization

5.1 Introduction

Consider an N-node multi-hop network, where each node i observes a convex function f_i, and all the N nodes wish to determine an optimal consensus x^*, which minimizes the sum of the f_i's:

$$x^* \in \arg\min_x \sum_{i=1}^{N} f_i(x). \tag{5.1}$$

Since each node i knows only its own f_i, the nodes cannot individually compute the optimal consensus x^* and, thus, must collaborate to do so. This problem of achieving unconstrained, separable, convex consensus optimization has many applications in multi-agent systems and wired/wireless/social networks [57,65].

The current literature offers a large body of work on distributed consensus (see [50] for a survey), including a line of research that focuses on solving problem (5.1) for an optimal consensus x^* [27,28,32,42–47,57–62,65]. This line of work has resulted in a family of discrete-time subgradient algorithms, including the *incremental* subgradient algorithms [28,42–44,57–59,61,65], whereby an estimate of x^* is passed around the network, and the *non-incremental* ones [27,32,45–47,60,62], whereby each node maintains an estimate of x^* and updates it iteratively by exchanging information with neighbors.

Although the aforementioned subgradient algorithms are capable of solving problem (5.1) under fairly weak assumptions, they suffer from one or more of the following limitations:

L1. *Stepsizes*: The algorithms require selection of stepsizes, which may be constant, diminishing, or dynamic. In general, constant stepsizes ensure only convergence to neighborhoods of x^*, rather than to x^* itself. Moreover, they present an inevitable trade-off: larger stepsizes tend to yield larger convergence neighborhoods, while smaller ones tend to yield slower convergence. In contrast, diminishing stepsizes typically ensure asymptotic convergence. However, the convergence may be very slow, since the stepsizes may diminish too quickly. Finally, dynamic stepsizes allow shaping of the convergence behavior [42,43]. Unfortunately, their dynamics depend on global information that is often costly to obtain. Hence, selecting appropriate stepsizes is not a trivial task, and inappropriate choices can cause poor performance.

L2. *Hamiltonian cycle*: Many incremental subgradient algorithms [42–44, 57–59, 61, 65] require the nodes to construct and maintain a Hamiltonian cycle (i.e., a closed path that visits every node exactly once) or a pseudo one (i.e., that allows multiple visits), which may be very difficult to carry out, especially in a decentralized, leaderless fashion.

L3. *Multi-hop transmissions*: Some incremental subgradient algorithms [42–44] require the node that has the latest estimate of x^* to pass it on to a randomly and equiprobably chosen node in the network. This implies that every node must be aware of all the nodes in the network, and the algorithms must run alongside a routing protocol that enables such passing, which may not always be the case. The fact that the chosen node is typically multiple hops away also implies that these algorithms are communication inefficient, requiring plenty of transmissions (up to the network diameter) just to complete a single iteration.

L4. *Lack of asymptotic convergence*: A variety of convergence properties have been established for the subgradient algorithms in [27,28,32,42–47,57–62, 65], including error bounds, convergence in expectations, convergence in limit inferiors, convergence rates, etc. In contrast, relatively few asymptotic convergence results have been reported, except for the subgradient algorithms with diminishing or dynamic stepsizes in [42–44, 59–62].

Limitations L1–L4 facing the subgradient algorithms raise the question of whether it is possible to devise algorithms, which require neither the notion of a stepsize, the construction of a (pseudo-)Hamiltonian cycle, nor the use of a routing protocol for multi-hop transmissions, and yet guarantee asymptotic convergence, bypassing L1–L4. In this chapter, we show that, for the *one-dimensional* case and with a few mild assumptions, such algorithms can be constructed. Specifically, instead of letting the network be directed, we assume that it is undirected, with possibly a time-varying topology unknown to any of the nodes. In addition, instead of letting each f_i in (5.1) be convex but not necessarily differentiable, we assume that it is strictly convex, continuously differentiable, and has a minimizer. Based on these assumptions, we develop two gossip-style, distributed asynchronous iterative algorithms, referred to as *Pairwise Equalizing* (PE) and *Pairwise Bisectioning* (PB), which not only solve problem (5.1) and circumvent limitations L1–L4, but also are rather easy to implement—although computationally they are more demanding than the subgradient algorithms.

As will be shown in the chapter, PE and PB exhibit a number of notable features. First, they produce switched, nonlinear, networked dynamical systems whose state evolves along an invariant manifold whenever nodes gossip with each other. The switched systems are proved, using Lyapunov stability theory, to be asymptotically convergent, as long as the gossiping pattern is sufficiently rich. In particular, we show that the first-order convexity condition [1] can be used to form a common Lyapunov function, as well as to characterize drops in its value after every gossip. Second, PE and PB do not belong to the family of subgradient algorithms as they utilize fundamentally different, non-gradient-based update rules that involve no stepsize. These update rules are synthesized from two simple ideas—*conservation* and *dissipation*—which are somewhat similar to how Pairwise Averaging [72] was conceived back in the 1980s. Indeed, we show that PE reduces to Pairwise Averaging [72] and Randomized Gossip Algorithm [8] when problem (5.1) specializes to an averaging problem. In addition, we show that PE can be extended to handle networks with both time-varying topologies and time-varying node memberships, where nodes may freely join and leave. Finally, PE requires one-time sharing of the f_i's between gossiping nodes, which may be costly or impermissible in some applications. This requirement is eliminated by PB at the expense of more communications per iteration.

The outline of this chapter is as follows: Section 5.2 formulates the distributed convex optimization problem. Section 5.3 describes the proposed algorithm PE, while Section 5.4 illustrates the effectiveness of PE through an example. Section 5.5 presents an extension of PE. Section 5.6 introduces

[1] Also known as Bregman divergence [10].

another proposed algorithm PB. Finally, Section 5.7 concludes the chapter. The proofs of the main results are included in Appendix D. Throughout this chapter, let $\mathbb{N}$ and $\mathbb{P}$ denote, respectively, the sets of nonnegative and positive integers.

5.2 Problem Formulation

Consider a multi-hop network consisting of $N \geq 2$ nodes, connected by bidirectional links in a time-varying topology. The network is modeled as an undirected graph $\mathcal{G}(k) = (\mathcal{V}, \mathcal{E}(k))$, where $k \in \mathbb{N}$ denotes time, $\mathcal{V} = \{1, 2, \ldots, N\}$ represents the set of N nodes (vertices), and $\mathcal{E}(k) \subset \{\{i, j\} : i, j \in \mathcal{V}, i \neq j\}$ represents the nonempty set of links (edges) at time k. The graph $\mathcal{G}(k)$ is allowed to vary in order to reflect node mobility and changing channel conditions, and the variations are assumed to be exogenous, beyond control of the nodes. Any two nodes $i, j \in \mathcal{V}$ are one-hop neighbors and can communicate at time $k \in \mathbb{N}$ if and only if $\{i, j\} \in \mathcal{E}(k)$, and the communications are assumed to be delay- and error-free, with no quantization.

Suppose, at time $k = 0$, each node $i \in \mathcal{V}$ observes a function $f_i : \mathcal{X} \to \mathbb{R}$, which maps a nonempty open interval $\mathcal{X} \subset \mathbb{R}$ to $\mathbb{R}$, and which satisfies the following assumption:

Assumption 5.1. For each $i \in \mathcal{V}$, the function f_i is strictly convex, continuously differentiable, and has a minimizer $x_i^* \in \mathcal{X}$.

Note that the conditions in Assumption 5.1 are not redundant, as strict convexity alone does not imply continuous differentiability (e.g., with $\mathcal{X} = \mathbb{R}$, $f_i(x) = e^{|x|}$ is strictly convex but not differentiable at $x = 0$), and strict convexity and continuous differentiability together do not imply the existence of a minimizer in $\mathcal{X}$ (e.g., with $\mathcal{X} = (0,1)$, $f_i(x) = e^{-x}$ is strictly convex and continuously differentiable but has no minimizer in $\mathcal{X}$). On the other hand, strict convexity and the existence of a minimizer $x_i^* \in \mathcal{X}$ do ensure that the minimizer x_i^* is unique.

Suppose, upon observing the f_i's, all the N nodes wish to solve the following unconstrained, separable, convex optimization problem:

$$\min_{x \in \mathcal{X}} F(x), \tag{5.2}$$

where the function $F : \mathcal{X} \to \mathbb{R}$ is defined as $F(x) = \sum_{i \in \mathcal{V}} f_i(x)$. Clearly, F is strictly convex and continuously differentiable. To show that F has a unique minimizer in $\mathcal{X}$ so that problem (5.2) is well-posed, let $f_i' : \mathcal{X} \to \mathbb{R}$ and $F' : \mathcal{X} \to \mathbb{R}$ denote the derivatives of f_i and F, respectively, and consider the following lemma and proposition:

Lemma 5.1. *Let $g_i : \mathcal{X} \to \mathbb{R}$ be a strictly increasing and continuous function and $z_i \in \mathcal{X}$ for $i = 1, 2, \ldots, n$. Then, there exists a unique $z \in \mathcal{X}$ such that $\sum_{i=1}^n g_i(z) = \sum_{i=1}^n g_i(z_i)$. Moreover, $z \in [\min_{i\in\{1,2,\ldots,n\}} z_i, \max_{i\in\{1,2,\ldots,n\}} z_i]$.*

Proof. Since g_i is strictly increasing and continuous $\forall i \in \{1, 2, \ldots, n\}$, so is $\sum_{i=1}^n g_i : \mathcal{X} \to \mathbb{R}$. Thus,

$$\sum_{i=1}^n g_i(\min_{j\in\{1,2,\ldots,n\}} z_j) \le \sum_{i=1}^n g_i(z_i) \le \sum_{i=1}^n g_i(\max_{j\in\{1,2,\ldots,n\}} z_j).$$

It follows from the Intermediate Value Theorem that there exists a unique $z \in \mathcal{X}$ such that $\sum_{i=1}^n g_i(z) = \sum_{i=1}^n g_i(z_i)$, and that $z \in [\min\limits_{i\in\{1,2,\ldots,n\}} z_i, \max\limits_{i\in\{1,2,\ldots,n\}} z_i]$. □

Proposition 5.1. *With Assumption 5.1, there exists a unique $x^* \in \mathcal{X}$, which satisfies $F'(x^*) = 0$, minimizes F over $\mathcal{X}$, and solves problem (5.2), i.e., $x^* = \arg\min_{x\in\mathcal{X}} F(x)$.*

Proof. By Assumption 5.1, for every $i \in \mathcal{V}$, f_i' is strictly increasing and continuous. By Lemma 5.1, there exists a unique $x^* \in \mathcal{X}$ such that $\sum_{i\in\mathcal{V}} f_i'(x^*) = \sum_{i\in\mathcal{V}} f_i'(x_i^*)$. Since $F' = \sum_{i\in\mathcal{V}} f_i'$ and $f_i'(x_i^*) = 0\ \forall i \in \mathcal{V}$, $F'(x^*) = 0$. Since F is strictly convex, x^* minimizes F over $\mathcal{X}$, solving (5.2). $\square$

Given the above network and problem, the goal of this chapter is to construct a distributed asynchronous algorithm, with which each node $i \in \mathcal{V}$ repeatedly communicates with its one-hop neighbors, iteratively updates its estimate $\hat{x}_i$ of the unknown optimizer x^*, and asymptotically drives $\hat{x}_i$ to x^*. The algorithm should be easy to implement and free of limitations L1–L4 discussed in Section 5.1.

5.3 Pairwise Equalizing

In this section, we develop a gossip algorithm having the aforementioned features.

Suppose, at time $k = 0$, each node $i \in \mathcal{V}$ creates a state variable $\hat{x}_i \in \mathcal{X}$ in its local memory, which represents its estimate of the unknown optimizer x^*. Also suppose, at each subsequent time $k \in \mathbb{P}$, an iteration involving a subset of the N nodes, referred to as *iteration* k, takes place. Let $\hat{x}_i(0)$ represent the initial value of $\hat{x}_i$, and $\hat{x}_i(k)$ its value upon completing each iteration $k \in \mathbb{P}$. With this setup, the goal of asymptotically driving all the $\hat{x}_i(k)$'s to x^* may be stated as

$$\lim_{k\to\infty} \hat{x}_i(k) = x^*, \quad \forall i \in \mathcal{V}. \tag{5.3}$$

To design an algorithm that guarantees (5.3), consider a *conservation* *condition*

$$\sum_{i\in\mathcal{V}} f_i'(\hat{x}_i(k)) = 0, \quad \forall k \in \mathbb{N}, \tag{5.4}$$

which says that the state variables $\hat{x}_i(k)$'s evolve in such a manner that the sum of the derivatives f_i''s, evaluated respectively at the $\hat{x}_i(k)$'s, is always conserved at zero. Moreover, consider a *dissipation condition*

$$\lim_{k\to\infty} \hat{x}_i(k) = \tilde{x}, \quad \forall i \in \mathcal{V}, \text{ for some } \tilde{x} \in \mathcal{X}, \tag{5.5}$$

which says that the $\hat{x}_i(k)$'s gradually dissipate their differences and asymptotically achieve some arbitrary consensus $\tilde{x} \in \mathcal{X}$. Note that if the conservation condition (5.4) is met, then

$$\lim_{k\to\infty} \sum_{i\in\mathcal{V}} f_i'(\hat{x}_i(k)) = \lim_{k\to\infty} 0 = 0. \tag{5.6}$$

If, in addition, the dissipation condition (5.5) is met, then due to the continuity of every f_i',

$$\sum_{i\in\mathcal{V}} \lim_{k\to\infty} f_i'(\hat{x}_i(k)) = \sum_{i\in\mathcal{V}} f_i'(\lim_{k\to\infty} \hat{x}_i(k)) = \sum_{i\in\mathcal{V}} f_i'(\tilde{x}) = F'(\tilde{x}). \tag{5.7}$$

Because $\lim_{k\to\infty} f_i'(\hat{x}_i(k))$ exists for every $i \in \mathcal{V}$, we can write

$$\lim_{k\to\infty} \sum_{i\in\mathcal{V}} f_i'(\hat{x}_i(k)) = \sum_{i\in\mathcal{V}} \lim_{k\to\infty} f_i'(\hat{x}_i(k)). \tag{5.8}$$

Combining (5.6), (5.7), and (5.8), we obtain $F'(\tilde{x}) = 0$. From Proposition 5.1, we see that the arbitrary consensus $\tilde{x}$ must be the unknown optimizer x^*, i.e., $\tilde{x} = x^*$, so that (5.3) holds. Therefore, to design an algorithm that ensures (5.3)—where x^* explicitly appears, it suffices to make the algorithm satisfy both the conservation and dissipation conditions (5.4) and (5.5)—where x^* is implicitly encoded.

To come up with such an algorithm, observe that the conservation condition (5.4) holds if and only if the initial values $\hat{x}_i(0)$'s are such that

$$\sum_{i\in\mathcal{V}} f_i'(\hat{x}_i(0)) = 0, \tag{5.9}$$

and the values $\hat{x}_i(k)$'s upon completing each iteration $k \in \mathbb{P}$ are related to the values $\hat{x}_i(k-1)$'s prior to the iteration through

$$\sum_{i\in\mathcal{V}} f_i'(\hat{x}_i(k)) = \sum_{i\in\mathcal{V}} f_i'(\hat{x}_i(k-1)), \quad \forall k \in \mathbb{P}. \tag{5.10}$$

To satisfy (5.9), recall from Section 5.2 that every node $i \in \mathcal{V}$ knows the function f_i and knows that f_i has a unique minimizer $x_i^* \in \mathcal{X}$, which yields $f_i'(x_i^*) = 0$. Thus, (5.9) can be met by having every node $i \in \mathcal{V}$ compute x_i^* on its own and then initialize $\hat{x}_i(0)$ to x_i^*, i.e.,

$$\hat{x}_i(0) = x_i^*, \quad \forall i \in \mathcal{V}. \tag{5.11}$$

On the other hand, to satisfy (5.10), consider a gossip algorithm, whereby at each iteration $k \in \mathbb{P}$, a pair $u(k) = \{u_1(k), u_2(k)\} \in \mathcal{E}(k)$ of one-hop neighbors $u_1(k)$ and $u_2(k)$ communicate with each other and update their $\hat{x}_{u_1(k)}(k)$ and $\hat{x}_{u_2(k)}(k)$, while the rest of the N nodes stay idle and experience no change in their $\hat{x}_i(k)$'s, i.e.,

$$\hat{x}_i(k) = \hat{x}_i(k-1), \quad \forall k \in \mathbb{P},\ \forall i \in \mathcal{V} - u(k). \tag{5.12}$$

Notice that with (5.12), equation (5.10) simplifies to

$$\begin{aligned} &f'_{u_1(k)}(\hat{x}_{u_1(k)}(k)) + f'_{u_2(k)}(\hat{x}_{u_2(k)}(k)) \\ &\quad = f'_{u_1(k)}(\hat{x}_{u_1(k)}(k-1)) + f'_{u_2(k)}(\hat{x}_{u_2(k)}(k-1)), \quad \forall k \in \mathbb{P}. \end{aligned} \tag{5.13}$$

Also note that the entire expression (5.13) is known to nodes $u_1(k)$ and $u_2(k)$: $f'_{u_1(k)}$ and $f'_{u_2(k)}$ are derivatives of $f_{u_1(k)}$ and $f_{u_2(k)}$ they observe, $\hat{x}_{u_1(k)}(k-1)$ and

$\hat{x}_{u_2(k)}(k-1)$ are "old" values of the state variables they maintain, and $\hat{x}_{u_1(k)}(k)$ and $\hat{x}_{u_2(k)}(k)$ are "new" values they seek to jointly determine, respectively. Hence, all that is needed for (5.10) to hold is a gossip between nodes $u_1(k)$ and $u_2(k)$ to share their $f_{u_1(k)}$, $f_{u_2(k)}$, $\hat{x}_{u_1(k)}(k-1)$, and $\hat{x}_{u_2(k)}(k-1)$, followed by a joint update of their $\hat{x}_{u_1(k)}(k)$ and $\hat{x}_{u_2(k)}(k)$, which ensures (5.13).

Obviously, (5.13) alone does not uniquely determine $\hat{x}_{u_1(k)}(k)$ and $\hat{x}_{u_2(k)}(k)$, since there are two variables but only one equation. This suggests that the available degree of freedom may be used to account for the dissipation condition (5.5), which has yet to be addressed. Unlike the conservation condition (5.4), however, the dissipation condition (5.5) is not about how the state variables $\hat{x}_i(k)$'s should evolve for every finite k. Instead, it is about where the $\hat{x}_i(k)$'s should approach as k goes to infinity, which nodes $u_1(k)$ and $u_2(k)$ cannot guarantee themselves since they are only responsible for two of the N $\hat{x}_i(k)$'s. Nevertheless, given that all the N $\hat{x}_i(k)$'s should approach the *same* limit, nodes $u_1(k)$ and $u_2(k)$ can help make this happen by imposing an *equalizing condition*, forcing $\hat{x}_{u_1(k)}(k)$ and $\hat{x}_{u_2(k)}(k)$ to be equal, i.e.,

$$\hat{x}_{u_1(k)}(k) = \hat{x}_{u_2(k)}(k), \quad \forall k \in \mathbb{P}. \tag{5.14}$$

With the equalizing condition (5.14) added, there are now two equations with two variables, providing nodes $u_1(k)$ and $u_2(k)$ a chance to uniquely determine $\hat{x}_{u_1(k)}(k)$ and $\hat{x}_{u_2(k)}(k)$ from (5.13) and (5.14).

The following proposition asserts that (5.13) and (5.14) always have a unique solution, so that the evolution of the $\hat{x}_i(k)$'s is well-defined:

Proposition 5.2. *With Assumption 5.1 and* (5.11)–(5.14)*,* $\hat{x}_i(k)$ $\forall k \in \mathbb{N}$ $\forall i \in$

$\mathcal{V}$ are well-defined, i.e., unambiguous and in $\mathcal{X}$. Moreover,

$$[\min_{i\in\mathcal{V}} \hat{x}_i(k), \max_{i\in\mathcal{V}} \hat{x}_i(k)] \subset [\min_{i\in\mathcal{V}} \hat{x}_i(k-1), \max_{i\in\mathcal{V}} \hat{x}_i(k-1)], \quad \forall k \in \mathbb{P}.$$

Proof. By induction on $k \in \mathbb{N}$. By Assumption 5.1 and (5.11), $\hat{x}_i(0)$ $\forall i \in \mathcal{V}$ are unambiguous and in $\mathcal{X}$. Next, let $k \in \mathbb{P}$ and suppose $\hat{x}_i(k-1)$ $\forall i \in \mathcal{V}$ are unambiguous and in $\mathcal{X}$. We show that so are $\hat{x}_i(k)$ $\forall i \in \mathcal{V}$. From (5.12), $\hat{x}_i(k)$ $\forall i \in \mathcal{V} - u(k)$ are unambiguous and in $\mathcal{X}$. To show that so are $\hat{x}_{u_1(k)}(k)$ and $\hat{x}_{u_2(k)}(k)$, we show that (5.13) and (5.14) have a unique solution $(\hat{x}_{u_1(k)}(k), \hat{x}_{u_2(k)}(k)) \in \mathcal{X}^2$. By Lemma 5.1, there is a unique $z \in \mathcal{X}$ such that

$$f'_{u_1(k)}(z) + f'_{u_2(k)}(z) = f'_{u_1(k)}(\hat{x}_{u_1(k)}(k-1)) + f'_{u_2(k)}(\hat{x}_{u_2(k)}(k-1)), \qquad (5.15)$$

which satisfies $z \in [\min_{i\in u(k)} \hat{x}_i(k-1), \max_{i\in u(k)} \hat{x}_i(k-1)]$. Setting $\hat{x}_{u_1(k)}(k) = \hat{x}_{u_2(k)}(k) = z$, we see that $(\hat{x}_{u_1(k)}(k), \hat{x}_{u_2(k)}(k))$ is a solution to (5.13) and (5.14), confirming the existence. Now let $(a_1, a_2) \in \mathcal{X}^2$ and $(b_1, b_2) \in \mathcal{X}^2$ be two solutions of (5.13) and (5.14). Then, due to (5.14), (5.13), and Lemma 5.1, we have $a_1 = a_2 = b_1 = b_2$, confirming the uniqueness. Therefore, $\hat{x}_i(k)$ $\forall i \in \mathcal{V}$ are well-defined as desired. Finally, the second statement follows from (5.12) and the fact that $\hat{x}_{u_1(k)}(k) = \hat{x}_{u_2(k)}(k) \in [\min_{i\in u(k)} \hat{x}_i(k-1), \max_{i\in u(k)} \hat{x}_i(k-1)]$ $\forall k \in \mathbb{P}$. □

Proposition 5.2 calls for a few remarks. First, the interval $[\min_{i\in\mathcal{V}} \hat{x}_i(k), \max_{i\in\mathcal{V}} \hat{x}_i(k)]$ can only shrink or remain unchanged over time k. While this does not guarantee the dissipation condition (5.5), it shows that the $\hat{x}_i(k)$'s are "trying" to converge and are, at the very least, bounded even if $\mathcal{X}$ is not. Second, the proofs of Proposition 5.2 and Lemma 5.1 suggest a simple, practical procedure for nodes $u_1(k)$ and $u_2(k)$ to solve (5.13) and (5.14) for $(\hat{x}_{u_1(k)}(k), \hat{x}_{u_2(k)}(k))$: apply a numerical *root-finding method*, such as the *bisection method* with initial bracket $[\min_{i\in u(k)} \hat{x}_i(k-1), \max_{i\in u(k)} \hat{x}_i(k-1)]$, to solve (5.15) for the unique z and then set $\hat{x}_{u_1(k)}(k) = \hat{x}_{u_2(k)}(k) = z$. Finally, since (5.15) always has a unique solution z, we can eliminate z and write

$$\begin{aligned}\hat{x}_{u_1(k)}(k) &= \hat{x}_{u_2(k)}(k)\\ &= (f'_{u_1(k)} + f'_{u_2(k)})^{-1}(f'_{u_1(k)}(\hat{x}_{u_1(k)}(k-1)) + f'_{u_2(k)}(\hat{x}_{u_2(k)}(k-1))), \quad \forall k \in \mathbb{P},\end{aligned} \tag{5.16}$$

where $(f'_i+f'_j)^{-1} : (f'_i+f'_j)(\mathcal{X}) \to \mathcal{X}$ denotes the inverse of the injective function $f'_i + f'_j$ with its codomain restricted to its range.

Expressions (5.11), (5.12), and (5.16) collectively define a gossip-style, distributed asynchronous iterative algorithm that yields a switched, nonlinear, networked dynamical system

$$\hat{x}_i(k) = \begin{cases} (\sum_{j\in u(k)} f'_j)^{-1}(\sum_{j\in u(k)} f'_j(\hat{x}_j(k-1))), & \text{if } i \in u(k),\\ \hat{x}_i(k-1), & \text{otherwise,}\end{cases} \quad \forall k \in \mathbb{P},\ \forall i \in \mathcal{V}, \tag{5.17}$$

with initial condition (5.11), and with $(u(k))_{k=1}^{\infty}$ representing the sequence of gossiping nodes that trigger the switchings. As this algorithm ensures the conservation condition (5.4), the state trajectory $(\hat{x}_1(k), \hat{x}_2(k), \ldots, \hat{x}_N(k))$ must remain on an $(N-1)$-dimensional manifold $\mathcal{M} = \{(x_1, x_2, \ldots, x_N) \in \mathcal{X}^N : \sum_{i\in\mathcal{V}} f'_i(x_i) = 0\} \subset \mathcal{X}^N \subset \mathbb{R}^N$, making $\mathcal{M}$ an invariant set. Given that the algorithm involves repeated, pairwise equalizing of the $\hat{x}_i(k)$'s, we refer to it as *Pairwise Equalizing* (PE). PE may be expressed in a compact algorithmic form as follows:

Algorithm 5.1 (Pairwise Equalizing)**.**

Initialization:

1. Each node $i \in \mathcal{V}$ computes $x_i^* \in \mathcal{X}$.

2. Each node $i \in \mathcal{V}$ creates a variable $\hat{x}_i \in \mathcal{X}$ and initializes it:

   $\hat{x}_i \leftarrow x_i^*$.

Operation: At each iteration:

3. A node with one or more one-hop neighbors, say, node i, initiates the iteration and selects a one-hop neighbor, say, node j, to gossip.
4. Nodes i and j select one of two ways to gossip by labeling themselves as either nodes a and b, or nodes b and a, respectively, where $\{a, b\} = \{i, j\}$.
5. If node b does not know f_a, then node a transmits f_a to node b.
6. Node a transmits $\hat{x}_a$ to node b.
7. Node b updates $\hat{x}_b$:

   $\hat{x}_b \leftarrow (f_a' + f_b')^{-1}(f_a'(\hat{x}_a) + f_b'(\hat{x}_b))$.
8. Node b transmits $\hat{x}_b$ to node a.
9. Node a updates $\hat{x}_a$:

   $\hat{x}_a \leftarrow \hat{x}_b$. ■

Algorithm 5.1, or PE, consists of an *initialization* part that is executed once, and an *operation* part that is executed iteratively. Several remarks concerning their execution are as follows: Step 1 may be accomplished by letting every node $i \in \mathcal{V}$ calculate the root x_i^* of $f_i'(x_i^*) = 0$ analytically whenever possible (e.g., when $f_i(x) = x^2 + 2x + 3$), and numerically via a root-finding method whenever not (e.g., when $f_i(x) = x^2+2e^{-x}+3e^x$). In the latter case, as was alluded to earlier, a suitable choice is the bisection method, which can also be used to carry out Step 7. Step 2 is intended to create the node estimates, or state variables, and initialize them using the result of Step 1. Step 3 may be realized either deterministically (e.g., each node periodically initiates an iteration and cyclically picks a neighbor) or stochastically (e.g., each node initiates an iteration according to some Poisson process and equiprobably picks a neighbor), depending on which is more appropriate for the particular application.

Step 4 is intended to let nodes i and j pick one of two ways to gossip that are equivalent mathematically, but different communicatively and computationally: notice from Steps 5–9 that the node that labels itself as node a has little to compute but has to communicate the function f_a once in Step 5, which consumes bandwidth and transmission power. In contrast, the node that labels itself as node b has not much to communicate but has to compute the update in Step 7, which demands processor time and effort. Thus, Step 4 offers nodes i and j an opportunity to take advantage of the asymmetry in their actions, to better utilize their communication and computational resources. This feature may be useful, especially in a resource-constrained network. For instance, if f_i requires fewer data symbols to represent—and, hence, less bandwidth and power to transmit—than f_j, or if node i's processor is slower or busier than node j's, then nodes i and j might want to label themselves as nodes a and b, as opposed to nodes b and a, respectively.

Steps 5 and 6 are introduced so that node b can perform Step 7, whereas Step 8 is introduced so that node a can perform Step 9. Note that Step 5 is a conditional step that is carried out if and only if the condition "node b does not know f_a" is true. For a wired network, this condition is true if and only if nodes i and j are gossiping or alternating their a-b labels for the first time, since the function f_a, upon reception by node b, could be stored in its local memory for later use. However, for a wireless network, this condition may be false even if nodes i and j are gossiping or alternating their a-b labels for the first time, since node b may have quietly learned about f_a by overhearing the wireless transmission of f_a from node a to another neighbor during a previous iteration. Observe that whenever the condition is false (which it almost always is), only *two* real-number transmissions are needed per iteration, in Steps 6 and 8.

Notice that PE does not rely on a stepsize parameter to execute, nor does it require the construction of a (pseudo-)Hamiltonian cycle, as well as the concurrent use of a routing protocol for multi-hop transmissions. Indeed, all it essentially needs is that every node is capable of applying a root-finding method, maintaining a list of its one-hop neighbors, and remembering the functions it learns along the way. Therefore, PE overcomes limitations L1–L3, while being rather easy to implement—although computationally it is more demanding than the subgradient algorithms.

To show that PE asymptotically converges and, thus, circumvents L4, let $\mathbf{x}^* = (x^*, x^*, \ldots, x^*)$ and $\mathbf{x}(k) = (\hat{x}_1(k), \hat{x}_2(k), \ldots, \hat{x}_N(k))$. Then, from Propositions 5.1 and 5.2, $\mathbf{x}^* \in \mathcal{X}^N$ and $\mathbf{x}(k) \in \mathcal{X}^N$ $\forall k \in \mathbb{N}$. In addition, due to (5.17), if $\mathbf{x}(k) = \mathbf{x}^*$ for some $k \in \mathbb{N}$, then $\mathbf{x}(\ell) = \mathbf{x}^*$ $\forall \ell > k$. Hence, $\mathbf{x}^*$ is an equilibrium point of the system (5.17). To show that $\lim_{k\to\infty} \mathbf{x}(k) = \mathbf{x}^*$, i.e., (5.3) holds, we seek to construct a Lyapunov function. To this end, recall that for any strictly convex and differentiable function $f : \mathcal{X} \to \mathbb{R}$, the first-order convexity condition says that

$$f(y) \geq f(x) + f'(x)(y - x), \quad \forall x, y \in \mathcal{X}, \tag{5.18}$$

where the equality holds if and only if $x = y$. This suggests the following Lyapunov function candidate $V : \mathcal{X}^N \subset \mathbb{R}^N \to \mathbb{R}$, which exploits the convexity of the f_i's:

$$V(\mathbf{x}(k)) = \sum_{i \in \mathcal{V}} f_i(x^*) - f_i(\hat{x}_i(k)) - f_i'(\hat{x}_i(k))(x^* - \hat{x}_i(k)). \tag{5.19}$$

Notice that V in (5.19) is well-defined. Moreover, due to Assumption 5.1 and (5.18), V is continuous and positive definite with respect to $\mathbf{x}^*$, i.e., $V(\mathbf{x}(k)) \geq 0$ $\forall \mathbf{x}(k) \in \mathcal{X}^N$, where the equality holds if and only if $\mathbf{x}(k) = \mathbf{x}^*$. Therefore, to prove (5.3), it suffices to show that

$$\lim_{k\to\infty} V(\mathbf{x}(k)) = 0. \tag{5.20}$$

The following lemma represents the first step toward establishing (5.20):

Lemma 5.2. *Consider the use of PE described in Algorithm 5.1. Suppose Assumption 5.1 holds. Then, for any given* $(u(k))_{k=1}^\infty$, $(V(\mathbf{x}(k)))_{k=0}^\infty$ *is non-increasing and satisfies*

$$\begin{aligned} &V(\mathbf{x}(k)) - V(\mathbf{x}(k-1)) \\ &\quad = -\sum_{i\in u(k)} f_i(\hat{x}_i(k)) - f_i(\hat{x}_i(k-1)) - f_i'(\hat{x}_i(k-1))(\hat{x}_i(k) - \hat{x}_i(k-1)), \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \forall k \in \mathbb{P}. \end{aligned} \tag{5.21}$$

Proof. Let $(u(k))_{k=1}^\infty$ be given. Then, from (5.19) and (5.17), we have $V(\mathbf{x}(k)) - V(\mathbf{x}(k-1)) = -\sum_{i\in u(k)} f_i(\hat{x}_i(k)) - f_i(\hat{x}_i(k-1)) + f_i'(\hat{x}_i(k))x^* - f_i'(\hat{x}_i(k-1))x^* - f_i'(\hat{x}_i(k))\hat{x}_i(k) + f_i'(\hat{x}_i(k-1))\hat{x}_i(k-1)$ $\forall k \in \mathbb{P}$. Due to (5.17), $-\sum_{i\in u(k)} f_i'(\hat{x}_i(k))x^*$ cancels $\sum_{i\in u(k)} f_i'(\hat{x}_i(k-1))x^*$, while $\sum_{i\in u(k)} f_i'(\hat{x}_i(k))\hat{x}_i(k)$ becomes $\sum_{i\in u(k)} f_i'(\hat{x}_i(k-1))\hat{x}_i(k)$. This proves (5.21). Note that the right-hand side of (5.21) is nonpositive due to (5.18). Hence, $(V(\mathbf{x}(k)))_{k=0}^\infty$ is non-increasing. □

Lemma 5.2 has several implications. First, upon completing each iteration $k \in \mathbb{P}$ by *any* two nodes $u_1(k)$ and $u_2(k)$, the value of V must either decrease or, at worst, stay the same, where the latter occurs if and only if $\hat{x}_{u_1(k)}(k-1) = \hat{x}_{u_2(k)}(k-1)$. Second, since $(V(\mathbf{x}(k)))_{k=0}^\infty$ is non-increasing irrespective of $(u(k))_{k=1}^\infty$, V in (5.19) may be regarded as a *common* Lyapunov function for the nonlinear switched system (5.17), which has as many as $\frac{N(N-1)}{2}$ different dynamics, corresponding to the $\frac{N(N-1)}{2}$ possible gossiping pairs. Finally, the first-order convexity condition (5.18) can be used not only to form the common Lyapunov function V, but also to characterize drops in its value in (5.21) after every gossip. This is akin to how quadratic functions may be used to form a common Lyapunov function $V(k) = x^T(k)Px(k)$ for a linear switched system $x(k+1) = A(k)x(k)$, $A(k) \in \{A_1, A_2, \ldots, A_M\}$, as well as to characterize drops in $V(k)$ via $V(k+1) - V(k) = x^T(k)(A_i^T PA_i - P)x(k) = -x^T(k)Q_i x(k)$. Indeed, as we will show later, when problem (5.2) specializes to an averaging problem, where the nonlinear switched system (5.17) becomes linear, both V and its drop become quadratic functions.

As $(V(\mathbf{x}(k)))_{k=0}^{\infty}$ is nonnegative and non-increasing, $\lim_{k\to\infty} V(\mathbf{x}(k))$ exists and is nonnegative. This, however, is insufficient for us to conclude that $\lim_{k\to\infty} V(\mathbf{x}(k)) = 0$, since, for some pathological gossiping patterns, $\lim_{k\to\infty} V(\mathbf{x}(k))$ can be positive. To see this, suppose the set $\mathcal{V}$ of nodes can be partitioned into two nonempty subsets, such that the nodes in one subset never gossip with those in the other—either by force (e.g., $\mathcal{V} = \{1, 2, 3, 4\}$ and $\mathcal{E}(k) \equiv \{\{1, 2\}, \{3, 4\}\}$, so that $u(k)$ is forced to be $\{1, 2\}$ or $\{3, 4\}$) or by choice (e.g., $\mathcal{V} = \{1, 2, 3, 4\}$ and $\mathcal{E}(k) \equiv \{\{1, 2\}, \{2, 3\}, \{3, 4\}\}$, but $u(k)$ is chosen to be $\{1, 2\}$ or $\{3, 4\}$). Then, $V(\mathbf{x}(k))$ in general would be bounded away from zero by a positive constant, since x^* depends on all the f_i's, but information never flows between the subsets. Thus, some restrictions must be imposed on the gossiping pattern, in order to establish (5.20).

Given that PE—or, specifically, its Step 3—may be realized either *deterministically* or *stochastically*, we will introduce restrictions on the gossiping pattern in both of these frameworks. Moreover, since the sequence $(\mathcal{E}(k))_{k=0}^{\infty}$

was assumed in Section 5.2 to be exogenous, below we will treat $(\mathcal{E}(k))_{k=0}^{\infty}$ simply as given, regardless of the frameworks.

In the *deterministic* framework, suppose each node initiates an iteration and picks a neighbor to gossip according to some deterministic policy, resulting in a deterministic sequence $(u(k))_{k=1}^{\infty}$, which must satisfy $u(k) \in \mathcal{E}(k)\ \forall k \in \mathbb{P}$. For any given $(u(k))_{k=1}^{\infty}$, define the set $\mathcal{E}_{\infty} \subset \{\{i,j\} : i,j \in \mathcal{V}, i \neq j\}$ as

$$\mathcal{E}_{\infty} = \{\{i,j\} : u(k) = \{i,j\} \text{ for infinitely many } k \in \mathbb{P}\}. \tag{5.22}$$

Equation (5.22) says that a link $\{i,j\}$ is in $\mathcal{E}_{\infty}$ if and only if nodes i and j gossip with each other infinitely often. With $\mathcal{E}_{\infty}$ defined as such, we may state the following restriction on the gossiping pattern, which was first adopted in [72]:

Assumption 5.2. The sequence $(u(k))_{k=1}^{\infty}$ is such that the graph $(\mathcal{V}, \mathcal{E}_{\infty})$ is connected.

Assumption 5.2 is not difficult to satisfy in practice, provided that the network is "connected in the long run." To justify this claim, consider the exogenous sequence $(\mathcal{E}(k))_{k=0}^{\infty}$ and let $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_M$ represent the sets of links that occur infinitely often in $(\mathcal{E}(k))_{k=0}^{\infty}$, i.e., for each $\ell \in \{1, 2, \ldots, M\}$, $\mathcal{E}(k) = \mathcal{E}_{\ell}$ for infinitely many k's. Note that if the graph $(\mathcal{V}, \cup_{\ell=1}^{M} \mathcal{E}_{\ell})$ is not connected, it means that the set $\mathcal{V}$ of nodes can be partitioned into two nonempty subsets $\mathcal{V}_1$ and $\mathcal{V}_2$, such that after some finite time, the nodes in $\mathcal{V}_1$ can no longer gossip with those in $\mathcal{V}_2$, even if they want to. Thus, we may say that the network is *connected in the long run* if and only if the graph $(\mathcal{V}, \cup_{\ell=1}^{M} \mathcal{E}_{\ell})$ is connected. Now suppose the graph $(\mathcal{V}, \cup_{\ell=1}^{M} \mathcal{E}_{\ell})$ is connected. Also suppose $\mathcal{E}(k)$ is slowly varying, in the sense that it is constant for many consecutive k's. This assumption is reasonable because the topology of a network typically changes at a rate that is much slower compared to the rate at which iterations can occur (e.g., in a wireless network, although path losses and shadowing may cause a link to fail or recover, such a change occurs at a much slower time scale compared to the propagation of electromagnetic waves). Since the graph $(\mathcal{V}, \cup_{\ell=1}^{M} \mathcal{E}_\ell)$ is connected and $\mathcal{E}(k)$ is slowly varying, if we simply let every possible pair of one-hop neighbors gossip frequently enough—at least once per change in $\mathcal{E}(k)$—then $\mathcal{E}_\infty = \cup_{\ell=1}^{M} \mathcal{E}_\ell$, so that Assumption 5.2 holds. Therefore, as long as the network is connected in the long run, Assumption 5.2 can be easily met.

Notice that in the previous paragraph, if the graph $(\mathcal{V}, \cup_{\ell=1}^{M} \mathcal{E}_\ell)$ is not connected, then for every $i \in \mathcal{V}_1$ and $j \in \mathcal{V}_2$, we have $\{i, j\} \notin \mathcal{E}_\infty$. This implies that the graph $(\mathcal{V}, \mathcal{E}_\infty)$ is also not connected, so that Assumption 5.2 fails. In this case, PE generally would fail to asymptotically converge, but so would most distributed iterative algorithms, including the consensus algorithms in [5,19,21, 24,25,50,52,63,73], as well as the averaging algorithms in [8,13,26,36,39,72,74].

Based on Assumption 5.2, the following theorem can be established:

Theorem 5.1. *Consider the use of PE described in Algorithm 5.1. Suppose Assumptions 5.1 and 5.2 hold. Then,* (5.20) *and* (5.3) *hold.*

Proof. See Appendix D.1. □

Theorem 5.1 says that, under Assumption 5.2 on the gossiping pattern, PE ensures asymptotic convergence of all the $\hat{x}_i(k)$'s to x^*, circumventing limitation L4 facing many of the existing subgradient algorithms.

Next, in the *stochastic* framework, suppose each node initiates an iteration and picks a neighbor to gossip according to some random strategy, resulting in a random sequence $(u(k))_{k=1}^{\infty}$, which satisfies $u(k) \in \mathcal{E}(k)$ $\forall k \in \mathbb{P}$, and which is independent, but not necessarily identically distributed, over time k. For each $k \in \mathbb{P}$ and each $\{i,j\} \in \mathcal{E}(k)$, let $p_{\{i,j\}}(k) \in [0,1]$ denote the probability of $u(k)$ being $\{i,j\}$. In addition, for each $\{i,j\} \notin \mathcal{E}(k)$, let $p_{\{i,j\}}(k)$ be undefined since the event $u(k) = \{i,j\}$ cannot happen. For any given $p_{\{i,j\}}(k)$ $\forall k \in \mathbb{P}$ $\forall \{i,j\} \in \mathcal{E}(k)$, define the set $\tilde{\mathcal{E}}_\infty \subset \{\{i,j\} : i,j \in \mathcal{V}, i \neq j\}$ as

$$\tilde{\mathcal{E}}_\infty = \{\{i,j\} : \exists \varepsilon > 0 \text{ such that } \forall k \in \mathbb{P},\ p_{\{i,j\}}(\ell) \geq \varepsilon \text{ for some } \ell > k\}. \quad (5.23)$$

Expression (5.23) says that a link $\{i,j\}$ is in $\tilde{\mathcal{E}}_\infty$ if and only if the probability with which nodes i and j gossip with each other is no less than a positive constant ε for infinitely many iterations. In other words, $\{i,j\} \in \tilde{\mathcal{E}}_\infty$ if and only if the sequence $(p_{\{i,j\}}(k))_{k=1}^{\infty}$ has a subsequence whose elements are no less than ε. For instance, if $p_{\{i,j\}}(k) = \frac{1}{k}$ $\forall k \in \mathbb{P}$, then $\{i,j\} \notin \tilde{\mathcal{E}}_\infty$. In contrast, if

$$(p_{\{i,j\}}(k))_{k=1}^{\infty} = (0.1, \underbrace{\#, \ldots, \#}_{\text{10 times}}, 0.1, \underbrace{\#, \ldots, \#}_{\text{100 times}}, 0.1, \underbrace{\#, \ldots, \#}_{\text{1000 times}}, \ldots),$$

where # represents either zero or "undefined," then $\{i,j\} \in \tilde{\mathcal{E}}_\infty$. Based on this definition of $\tilde{\mathcal{E}}_\infty$, we may introduce the following restriction on the random gossiping pattern:

Assumption 5.3. The random sequence $(u(k))_{k=1}^{\infty}$ is such that the graph $(\mathcal{V}, \tilde{\mathcal{E}}_\infty)$ is connected.

Similar to Assumption 5.2, it is not difficult to satisfy Assumption 5.3, so long that the network is connected in the long run. To explain this, suppose the graph $(\mathcal{V}, \cup_{\ell=1}^{M} \mathcal{E}_\ell)$ is connected. Note that at each time $k \in \mathbb{P}$ and for each node $i \in \mathcal{V}$ who has one or more one-hop neighbors at time k, if we simply let the probabilities P{node i initiates iteration k} be no less than some $\varepsilon_1 >$

0 and P{node i picks node j to gossip | node i initiates iteration k} be no less than some $\varepsilon_2 > 0$, then $p_{\{i,j\}}(k) \geq 2\varepsilon_1\varepsilon_2 \ \forall k \in \mathbb{P} \ \forall\{i,j\} \in \mathcal{E}(k)$. This implies that $\tilde{\mathcal{E}}_\infty = \cup_{\ell=1}^{M}\mathcal{E}_\ell$, so that Assumption 5.3 is met, explaining the argument.

With Assumption 5.3, the following stochastic version of Theorem 5.1 can be stated:

Theorem 5.2. *Consider the network modeled in Section 5.2 and the use of PE described in Algorithm 5.1. Suppose Assumptions 5.1 and 5.3 hold. Then, with probability* 1, (5.20) *and* (5.3) *hold.*

Proof. See Appendix D.2. □

Theorem 5.2 shows that, under Assumption 5.3 on the random gossiping pattern, PE is almost surely asymptotically convergent, again overcoming limitation L4.

Finally, we point out that the above results may be viewed as a natural generalization of some known results in distributed averaging. Consider a special case where each node $i \in \mathcal{V}$ observes not an arbitrary function f_i, but a quadratic one of the form $f_i(x) = \frac{1}{2}(x - y_i)^2 + c_i$ with domain $\mathcal{X} = \mathbb{R}$ and parameters $y_i, c_i \in \mathbb{R}$. In this case, finding the unknown optimizer x^* amounts to calculating the network-wide average $\frac{1}{N}\sum_{i\in\mathcal{V}} y_i$ of the node "observations" y_i's, so that the convex optimization problem (5.2) becomes an averaging problem. In addition, initializing the node estimates $\hat{x}_i(0)$'s simply means setting them to the y_i's, and equalizing $\hat{x}_{u_1(k)}(k)$ and $\hat{x}_{u_2(k)}(k)$ simply means averaging them, so that PE reduces to Pairwise Averaging [72] and Randomized Gossip Algorithm [8]. Moreover, the invariant manifold $\mathcal{M}$ becomes the invariant hyperplane $\mathcal{M} = \{(x_1, x_2, \ldots, x_N) \in \mathbb{R}^N : \sum_{i\in\mathcal{V}} x_i = \sum_{i\in\mathcal{V}} y_i\}$ in distributed averaging. Furthermore, both the common Lyapunov function V in (5.19) and its drop in (5.21) take a quadratic form: $V(\mathbf{x}(k)) = \frac{1}{2}(\mathbf{x}(k)-\mathbf{x}^*)^T(\mathbf{x}(k)-\mathbf{x}^*)$ and $V(\mathbf{x}(k))-V(\mathbf{x}(k-1)) = -\frac{1}{2}\mathbf{x}^T(k-1)Q_{u(k)}\mathbf{x}(k-1)\ \forall k \in \mathbb{P}$, where $Q_{\{i,j\}} \in \mathbb{R}^{N\times N}$ is a symmetric positive semidefinite matrix whose ii and jj entries are $\frac{1}{2}$, ij and ji entries are $-\frac{1}{2}$, and all other entries are zero. Therefore, the first-order-convexity-condition-based Lyapunov function (5.19) generalizes the quadratic Lyapunov function in distributed averaging.

5.4 Illustrative Example of Pairwise Equalizing

In this section, we illustrate the effectiveness of PE via a simple example.

Consider a network of 20 nodes, connected by 30 links in a fixed topology, as shown in Figure 5.1. Suppose each node i observes a function $f_i : \mathbb{R} \to \mathbb{R}$, given by

$$f_i(x) = a_i x + b_i(x - c_i)^2 + d_i(x - e_i)^4, \tag{5.24}$$

where a_i, b_i, c_i, d_i, e_i are parameters of f_i, whose values are randomly chosen from the intervals $(-1, 1), (0, 1), (-1, 1), (0, 2), (-1, 1)$ and tabulated in Table 5.1. The f_i's in (5.24) fulfill Assumption 5.1 because the b_i's and d_i's are positive. To visualize these f_i's, their graphs are displayed as thumbnails in Figure 5.1 and superimposed on the same plot in Figure 5.2. Also depicted in Figure 5.2 are the graph of the function F, scaled by $\frac{1}{N}$ so that it fits into the figure, and the unknown optimizer x^* of F, that all the nodes wish to determine.

Suppose the nodes apply PE and carry out its Step 3 stochastically, such that every pair of one-hop neighbors has equal probability (i.e., $\frac{1}{30}$) of

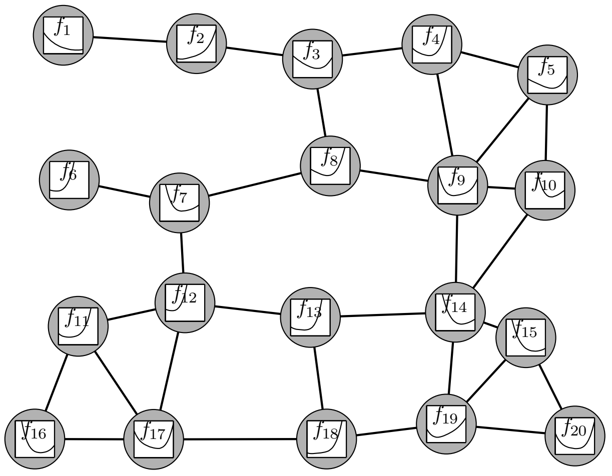


Figure 5.1: A 20-node, 30-link network with each node i observing a function f_i.

i	a_i	b_i	c_i	d_i	e_i
1	−0.2810	0.4370	0.3953	0.1205	0.3335
2	0.3413	0.2104	−0.7421	0.6309	−0.2726
3	0.1404	0.4386	0.9767	0.2041	−0.5822
4	−0.6774	0.6531	−0.4934	0.9326	−0.5111
5	−0.6821	0.1104	0.3127	0.2764	−0.6068
6	−0.2625	0.8210	−0.8058	1.6759	−0.8078
7	0.9529	0.4687	0.9535	1.2097	0.4785
8	−0.9216	0.2828	−0.7596	0.5923	−0.7625
9	−0.3640	0.4143	−0.8717	1.3849	0.1332
10	−0.4692	0.5232	−0.8121	1.1519	0.8586
11	−0.3629	0.6674	−0.7364	1.4327	−0.4212
12	−0.6336	0.5865	−0.9598	1.6579	−0.9906
13	0.3556	0.2700	0.4704	1.9244	−0.5025
14	0.1523	0.5920	0.1445	0.4462	0.9055
15	−0.1057	0.8464	0.3990	0.5949	0.6276
16	−0.2070	0.8811	0.1625	1.7635	0.3851
17	0.4505	0.5013	0.9122	1.2880	−0.1523
18	0.2128	0.0192	−0.3969	1.3203	−0.4198
19	0.2360	0.4288	−0.7291	0.5966	0.1399
20	0.1817	0.5743	0.3064	1.3042	−0.1372

Table 5.1: Parameters of the functions f_i's.

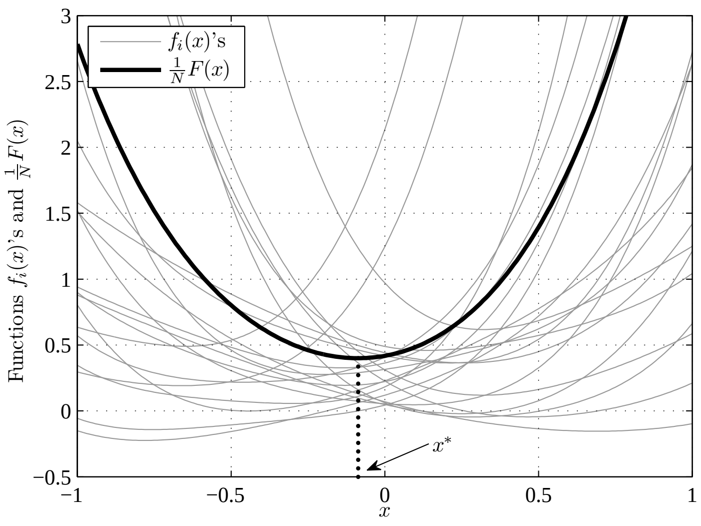


Figure 5.2: Graphs of the functions f_i's and $\frac{1}{N}F$, along with the unknown optimizer x^*.

being the pair $u(k)$ that gossips at iteration k, for every k. Figure 5.3 shows a realization of the random sequence $(u(k))_{k=1}^{1200}$ of gossiping pairs, obtained by simulating PE for 1200 iterations. Figure 5.4 shows, on a logarithmic scale, the value $V(\mathbf{x}(k))$ of the common Lyapunov function along the state trajectory $\mathbf{x}(k)$ of the system. Note that $V(\mathbf{x}(k))$ is indeed non-increasing, agreeing with Lemma 5.2. Moreover, it is converging to zero, at a rate that is roughly exponential. Figure 5.5 shows the individual components $\hat{x}_i(k)$'s of $\mathbf{x}(k)$, which represent the estimates of the unknown optimizer x^*. Observe that the $\hat{x}_i(k)$'s gradually approach x^*, converging to $x^* \pm 0.005$ after 1008 iterations. Furthermore, the closed interval $[\min_i \hat{x}_i(k), \max_i \hat{x}_i(k)]$ indeed can only shrink or remain unchanged, concurring with Proposition 5.2.

Notice that the network in Figure 5.1 contains no Hamiltonian cycle. Hence, it may be difficult to apply the subgradient algorithms mentioned in L2. Also, if the nodes are not fully aware of one another, or if they do not have a routing protocol, then the same can be said about the subgradient algorithms

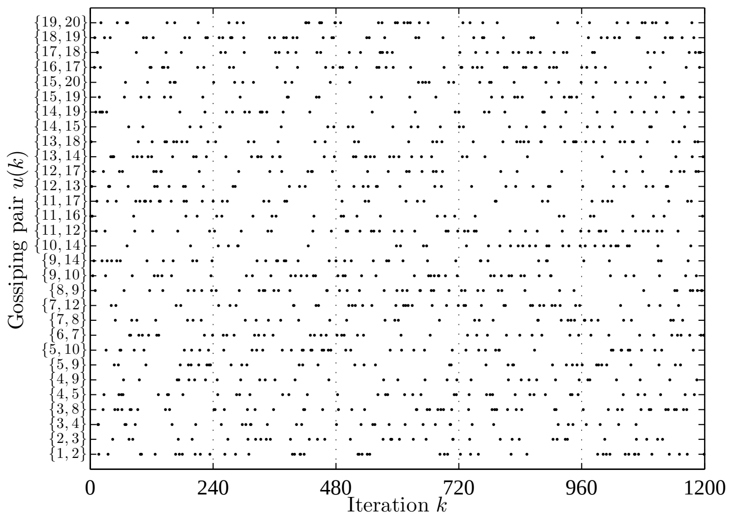


Figure 5.3: A realization of the random sequence $(u(k))_{k=1}^{1200}$ of gossiping pairs.

mentioned in L3, since it may be difficult to randomly and equiprobably pass the latest estimate of x^* among the nodes. In fact, even if such passing can be realized, each pass requires, on average, 2.98 real-number transmissions (or hops) to complete if shortest-path routing is used, and a higher number if it is not, or if the network diameter were larger. In comparison, although PE requires, in its Step 5, one-time transmissions of the f_i's as communication overhead, it requires only 2 real-number transmissions per iteration, regardless of the network size and topology.

5.5 Extended Pairwise Equalizing

In this section, we present an extension of PE to networks with not only time-varying topologies, but also time-varying node memberships, where nodes may freely join and leave, possibly due to node redeployment, failures/ recoveries, and battery depletion/recharge, which are quite common.

Consider a network whose topology and node memberships are time-

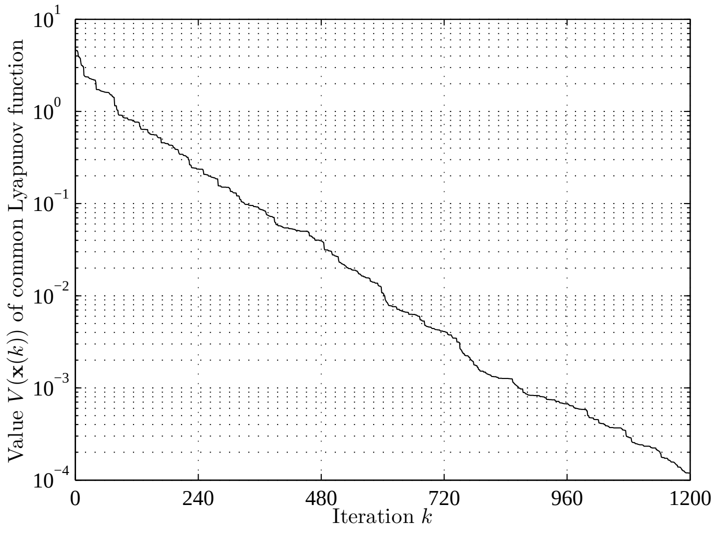


Figure 5.4: Convergence of the value $V(\mathbf{x}(k))$ of the common Lyapunov function to zero.

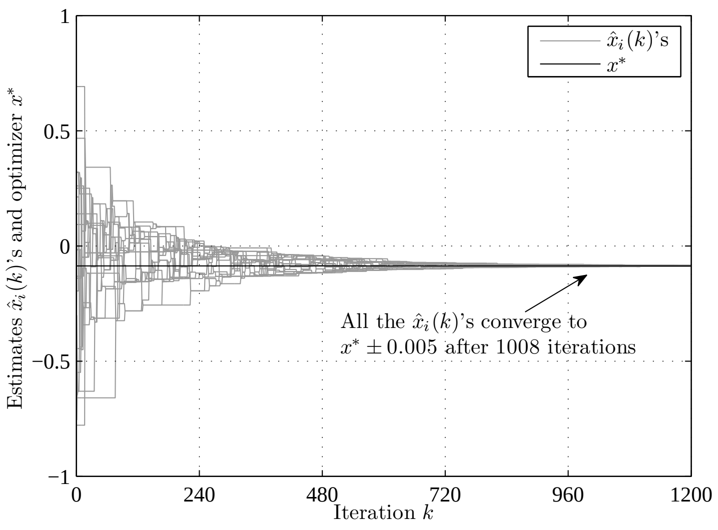


Figure 5.5: Convergence of the estimates $\hat{x}_i(k)$'s to the unknown optimizer x^*.

varying, where time is real-valued and nonnegative. Let $t \in [0, \infty)$ denote time, $\mathcal{V}$ denote the set of nodes in the network at time $t = 0$, and $\mathcal{F}$ denote the set of strictly convex, continuously differentiable functions mapping $\mathcal{X}$ to $\mathbb{R}$ and having minimizers in $\mathcal{X}$. Suppose, at time $t = 0$, each node $i \in \mathcal{V}$ observes a function $\tilde{f}_i \in \mathcal{F}$ with a minimizer $x_i^* \in \mathcal{X}$. Moreover, at any time $t \geq 0$, all the nodes in the network at that time wish to compute the unique minimizer $x^* \in \mathcal{X}$ of $\sum_{i \in \mathcal{V}} \tilde{f}_i$. To enable such computation, suppose at each time $t \in \mathbb{P}$, an iteration involving at least two nodes takes place. In addition, at each of the time instants $t_1, t_2, \ldots, t_p \in (0, \infty) - \mathbb{P}$, where $t_1 < t_2 < \cdots < t_p$ and $p \in \mathbb{N}$, one of the following two *events* occurs: *node joining*, whereby a node joins the network and has one or more one-hop neighbors upon joining, and *node leaving*, whereby a node leaves the network and has one or more one-hop neighbors before leaving. Notice that each node may freely join and leave the network, at essentially any time, for arbitrary but finite number of times. Also note that iterations and events cannot simultaneously occur, as the former occur at positive integer time instants, while the latter do not. Furthermore, if $p = 0$, i.e., there are no events, then the network becomes identical to the one modeled in Section 5.2.

To solve the problem formulated above, we have developed, again using the idea of conservation, an extension of PE, referred to as *Extended Pairwise Equalizing* (EPE) and described as follows:

Algorithm 5.2 (Extended Pairwise Equalizing)**.**

Initialization:

1. Same as the initialization part of Algorithm 5.1.
2. Each node $i \in \mathcal{V}$ creates a variable $f_i \in \mathcal{F}$ and initializes it:

$f_i \leftarrow \tilde{f}_i$.

Operation: At each iteration:

3. Same as the operation part of Algorithm 5.1.

Node joining:

4. A node, say, node i, joins the network and selects a one-hop neighbor, say, node j, to gossip.
5. Node j transmits $\hat{x}_j$ and f_j to node i.
6. Node j updates f_j:

   $f_j \leftarrow \frac{1}{2} f_j$.
7. Node i creates variables $\hat{x}_i \in \mathcal{X}$ and $f_i \in \mathcal{F}$ and initializes them sequentially:

   $\hat{x}_i \leftarrow \hat{x}_j$,

   $f_i \leftarrow \frac{1}{2} f_j$.

Node leaving:

8. A node, say, node i, wants to leave the network and selects a one-hop neighbor, say, node j, to gossip.
9. Node i transmits $\hat{x}_i$ and f_i to node j.
10. Node j updates $\hat{x}_j$ and f_j sequentially:

    $\hat{x}_j \leftarrow (f_i' + f_j')^{-1}(f_i'(\hat{x}_i) + f_j'(\hat{x}_j))$,

    $f_j \leftarrow f_i + f_j$.
11. Node i leaves the network. ■

Algorithm 5.2, or EPE, consists of four parts: *initialization*, *operation*, *node joining*, and *node leaving*. The first and second parts are identical to those of PE in Algorithm 5.1, with an exception: each node $i \in \mathcal{V}$, in addition to

$\hat{x}_i$, must create a variable f_i that is a function in $\mathcal{F}$ and initialize f_i to the observed $\tilde{f}_i$, in Step 2. The third and fourth parts are executed only when the *node joining* and *node leaving* events take place, respectively. Notice that the variables f_i's are constant throughout the iterative operation and are updated only if they are involved in an event, in Steps 6, 7, and 10. Moreover, these variables f_i's are always in $\mathcal{F}$, which can be seen by induction and from the fact that $\mathcal{F}$ is a convex cone. Furthermore, by induction and Lemma 5.1, the variables $\hat{x}_i$'s are always in $\mathcal{X}$. Finally, note that EPE allows a node to join the network with empty memory, and lose all its memory upon leaving.

Similar to PE, as long as the gossiping pattern is rich enough, EPE is deterministically and stochastically asymptotically convergent. To see this, we first introduce a few notations: for each $t \in [0, \infty)$, let $\mathcal{V}(t)$ represent the set of nodes in the network at time t, with $\mathcal{V}(0)$ denoted simply as $\mathcal{V}$. Moreover, for each $i \in \mathcal{V}(t)$, let $\hat{x}_i(t)$ and $f_{i,t}$ represent, respectively, the values of $\hat{x}_i$ and f_i maintained by node i at time t. Furthermore, let t^- and t^+ represent the times immediately before and after time t. Next, we show that EPE satisfies an *extended conservation condition*, defined as

$$\sum_{i \in \mathcal{V}(t)} f'_{i,t}(\hat{x}_i(t)) = 0, \quad \forall t \in [0, \infty), \tag{5.25}$$

and

$$\sum_{i \in \mathcal{V}(t)} f_{i,t} = \sum_{i \in \mathcal{V}} \tilde{f}_i, \quad \forall t \in [0, \infty), \tag{5.26}$$

where (5.25) parallels the conservation condition (5.4), while (5.26) says that the sum of the $f_{i,t}$'s is always conserved. According to Steps 1 and 2, at time $t = 0$, $f'_{i,0}(\hat{x}_i(0)) = 0$ and $f_{i,0} = \tilde{f}_i$ $\forall i \in \mathcal{V}$. According to Step 3, at each time $t \in \mathbb{P}$, $f'_{i,t^-}(\hat{x}_i(t^-)) + f'_{j,t^-}(\hat{x}_j(t^-)) = f'_{i,t^+}(\hat{x}_i(t^+)) + f'_{j,t^+}(\hat{x}_j(t^+))$ and

$f_{i,t^-} + f_{j,t^-} = f_{i,t^+} + f_{j,t^+}$. Due to Steps 4–7, at each time $t \in \{t_1, t_2, \ldots, t_p\}$, if the *node joining* event occurs, then $f'_{j,t^-}(\hat{x}_j(t^-)) = f'_{i,t^+}(\hat{x}_i(t^+)) + f'_{j,t^+}(\hat{x}_j(t^+))$ and $f_{j,t^-} = f_{i,t^+} + f_{j,t^+}$. Because of Steps 8–11, if the *node leaving* event occurs instead, then $f'_{i,t^-}(\hat{x}_i(t^-)) + f'_{j,t^-}(\hat{x}_j(t^-)) = f'_{j,t^+}(\hat{x}_j(t^+))$ and $f_{i,t^-} + f_{j,t^-} = f_{j,t^+}$. Combining the above, we get $\sum_{i\in\mathcal{V}} f'_{i,0}(\hat{x}_i(0)) = 0$, $\sum_{i\in\mathcal{V}} f_{i,0} = \sum_{i\in\mathcal{V}} \tilde{f}_i$, $\sum_{i\in\mathcal{V}(t^-)} f'_{i,t^-}(\hat{x}_i(t^-)) = \sum_{i\in\mathcal{V}(t^+)} f'_{i,t^+}(\hat{x}_i(t^+))$ $\forall t \in (0,\infty)$, and $\sum_{i\in\mathcal{V}(t^-)} f_{i,t^-} = \sum_{i\in\mathcal{V}(t^+)} f_{i,t^+}$ $\forall t \in (0,\infty)$. Therefore, with EPE, the extended conservation condition (5.25) and (5.26) hold. Finally, notice that after time t_p (which may be very large but finite), although the network topology can still vary, the node memberships can no longer change. Thus, after time t_p, EPE behaves just like PE, executing nothing but iterations at times $\lceil t_p \rceil$, $\lceil t_p \rceil + 1$, and so on. Because of this and because EPE ensures the extended conservation condition (5.25) and (5.26), by treating time $\lceil t_p \rceil$ as time 0, the convergence analysis of EPE may be carried out in the same way as that of PE reported in Section 5.3. Hence, if EPE is realized deterministically and satisfies Assumption 5.2 (with time $\lceil t_p \rceil$ treated as time 0), then it is asymptotically convergent, i.e.,

$$\lim_{t\to\infty} \hat{x}_i(t) = x^*, \quad \forall i \in \mathcal{V}(\lceil t_p \rceil). \tag{5.27}$$

Analogously, if EPE is implemented stochastically and satisfies Assumption 5.3, then (5.27) holds almost surely.

5.6 Pairwise Bisectioning

Although PE solves problem (5.2) and bypasses L1–L4, it requires one-time, one-way sharing of the f_i's between gossiping nodes, which may be costly for certain f_i's, or impermissible for security and privacy reasons. In this section, we develop another gossip algorithm that eliminates this requirement at the expense of more real-number transmissions per iteration.

Note that PE can be traced back to four defining equations (5.11)–(5.14), and that its drawback of having to share the f_i's stems from having to solve (5.13) and (5.14). To overcome this drawback, consider a gossip algorithm satisfying (5.11)–(5.13) and a new condition but not (5.14). Assuming, without loss of generality, that $\hat{x}_{u_1(k)}(k-1) \le \hat{x}_{u_2(k)}(k-1)\ \forall k \in \mathbb{P}$, this new condition can be stated as

$$\hat{x}_{u_1(k)}(k-1) \le \hat{x}_{u_1(k)}(k) \le \hat{x}_{u_2(k)}(k) \le \hat{x}_{u_2(k)}(k-1), \quad \forall k \in \mathbb{P}. \tag{5.28}$$

Termed as the *approaching condition*, (5.28) says that at each iteration $k \in \mathbb{P}$, nodes $u_1(k)$ and $u_2(k)$ force $\hat{x}_{u_1(k)}(k)$ and $\hat{x}_{u_2(k)}(k)$ to approach each other while preserving their order. Observe that the approaching condition (5.28) includes the equalizing condition (5.14) as a special case. Furthermore, unlike (5.13) and (5.14), (5.13) and (5.28) do not uniquely determine $\hat{x}_{u_1(k)}(k)$ and $\hat{x}_{u_2(k)}(k)$. Rather, they allow $\hat{x}_{u_1(k)}(k)$ and $\hat{x}_{u_2(k)}(k)$ to increase gradually from $\hat{x}_{u_1(k)}(k-1)$ and decrease accordingly from $\hat{x}_{u_2(k)}(k-1)$, respectively, until the two become equal.

The following lemma characterizes the impact of the non-uniqueness on the value of V:

Lemma 5.3. *Consider* (5.11)–(5.13) *and* (5.28). *Suppose Assumption 5.1 holds. Then, for any given* $(u(k))_{k=1}^{\infty}$, $(V(\mathbf{x}(k)))_{k=0}^{\infty}$ *is non-increasing. Moreover, for any given* $k \in \mathbb{P}$ *and* $\mathbf{x}(k-1) \in \mathcal{X}^N$, $V(\mathbf{x}(k))$ *strictly increases with* $\hat{x}_{u_2(k)}(k) - \hat{x}_{u_1(k)}(k)$ *over* $[0, \hat{x}_{u_2(k)}(k-1) - \hat{x}_{u_1(k)}(k-1)]$.

Proof. Let $(u(k))_{k=1}^{\infty}$ be given. Then, from (5.19), (5.12), and (5.13), we have $V(\mathbf{x}(k)) - V(\mathbf{x}(k-1)) = -\sum_{i \in u(k)} f_i(\hat{x}_i(k)) - f_i(\hat{x}_i(k-1)) - f_i'(\hat{x}_i(k-$

$1))(\hat{x}_i(k) - \hat{x}_i(k-1)) + (f_i'(\hat{x}_i(k-1)) - f_i'(\hat{x}_i(k)))\hat{x}_i(k)\ \forall k \in \mathbb{P}$. Due to (5.13) and (5.28), $\sum_{i\in u(k)}(f_i'(\hat{x}_i(k-1)) - f_i'(\hat{x}_i(k)))\hat{x}_i(k) = (f_{u_1(k)}'(\hat{x}_{u_1(k)}(k-1)) - f_{u_1(k)}'(\hat{x}_{u_1(k)}(k)))(\hat{x}_{u_1(k)}(k) - \hat{x}_{u_2(k)}(k)) \geq 0$. This, along with (5.18), implies $V(\mathbf{x}(k)) - V(\mathbf{x}(k-1)) \leq 0\ \forall k \in \mathbb{P}$. Now let $k \in \mathbb{P}$ and $\mathbf{x}(k-1) \in \mathcal{X}^N$ be given. By Lemma 5.1, there exists a unique $x_{\text{eq}} \in \mathcal{X}$ such that $\sum_{i\in u(k)} f_i'(x_{\text{eq}}) = \sum_{i\in u(k)} f_i'(\hat{x}_i(k))$. Also, $x_{\text{eq}} \in [\hat{x}_{u_1(k)}(k), \hat{x}_{u_2(k)}(k)]$. Let $\mathbf{x}_{\text{eq}} \in \mathcal{X}^N$ be such that its ith entry is x_{eq} if $i \in u(k)$ and $\hat{x}_i(k-1)$ otherwise. Then, it follows from (5.19), (5.12), and (5.18) that $V(\mathbf{x}(k)) - V(\mathbf{x}_{\text{eq}}) = \sum_{i\in u(k)} f_i(x_{\text{eq}}) - f_i(\hat{x}_i(k)) - f_i'(\hat{x}_i(k))(x_{\text{eq}} - \hat{x}_i(k)) \geq 0$. Because $f_i(y) - f_i(x) - f_i'(x)(y-x)$ strictly increases with $|y-x|$ for each fixed $y \in \mathcal{X}\ \forall i \in \mathcal{V}$ and because of (5.13) and (5.28), the second claim is true. □

Lemma 5.3 says that the value of V can never increase. In addition, the closer $\hat{x}_{u_1(k)}(k)$ and $\hat{x}_{u_2(k)}(k)$ get, the larger the value of V drops, and the drop is maximized when $\hat{x}_{u_1(k)}(k)$ and $\hat{x}_{u_2(k)}(k)$ are equalized. These observations suggest that perhaps it is possible to design an algorithm that only forces $\hat{x}_{u_1(k)}(k)$ and $\hat{x}_{u_2(k)}(k)$ to approach each other (as opposed to becoming equal) to the detriment of a smaller drop in the value of V, but at the benefit of not having to share the f_i's. The following algorithm, referred to as *Pairwise Bisectioning* (PB), shows that this is indeed the case and utilizes a bisection step that allows $\hat{x}_{u_1(k)}(k)$ and $\hat{x}_{u_2(k)}(k)$ to get arbitrarily close:

Algorithm 5.3 (Pairwise Bisectioning)**.**

Initialization:

1. Each node $i \in \mathcal{V}$ computes $x_i^* \in \mathcal{X}$, creates variables $\hat{x}_i, a_i, b_i \in \mathcal{X}$, and sets $\hat{x}_i \leftarrow x_i^*$.

Operation: At each iteration:

2. A node with one or more one-hop neighbors, say, node i, initiates the iteration and selects a one-hop neighbor, say, node j, to gossip. Node i transmits $\hat{x}_i$ to node j. Node j sets $a_j \leftarrow \min\{\hat{x}_i, \hat{x}_j\}$ and $b_j \leftarrow \max\{\hat{x}_i, \hat{x}_j\}$ and transmits $\hat{x}_j$ to node i. Node i sets $a_i \leftarrow \min\{\hat{x}_i, \hat{x}_j\}$ and $b_i \leftarrow \max\{\hat{x}_i, \hat{x}_j\}$. Nodes i and j select the number of bisection rounds $R \in \mathbb{P}$.
3. Repeat the following R times: Node j transmits $f_j'(\frac{a_j+b_j}{2}) - f_j'(\hat{x}_j)$ to node i. Node i tests if $f_j'(\frac{a_j+b_j}{2}) - f_j'(\hat{x}_j) + f_i'(\frac{a_i+b_i}{2}) - f_i'(\hat{x}_i) \geq 0$. If so, node i sets $b_i \leftarrow \frac{a_i+b_i}{2}$ and transmits LEFT to node j, and node j sets $b_j \leftarrow \frac{a_j+b_j}{2}$. Otherwise, node i sets $a_i \leftarrow \frac{a_i+b_i}{2}$ and transmits RIGHT to node j, and node j sets $a_j \leftarrow \frac{a_j+b_j}{2}$. End repeat.
4. Node j transmits $f_j'(c_j) - f_j'(\hat{x}_j)$ to node i, where $c_j = \left\{\begin{smallmatrix} a_j & \text{if } \hat{x}_j \leq a_j \\ b_j & \text{if } \hat{x}_j \geq b_j \end{smallmatrix}\right.$. Node i tests if $\Big(f_j'(c_j) - f_j'(\hat{x}_j) + f_i'(c_i) - f_i'(\hat{x}_i)\Big)(\hat{x}_i - \frac{a_i+b_i}{2}) \geq 0$, where $c_i = \left\{\begin{smallmatrix} a_i & \text{if } \hat{x}_i \leq a_i \\ b_i & \text{if } \hat{x}_i \geq b_i \end{smallmatrix}\right.$. If so, node i sets $\hat{x}_i \leftarrow (f_i')^{-1}(f_i'(\hat{x}_i) - f_j'(c_j) + f_j'(\hat{x}_j))$ and node j sets $\hat{x}_j \leftarrow c_j$. Otherwise, node i transmits $f_i'(c_i) - f_i'(\hat{x}_i)$ to node j and sets $\hat{x}_i \leftarrow c_i$, and node j sets $\hat{x}_j \leftarrow (f_j')^{-1}(f_j'(\hat{x}_j) - f_i'(c_i) + f_i'(\hat{x}_i))$.

■

Notice that Step 1 of PB is identical to that of PE except that each node $i \in \mathcal{V}$ creates two additional variables, a_i and b_i, which are used in Step 2 to represent the initial bracket $[a_i, b_i] = [a_j, b_j] = [\min\{\hat{x}_i, \hat{x}_j\}, \max\{\hat{x}_i, \hat{x}_j\}]$ for bisection purposes. Step 3 describes execution of the bisection method, where $R \in \mathbb{P}$ denotes the number of bisection rounds, which may be different for each iteration (e.g., a large R may be advisable when $\hat{x}_i$ and $\hat{x}_j$ are very different). Observe that upon completing Step 3, $x_{\text{eq}} \in [a_i, b_i] = [a_j, b_j] \subset [\min\{\hat{x}_i, \hat{x}_j\}, \max\{\hat{x}_i, \hat{x}_j\}]$ and $b_i - a_i = b_j - a_j = \frac{1}{2^R}|\hat{x}_j - \hat{x}_i|$, where x_{eq} denotes the equalized value of $\hat{x}_i$ and $\hat{x}_j$ if PE were used. Moreover, upon completing Step 4, $x_{\text{eq}} \in [\min\{\hat{x}_i, \hat{x}_j\}, \max\{\hat{x}_i, \hat{x}_j\}] \subset [a_i, b_i] = [a_j, b_j]$, where $\hat{x}_i$ and $\hat{x}_j$

here represent new values. Therefore, upon completing each iteration $k \in \mathbb{P}$,

$$|\hat{x}_{u_1(k)}(k) - \hat{x}_{u_2(k)}(k)| \leq \frac{1}{2^R}|\hat{x}_{u_1(k)}(k-1) - \hat{x}_{u_2(k)}(k-1)|, \quad \forall k \in \mathbb{P}. \quad (5.29)$$

Finally, note that unlike PE which requires two real-number transmissions per iteration, PB requires as many as $3 + R$ or $4 + R$. However, it allows the nodes to never share their f_i's.

The following theorem establishes the asymptotic convergence of PB under Assumption 5.2:

Theorem 5.3. *Consider the use of PB described in Algorithm 5.3. Suppose Assumptions 5.1 and 5.2 hold. Then,* (5.20) *and* (5.3) *hold.*

Proof. See Appendix D.3. □

As it follows from the above, PB represents an alternative to PE, which is useful when nodes are either unable, or unwilling, to share their f_i's. Although not pursued here, it is straightforward to see that PE and PB may be combined, so that equalizing is used when one of the gossiping nodes can send the other its f_i, and approaching is used when none of them can.

5.7 Conclusion

In this chapter, based on the ideas of conservation and dissipation, we have developed PE and PB, two non-gradient-based gossip algorithms that enable nodes to cooperatively solve a class of convex optimization problems over networks. Using Lyapunov stability theory and the convexity structure, we have shown that PE and PB are asymptotically convergent, provided that the gossiping pattern is sufficiently rich. We have also discussed several salient features of PE and PB, including their comparison with the subgradient algorithms and their connection with distributed averaging.

Chapter 6 Control of Distributed Convex Optimization

6.1 Introduction

In many emerging and future applications of multi-agent systems and wired/wireless networks, agents or nodes are required to jointly accomplish sophisticated tasks in distributed fashions. In many instances [57,65], such tasks require the nodes to collaboratively solve, over the network, an unconstrained, separable, convex optimization problem of the form

$$x^* \in \arg\min_x \sum_{i=1}^{N} f_i(x), \tag{6.1}$$

where N is the number of nodes in the network, each f_i is a convex function observed by node i, and x^* is an optimizer every node wants to know.

This chapter is devoted to the design and analysis of distributed algorithms that solve problem (6.1) efficiently. The motivation of the chapter comes from two directions: first, the current literature offers a family of *subgradient algorithms* for solving problem (6.1), including the *incremental* subgradient algorithms [28, 42–44, 57–59, 61, 65], whereby an estimate of x^* is passed around the network, and the *non-incremental* ones [27, 32, 45–47], whereby each node maintains an estimate of x^* and updates it iteratively by exchanging information with neighbors. Although these algorithms are capable of solving the problem, as was pointed out in Chapter 5, they have one or more of the following limitations: (i) the need to select stepsizes, which may not be easy without *a priori* information on the network topology and the f_i's; (ii) the need to construct a Hamiltonian cycle (i.e., a closed path that visits every node exactly once), which may not exist in many networks; and (iii) the need to perform multi-hop passing of the estimate of x^*, which may be costly. To overcome these limitations, in Chapter 5 we develop *Pairwise Equalizing* (PE), a non-gradient-based, gossip-style, distributed asynchronous algorithm that solves problem (6.1) under additional assumptions, bypasses the limitations, and is relatively easy to implement. We also show in Chapter 5 that PE admits a common Lyapunov function for convergence analysis, which is inspired by the first-order convexity condition [9].

Second, the current literature also offers numerous algorithms for solving the closely related problem of *distributed averaging*, including [8, 11, 13, 16, 36, 38, 53, 72, 74, 75, 78]. In Chapter 2, we show that these existing algorithms are bandwidth/energy inefficient, producing networked dynamical systems that evolve with wasteful communications. To improve efficiency in Chapter 2, we develop *Controlled Hopwise Averaging* (CHA), a distributed asynchronous algorithm that enables the nodes to use potential drops in the value of a common quadratic Lyapunov function as feedback to greedily and distributively control when to initiate an iteration. Through extensive simulation on wireless networks modeled by random geometric graphs, we show that CHA is substantially more efficient than several existing schemes, including Pairwise Averaging [72], Consensus Propagation [38], Algorithm A2 of [36], and Distributed Random Grouping [13], requiring far fewer transmissions to complete an averaging task.

In this chapter, we show that ideas from Chapters 2 and 5 may be combined to produce an algorithm that controls the order by which the asynchronous iterations in solving a distributed optimization problem occur. More specifically, we show that it is possible to extend the main idea of CHA in Chapter 2—of using potential drops in the value of a common quadratic Lyapunov function to perform greedy, decentralized feedback iteration control—from solving a distributed averaging problem to solving a distributed optimization problem of the form (6.1). We show that the first-order-convexity-condition-based common Lyapunov function, used to analyze the convergence of PE in Chapter 5, may be used in place of the common quadratic Lyapunov function in Chapter 2 to arrive at a new algorithm, referred to as *Controlled Hopwise Equalizing* (CHE). The chapter begins by developing *Hopwise Equalizing* (HE), an algorithm capable of solving problem (6.2) but perhaps not efficiently so due to not attempting to control how the asynchronous iterations should occur. We show that HE is asymptotically convergent as long as every node participates in the iterations. In addition, we show that HE provides a suitable framework for incorporating the notion of bandwidth/energy-efficient, feedback iteration control, leading to CHE. Finally, via extensive simulation on wirelessly connected, random geometric graphs, we show that CHE is substantially more bandwidth/energy efficient than several existing subgradient algorithms, requiring far fewer transmissions to solve the optimization problem (6.1).

The outline of this chapter is as follows: Section 6.2 formulates the problem. Section 6.3 introduces and analyzes HE, while Section 6.4 transforms it into ICHE and CHE. Section 6.5 compares CHE with PE and several existing subgradient algorithms via simulation. Finally, Section 6.6 concludes the chapter. The proofs of the main results are included in Appendix D. Throughout the chapter, let $\mathbb{N}$, $\mathbb{P}$, $|\cdot|$, f', and f^{-1} denote, respectively, the sets of nonnegative and positive integers, the cardinality of a set, and the derivative and inverse of a function f.

6.2 Problem Formulation

Consider a multi-hop network consisting of $N \geq 2$ nodes, connected by L bidirectional links in a fixed topology. The network is modeled as a connected, undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, \ldots, N\}$ represents the set of N nodes and $\mathcal{E} \subset \{\{i, j\} : i, j \in \mathcal{V}, i \neq j\}$ represents the set of L links. Any two nodes $i, j \in \mathcal{V}$ are one-hop neighbors and can communicate if and only if $\{i, j\} \in \mathcal{E}$. The set of one-hop neighbors of each node $i \in \mathcal{V}$ is denoted as $\mathcal{N}_i = \{j \in \mathcal{V} : \{i, j\} \in \mathcal{E}\}$, and the communications are assumed to be delay- and error-free, with no quantization.

Let $\mathcal{C}$ denote the set of strictly convex, continuously differentiable functions mapping a nonempty open interval $\mathcal{X} \subset \mathbb{R}$ to $\mathbb{R}$ and having a minimizer in $\mathcal{X}$, which must be unique. Suppose each node $i \in \mathcal{V}$ observes a function $f_i \in \mathcal{C}$ with a unique minimizer $x_i^* \in \mathcal{X}$, and all the N nodes wish to solve the following unconstrained, separable, convex optimization problem:

$$\min_{x \in \mathcal{X}} F(x), \tag{6.2}$$

where the objective function $F : \mathcal{X} \to \mathbb{R}$ is defined as

$$F(x) = \sum_{i \in \mathcal{V}} f_i(x). \tag{6.3}$$

As was shown in Chapter 5, F has a unique minimizer $x^* \in \mathcal{X}$ satisfying $F'(x^*) = 0$, so that $F \in \mathcal{C}$ and problem (6.2) is well-posed.

Given the above network, the goal of this chapter is to construct a distributed asynchronous iterative algorithm, which enables all the N nodes to cooperatively solve problem (6.2). The algorithm should be easy to implement and, particularly, be *bandwidth/energy efficient*, driving the estimate $\hat{x}_i$ at each node $i \in \mathcal{V}$ to a sufficiently small neighborhood of the unknown optimizer x^* with relatively few *real-number transmissions*.

6.3 Hopwise Equalizing

In this section, we develop an algorithm, which enables the nodes to collaboratively solve problem (6.2), and which provides a suitable framework for incorporating the notion of bandwidth/energy-efficient, feedback iteration control in Section 6.4.

Consider a networked dynamical system, defined on the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as follows: associated with each link $\{i,j\} \in \mathcal{E}$ are a function $f_{\{i,j\}} \in \mathcal{C}$ and a state variable $x_{\{i,j\}} \in \mathcal{X}$ of the system. Moreover, associated with each node $i \in \mathcal{V}$ is an output variable $\hat{x}_i \in \mathcal{X}$, which represents its estimate of the unknown optimizer x^*. In addition, associated with the system is a control variable $u \in \mathcal{V}$, which dictates its dynamics and represents, physically, the node that initiates an iteration (this physical interpretation will be clear shortly). Let $x_{\{i,j\}}(0)$ and $\hat{x}_i(0)$ represent the initial values of $x_{\{i,j\}}$ and $\hat{x}_i$, $x_{\{i,j\}}(k)$ and $\hat{x}_i(k)$ their values upon completing each iteration $k \in \mathbb{P}$, and $u(k)$ the node that initiates the iteration. Suppose the functions $f_{\{i,j\}}$'s are defined as

$$f_{\{i,j\}} = \frac{1}{|\mathcal{N}_i|} f_i + \frac{1}{|\mathcal{N}_j|} f_j, \quad \forall \{i,j\} \in \mathcal{E}, \tag{6.4}$$

where, because $f_i \in \mathcal{C}\ \forall i \in \mathcal{V}$, we have $f_{\{i,j\}} \in \mathcal{C}\ \forall \{i,j\} \in \mathcal{E}$. Also suppose the initial states $x_{\{i,j\}}(0)$'s are defined as

$$x_{\{i,j\}}(0) = x^*_{\{i,j\}}, \quad \forall \{i,j\} \in \mathcal{E}, \tag{6.5}$$

where $x^*_{\{i,j\}} \in \mathcal{X}$ is the unique minimizer of $f_{\{i,j\}}$, which yields $f'_{\{i,j\}}(x^*_{\{i,j\}}) = 0$.

To describe the system dynamics, consider an iteration $k \in \mathbb{P}$, initiated by node $u(k)$. Suppose, at this iteration k, the state variables associated with links incident to node $u(k)$ are set equal to some $z \in \mathcal{X}$, which satisfies

$$\sum_{\ell \in \mathcal{N}_{u(k)}} f'_{\{u(k),\ell\}}(z) = \sum_{\ell \in \mathcal{N}_{u(k)}} f'_{\{u(k),\ell\}}(x_{\{u(k),\ell\}}(k-1)), \tag{6.6}$$

while the rest of the state variables experience no change in their values, i.e.,

$$x_{\{u(k),j\}}(k) = z, \quad \forall j \in \mathcal{N}_{u(k)}, \tag{6.7}$$

$$x_{\{i,j\}}(k) = x_{\{i,j\}}(k-1), \quad \forall \{i,j\} \in \mathcal{E} - \cup_{\ell \in \mathcal{N}_u(k)} \{u(k), \ell\}. \tag{6.8}$$

The following lemma shows that there always exists a unique $z \in \mathcal{X}$ satisfying (6.6), so that the evolution of the $x_{\{i,j\}}(k)$'s, governed by (6.4)–(6.8), is well-defined:

Lemma 6.1. *For any $n \in \mathbb{P}$, any $g_1, g_2, \ldots, g_n \in \mathcal{C}$, and any $z_1, z_2, \ldots, z_n \in \mathcal{X}$, there exists a unique $z \in \mathcal{X}$ such that $\sum_{i=1}^n g'_i(z) = \sum_{i=1}^n g'_i(z_i)$. Moreover, $z \in [\min_{i \in \{1,2,\ldots,n\}} z_i, \max_{i \in \{1,2,\ldots,n\}} z_i]$.*

Proof. Let $n \in \mathbb{P}$, $g_i \in \mathcal{C}$, and $z_i \in \mathcal{X}$ $\forall i \in \{1,2,\ldots,n\}$. Since $g_i \in \mathcal{C}$ $\forall i$, $g'_i : \mathcal{X} \to \mathbb{R}$ is continuous and strictly increasing $\forall i$, and so is the function $\sum_{i=1}^n g'_i : \mathcal{X} \to \mathbb{R}$. It follows that $\sum_{i=1}^n g'_i(\min_{j \in \{1,2,\ldots,n\}} z_j) \le \sum_{i=1}^n g'_i(z_i) \le \sum_{i=1}^n g'_i(\max_{j \in \{1,2,\ldots,n\}} z_j)$. In addition, by the Intermediate Value Theorem, there exists a unique $z \in \mathcal{X}$ such that $\sum_{i=1}^n g'_i(z) = \sum_{i=1}^n g'_i(z_i)$, and that $z \in [\min_{i \in \{1,2,\ldots,n\}} z_i, \max_{i \in \{1,2,\ldots,n\}} z_i]$. □

Lemma 6.1 has a few implications. First, by induction over k and using the lemma along with (6.4)–(6.8), we see that $x_{\{i,j\}}(k)$ $\forall k \in \mathbb{N}$ $\forall \{i,j\} \in \mathcal{E}$ are well-defined, i.e., are unambiguous and in $\mathcal{X}$. Second, for every iteration $k \in \mathbb{P}$, Lemma 6.1 says that the unique $z \in \mathcal{X}$ satisfying (6.6) also satisfies

$$z \in [\min_{\ell \in \mathcal{N}_{u(k)}} x_{\{u(k),\ell\}}(k-1), \max_{\ell \in \mathcal{N}_{u(k)}} x_{\{u(k),\ell\}}(k-1)]. \tag{6.9}$$

This suggests that one may solve (6.6) and (6.7) for $x_{\{u(k),j\}}(k)$ $\forall j \in \mathcal{N}_{u(k)}$ by first applying a numerical *root-finding method*, such as the *bisection method* with initial bracket given in (6.9), to solve (6.6) for z and then set $x_{\{u(k),j\}}(k)$ $\forall j \in \mathcal{N}_{u(k)}$ to z, via (6.7). Third, from (6.7)–(6.9), we obtain

$$[\min_{\{i,j\} \in \mathcal{E}} x_{\{i,j\}}(k), \max_{\{i,j\} \in \mathcal{E}} x_{\{i,j\}}(k)] \subset [\min_{\{i,j\} \in \mathcal{E}} x_{\{i,j\}}(k-1), \max_{\{i,j\} \in \mathcal{E}} x_{\{i,j\}}(k-1)], \quad \forall k \in \mathbb{P}, \tag{6.10}$$

implying that the closed interval containing the $x_{\{i,j\}}(k)$'s can never grow nor drift over time k. Finally, because there exists a unique $z \in \mathcal{X}$ such that $\sum_{i=1}^{n} g_i'(z) = \sum_{i=1}^{n} g_i'(z_i)$ in Lemma 6.1, we may express this unique z as $z = (\sum_{i=1}^{n} g_i')^{-1}(\sum_{i=1}^{n} g_i'(z_i))$, where $(\sum_{i=1}^{n} g_i')^{-1} : (\sum_{i=1}^{n} g_i')(\mathcal{X}) \to \mathcal{X}$ is the inverse of the injective function $\sum_{i=1}^{n} g_i'$ with its codomain restricted to its range. With this notation, we may similarly express $x_{\{u(k),j\}}(k)$ $\forall j \in \mathcal{N}_{u(k)}$ in (6.7) as $x_{\{u(k),j\}}(k) = z = (\sum_{\ell \in \mathcal{N}_{u(k)}} f'_{\{u(k),\ell\}})^{-1} \sum_{\ell \in \mathcal{N}_{u(k)}} f'_{\{u(k),\ell\}}(x_{\{u(k),\ell\}}(k-1))$ $\forall k \in \mathbb{P}$ $\forall j \in \mathcal{N}_{u(k)}$.

As it follows from the above, the state equation describing the networked dynamical system is given by

$$x_{\{i,j\}}(k) = \begin{cases} (\sum\limits_{\ell \in \mathcal{N}_{u(k)}} f'_{\{u(k),\ell\}})^{-1}(\sum\limits_{\ell \in \mathcal{N}_{u(k)}} f'_{\{u(k),\ell\}}(x_{\{u(k),\ell\}}(k-1))), & \text{if } u(k) \in \{i,j\}, \\ x_{\{i,j\}}(k-1), & \text{otherwise,} \end{cases} \quad \forall k \in \mathbb{P},\ \forall \{i,j\} \in \mathcal{E}. \tag{6.11}$$

To complete description of the system, let its output equation be given by

$$\hat{x}_i(k) = (\sum_{j \in \mathcal{N}_i} f'_{\{i,j\}})^{-1}(\sum_{j \in \mathcal{N}_i} f'_{\{i,j\}}(x_{\{i,j\}}(k))), \quad \forall k \in \mathbb{N},\ \forall i \in \mathcal{V}. \tag{6.12}$$

Observe from (6.11) and (6.12) that the networked dynamical system is a switched, nonlinear system with L state variables $x_{\{i,j\}}(k)$'s, N output variables $\hat{x}_i(k)$'s, and a control variable $u(k)$ that dictates how the system switches. Also notice from Lemma 6.1 that the $\hat{x}_i(k)$'s must satisfy

$$\hat{x}_i(k) \in [\min_{j\in\mathcal{N}_i} x_{\{i,j\}}(k), \max_{j\in\mathcal{N}_i} x_{\{i,j\}}(k)], \quad \forall k \in \mathbb{N},\ \forall i \in \mathcal{V}, \tag{6.13}$$

and that they may be calculated using a root-finding method.

Having described the networked dynamical system (6.11) and (6.12) with initial state (6.5), we now show that it may be realized over the network via a distributed asynchronous algorithm. Suppose each node $i \in \mathcal{V}$, besides maintaining $\hat{x}_i$, maintains a local copy of $f_{\{i,j\}}$ and $x_{\{i,j\}}(k)$ $\forall j \in \mathcal{N}_i$, denoted as f_{ij} and $x_{ij}(k)$, respectively. For each link $\{i,j\} \in \mathcal{E}$, the local copies are meant to be equal, i.e., $f_{ij} = f_{ji}$ and $x_{ij}(k) = x_{ji}(k)$ $\forall k \in \mathbb{N}$, so that the order of the subscripts is only used to indicate where they physically reside. To evolve the system, at each iteration $k \in \mathbb{P}$, a node $u(k)$ would spontaneously initiate the iteration and, along with its one-hop neighbors, would perform a sequence of communication and computing actions. The precise sequence of actions is stated in the following algorithm, which we refer to as *Hopwise Equalizing* (HE), since every iteration involves *equalizing* of state variables within one *hop* of one another:

Algorithm 6.1 (Hopwise Equalizing)**.**

Initialization:

1. Each node $i \in \mathcal{V}$ transmits $|\mathcal{N}_i|$ and f_i to every node $j \in \mathcal{N}_i$.
2. Each node $i \in \mathcal{V}$ creates a function $f_{ij} : \mathcal{X} \to \mathbb{R}$, creates variables $x_{ij} \in \mathcal{X}$ $\forall j \in \mathcal{N}_i$ and $\hat{x}_i \in \mathcal{X}$, and initializes them sequentially:

$f_{ij} \leftarrow \frac{1}{|\mathcal{N}_i|} f_i + \frac{1}{|\mathcal{N}_j|} f_j, \quad \forall j \in \mathcal{N}_i,$

$x_{ij} \leftarrow x^*_{\{i,j\}}, \quad \forall j \in \mathcal{N}_i,$

$\hat{x}_i \leftarrow (\sum_{j \in \mathcal{N}_i} f'_{ij})^{-1} (\sum_{j \in \mathcal{N}_i} f'_{ij}(x_{ij})).$

Operation: At each iteration:

3. A node, say, node i, initiates the iteration.
4. Node i updates x_{ij} $\forall j \in \mathcal{N}_i$:

   $x_{ij} \leftarrow \hat{x}_i, \quad \forall j \in \mathcal{N}_i.$
5. Node i transmits $\hat{x}_i$ to every node $j \in \mathcal{N}_i$.
6. Each node $j \in \mathcal{N}_i$ updates x_{ji} and $\hat{x}_j$ sequentially:

   $x_{ji} \leftarrow \hat{x}_i,$

   $\hat{x}_j \leftarrow (\sum_{\ell \in \mathcal{N}_j} f'_{j\ell})^{-1} (\sum_{\ell \in \mathcal{N}_j} f'_{j\ell}(x_{j\ell})).$ ■

Algorithm 6.1, or HE, consists of an *initialization* part that is executed once, and an *operation* part that is executed iteratively. Note that in Step 1, each node $i \in \mathcal{V}$ transmits f_i $|\mathcal{N}_i|$ times if the network is wired, and only once if it is wireless. Moreover, the node, say, node i, that initiates an iteration, say, iteration k, in Step 3 is, by definition, node $u(k)$ (this explains the physical interpretation made earlier). Furthermore, although (6.11) and (6.12) appear to be somewhat complex, its realization is quite simple, involving only three successive steps (Steps 4–6), and only one of which requires communication (Step 5). Notice that this Step 5 requires transmission of $|\mathcal{N}_i|$ real numbers if the network is wired, and only one real number if it is wireless. Thus, HE is an algorithm capable of fully exploiting the broadcast nature of wireless medium.

Having presented HE, we next show that this algorithm indeed does allow the nodes to collaboratively solve problem (6.2), i.e., ensuring that

$$\lim_{k \to \infty} \hat{x}_i(k) = x^*, \quad \forall i \in \mathcal{V}. \tag{6.14}$$

To establish (6.14), let $\mathbf{x}^* \in \mathcal{X}^L$ and $\mathbf{x}(k) \in \mathcal{X}^L\ \forall k \in \mathbb{N}$ denote, respectively, the vectors obtained by stacking L copies of x^* and all the $x_{\{i,j\}}(k)$'s. Notice from Lemma 6.1 that $\mathbf{x}^*$ is an equilibrium point of the system (6.11) and (6.12). To show that $\mathbf{x}(k)$ would asymptotically converge to the equilibrium point $\mathbf{x}^*$, recall that for any strictly convex and differentiable function $f : \mathcal{X} \to \mathbb{R}$, the first-order convexity condition says that

$$f(y) \geq f(x) + f'(x)(y - x), \quad \forall x, y \in \mathcal{X}, \tag{6.15}$$

where the equality holds if and only if $x = y$. This property suggests the following Lyapunov function candidate $V : \mathcal{X}^L \subset \mathbb{R}^L \to \mathbb{R}$, which uses the convexity of the $f_{\{i,j\}}$'s:

$$V(\mathbf{x}(k)) = \sum_{\{i,j\}\in\mathcal{E}} f_{\{i,j\}}(x^*) - f_{\{i,j\}}(x_{\{i,j\}}(k)) - f'_{\{i,j\}}(x_{\{i,j\}}(k))(x^* - x_{\{i,j\}}(k)). \tag{6.16}$$

Because $f_{\{i,j\}} \in \mathcal{C}\ \forall \{i,j\} \in \mathcal{C}$ and because of the first-order convexity condition (6.15), V in (6.16) is continuous and positive definite with respect to $\mathbf{x}^*$, i.e., $V(\mathbf{x}(k)) \geq 0\ \forall \mathbf{x}(k) \in \mathcal{X}^L$, and $V(\mathbf{x}(k)) = 0$ if and only if $\mathbf{x}(k) = \mathbf{x}^*$. Hence, if

$$\lim_{k\to\infty} V(\mathbf{x}(k)) = 0, \tag{6.17}$$

then

$$\lim_{k\to\infty} x_{\{i,j\}}(k) = x^*, \quad \forall \{i,j\} \in \mathcal{E}, \tag{6.18}$$

which, along with (6.12) and Lemma 6.1, implies (6.14).

To establish (6.17), consider the following lemma:

Lemma 6.2. *Consider the network modeled in Section 6.2 and the use of HE described in Algorithm 6.1. Then, for any given sequence* $(u(k))_{k=1}^{\infty}$*, the* *sequence* $(V(\mathbf{x}(k)))_{k=0}^{\infty}$ *is non-increasing and satisfies*

$$V(\mathbf{x}(k))-V(\mathbf{x}(k-1))=-\sum_{j\in\mathcal{N}_{u(k)}} f_{\{u(k),j\}}(\hat{x}_{u(k)}(k-1))-f_{\{u(k),j\}}(x_{\{u(k),j\}}(k-1)) - f'_{\{u(k),j\}}(x_{\{u(k),j\}}(k-1))(\hat{x}_{u(k)}(k-1)-x_{\{u(k),j\}}(k-1)), \quad \forall k\in\mathbb{P}. \tag{6.19}$$

Proof. Let $(u(k))_{k=1}^{\infty}$ be given. Then, it follows from (6.16) and (6.11) that

$$\begin{aligned} &V(\mathbf{x}(k))-V(\mathbf{x}(k-1))\\ &=-\sum_{j\in\mathcal{N}_{u(k)}} f_{\{u(k),j\}}(x_{\{u(k),j\}}(k))-f_{\{u(k),j\}}(x_{\{u(k),j\}}(k-1))+f'_{\{u(k),j\}}(x_{\{u(k),j\}}(k))x^*\\ &\quad - f'_{\{u(k),j\}}(x_{\{u(k),j\}}(k-1))x^* - f'_{\{u(k),j\}}(x_{\{u(k),j\}}(k))x_{\{u(k),j\}}(k)\\ &\quad + f'_{\{u(k),j\}}(x_{\{u(k),j\}}(k-1))x_{\{u(k),j\}}(k-1), \quad \forall k\in\mathbb{P}. \end{aligned}$$

Due to (6.11), $-\sum_{j\in\mathcal{N}_{u(k)}} f'_{\{u(k),j\}}(x_{\{u(k),j\}}(k))x^*$ in the right-hand side of the above equation cancels $\sum_{j\in\mathcal{N}_{u(k)}} f'_{\{u(k),j\}}(x_{\{u(k),j\}}(k-1))x^*$, while $\sum\limits_{j\in\mathcal{N}_{u(k)}} f'_{\{u(k),j\}}(x_{\{u(k),j\}}(k))x_{\{u(k),j\}}(k) = \sum\limits_{j\in\mathcal{N}_{u(k)}} f'_{\{u(k),j\}}(x_{\{u(k),j\}}(k-1))x_{\{u(k),j\}}(k)$. Moreover, notice from (6.11) and (6.12) that $x_{\{u(k),j\}}(k)=\hat{x}_{u(k)}(k-1)$. Thus, (6.19) holds. Moreover, since the right-hand side of (6.19) is nonpositive, $(V(\mathbf{x}(k)))_{k=0}^{\infty}$ is non-increasing. □

Lemma 6.2, together with (6.15), says that upon completing each iteration $k\in\mathbb{P}$, the value of V must either decrease or, at worst, stay the same, no matter which nodes initiates the iteration, i.e., no matter what $u(k)$ is. In addition, $V(\mathbf{x}(k))=V(\mathbf{x}(k-1))$ if and only if $x_{\{u(k),j\}}(k-1)$ $\forall j\in\mathcal{N}_{u(k)}$ are equal. Second, since $(V(\mathbf{x}(k)))_{k=0}^{\infty}$ is non-increasing regardless of $(u(k))_{k=1}^{\infty}$, V in (6.16) may be viewed as a *common* Lyapunov function for the nonlinear switched system (6.11). Finally, observe that the first-order convexity condition (6.15) can be used not only to form the common Lyapunov function V, but also to characterize drops in its value, as shown in (6.19).

Since $V(\mathbf{x}(k)) \leq V(\mathbf{x}(k-1))\ \forall k \in \mathbb{P}$ and $V(\mathbf{x}(k)) \geq 0$, the limit $\lim_{k\to\infty} V(\mathbf{x}(k))$ exists and is nonnegative. Obviously, this property does not mean $\lim_{k\to\infty} V(\mathbf{x}(k)) = 0$. In fact, in general, $\lim_{k\to\infty} V(\mathbf{x}(k))$ may be positive if some nodes in the network never initiate an iteration. Therefore, to establish (6.17), some restrictions must be imposed on the "initiation" pattern. The following theorem provides a mild, sufficient condition on the initiation pattern, with which $\lim_{k\to\infty} V(\mathbf{x}(k)) = 0$ is guaranteed:

Theorem 6.1. *Consider the network modeled in Section 6.2 and the use of HE described in Algorithm 6.1. Suppose each node $i \in \mathcal{V}$ appears infinitely often in the sequence $(u(k))_{k=1}^{\infty}$. Then,* (6.17), (6.18), *and* (6.14) *hold.*

Proof. See Appendix E.1. □

Theorem 6.1 says that, as long as every node initiates an iteration infinitely many times, HE is asymptotically convergent, allowing the nodes to cooperatively solve problem (6.2). With the development of HE, we have also provided a suitable framework, to be utilized next for incorporating the notion of feedback iteration control.

6.4 Controlled Hopwise Equalizing

HE operates by executing (6.11) according to the sequence $(u(k))_{k=1}^{\infty}$ of nodes that initiate the iterations. Although Theorem 6.1 says that essentially every sequence $(u(k))_{k=1}^{\infty}$ can drive all the $\hat{x}_i(k)$'s in (6.12) to an arbitrarily small neighborhood of x^*, certain $(u(k))_{k=1}^{\infty}$ may be better at doing so than others, requiring fewer iterations and, thus, fewer real-number transmissions, making them more bandwidth/energy efficient. This raises the question of how to generate those efficient $(u(k))_{k=1}^{\infty}$'s distributively in real-time. To address this question, note that with HE, the sequence $(u(k))_{k=1}^{\infty}$ is undefined initially and is gradually defined, one element per iteration, as time elapses, i.e., when a node $i \in \mathcal{V}$ initiates an iteration $k \in \mathbb{P}$, the element $u(k)$ becomes defined and is given by $u(k) = i$. Thus, if we let the nodes control *when* to initiate an iteration, presumably using some form of locally available feedback, they may jointly shape the value of $(u(k))_{k=1}^{\infty}$, making it bandwidth/energy efficient.

As it turns out, HE offers a suitable setup which, together with the common Lyapunov function V in (6.16), enables the nodes to perform greedy, decentralized, feedback iteration control similar to that in Chapter 2. To see this, for each $i \in \mathcal{V}$, define $\Delta V_i : \mathcal{X}^L \to \mathbb{R}$ as

$$\begin{aligned}\Delta V_i(\mathbf{x}(k)) = \sum_{j \in \mathcal{N}_i} & f_{\{i,j\}}(\hat{x}_i(k)) - f_{\{i,j\}}(x_{\{i,j\}}(k)) \\ & - f'_{\{i,j\}}(x_{\{i,j\}}(k))(\hat{x}_i(k) - x_{\{i,j\}}(k)). \end{aligned} \tag{6.20}$$

Observe that each $\Delta V_i(\mathbf{x}(k))$ in (6.20) is in the form of the first-order convexity condition (6.15) and, thus, is nonnegative, taking the value of zero if and only if $x_{\{i,j\}}(k) = \hat{x}_i(k)\ \forall j \in \mathcal{N}_i$. Also note from Lemma 6.2 that for any $u(k) \in \mathcal{V}$, upon completing iteration k, the value of V would drop from $V(\mathbf{x}(k-1))$ to $V(\mathbf{x}(k))$ by an amount equal to $\Delta V_{u(k)}(\mathbf{x}(k))$, i.e.,

$$V(\mathbf{x}(k)) - V(\mathbf{x}(k-1)) = -\Delta V_{u(k)}(\mathbf{x}(k-1)),\ \forall k \in \mathbb{P}. \tag{6.21}$$

An immediate consequence of (6.21) is that the nodes may collaboratively produce efficient $(u(k))_{k=1}^{\infty}$ by letting every iteration $k \in \mathbb{P}$ be initiated by a node $u(k)$ having a relatively large $\Delta V_{u(k)}(\mathbf{x}(k-1))$, because the larger $\Delta V_{u(k)}(\mathbf{x}(k-1))$, the faster $V(\mathbf{x}(k))$ decreases to zero and, thus, the faster the $x_{\{i,j\}}(k)$'s and $\hat{x}_i(k)$'s converge to x^*. In other words, *nodes with larger*

$\Delta V_i(\mathbf{x}(\cdot))$*'s would rush to initiate the next iterations, while nodes with smaller* $\Delta V_i(\mathbf{x}(\cdot))$*'s would wait longer.* Of course, this approach to feedback iteration control is possible only if each node $i \in \mathcal{V}$ knows its own $\Delta V_i(\mathbf{x}(\cdot))$. According to (6.20), this is indeed the case, since $f_{\{i,j\}}$, $x_{\{i,j\}}(k)$, and $\hat{x}_i(k)$ $\forall j \in \mathcal{N}_i$ are all known to each node $i \in \mathcal{V}$.

The preceding paragraph describes a Lyapunov-based, greedy, decentralized approach to feedback iteration control, whereby each node $i \in \mathcal{V}$ uses its own potential drop $\Delta V_i(\mathbf{x}(\cdot))$ as feedback to control when to initiate an iteration, focusing on making the value of V drop as much as possible every time without worrying about the future (hence the term *greedy*). We point out that the idea of trying to control a distributed convex optimization process has not been explored in the literature [27, 28, 32, 42–47, 57–59, 61, 65]. Thus, the proposed approach represents a main contribution of this chapter.

Given that the nodes strive to behave greedily, a case of interest is how would the resulting algorithm perform if every iteration $k \in \mathbb{P}$ turns out to be initiated by a node i having the *largest* $\Delta V_i(\mathbf{x}(k-1))$, i.e.,

$$u(k) \in \arg\max_{i \in \mathcal{V}} \Delta V_i(\mathbf{x}(k-1)), \quad \forall k \in \mathbb{P}. \tag{6.22}$$

The following algorithm, referred to as *Ideal Controlled Hopwise Equalizing* (ICHE), realizes this ideal operation:

Algorithm 6.2 (Ideal Controlled Hopwise Equalizing)**.**

Initialization:

1. Each node $i \in \mathcal{V}$ transmits $|\mathcal{N}_i|$ and f_i to every node $j \in \mathcal{N}_i$.
2. Each node $i \in \mathcal{V}$ creates a function $f_{ij} : \mathcal{X} \to \mathbb{R}$, creates variables $x_{ij} \in \mathcal{X}$ $\forall j \in \mathcal{N}_i$, $\hat{x}_i \in \mathcal{X}$, and $\Delta V_i \in [0, \infty)$, and initializes them sequentially:

$f_{ij} \leftarrow \frac{1}{|\mathcal{N}_i|} f_i + \frac{1}{|\mathcal{N}_j|} f_j, \quad \forall j \in \mathcal{N}_i,$

$x_{ij} \leftarrow x^*_{\{i,j\}}, \quad \forall j \in \mathcal{N}_i,$

$\hat{x}_i \leftarrow (\sum_{j\in\mathcal{N}_i} f'_{ij})^{-1} (\sum_{j\in\mathcal{N}_i} f'_{ij}(x_{ij})),$

$\Delta V_i \leftarrow \sum_{j\in\mathcal{N}_i} f_{ij}(\hat{x}_i) - f_{ij}(x_{ij}) - f'_{ij}(x_{ij})(\hat{x}_i - x_{ij}).$

Operation: At each iteration:

3. Let $i \in \arg\max_{j\in\mathcal{V}} \Delta V_j$.
4. Node i updates x_{ij} $\forall j \in \mathcal{N}_i$ and ΔV_i sequentially:

   $x_{ij} \leftarrow \hat{x}_i, \quad \forall j \in \mathcal{N}_i,$

   $\Delta V_i \leftarrow 0.$
5. Node i transmits $\hat{x}_i$ to every node $j \in \mathcal{N}_i$.
6. Each node $j \in \mathcal{N}_i$ updates x_{ji}, $\hat{x}_j$, and ΔV_j sequentially:

   $x_{ji} \leftarrow \hat{x}_i,$

   $\hat{x}_j \leftarrow (\sum_{\ell\in\mathcal{N}_j} f'_{j\ell})^{-1} (\sum_{\ell\in\mathcal{N}_j} f'_{j\ell}(x_{j\ell})),$

   $\Delta V_j \leftarrow \sum_{\ell\in\mathcal{N}_j} f_{j\ell}(\hat{x}_j) - f_{j\ell}(x_{j\ell}) - f'_{j\ell}(x_{j\ell})(\hat{x}_j - x_{j\ell}).$ ■

Note that Algorithm 6.2, or ICHE, is similar to HE except for their Step 3 and except that with ICHE, each node $i \in \mathcal{V}$ also maintains its own $\Delta V_i(\mathbf{x}(\cdot))$. The following theorem asserts that ICHE is always asymptotically convergent:

Theorem 6.2. *Consider the network modeled in Section 6.2 and the use of ICHE described in Algorithm 6.2. Then,* (6.17)*,* (6.18)*, and* (6.14) *hold.*

Proof. See Appendix E.2. □

ICHE described above is not implementable because without a centralized node, it is difficult to realize (6.22), since it is necessary to collect all the $\Delta V_i(\mathbf{x}(k-1))$'s, compare them, and inform the node with the largest $\Delta V_i(\mathbf{x}(k-1))$ to initiate the next iteration. Fortunately, it is possible to closely mimic this greedy behavior of ICHE in a decentralized fashion. To see this, suppose each node $i \in \mathcal{V}$ maintains a time-to-initiate variable $\tau_i > 0$, so that when time t advances to $t = \tau_i$, node i initiates the next iteration. Suppose also that τ_i is a function of ΔV_i,

$$\tau_i(k-1) = \Phi(\Delta V_i(\mathbf{x}(k-1))), \tag{6.23}$$

where $\Phi : [0, \infty) \to (0, \infty]$ is a continuous, strictly decreasing function satisfying $\lim_{v \to 0} \Phi(v) = \infty$ and $\Phi(0) = \infty$. Because Φ is strictly decreasing, the larger $\Delta V_i(\mathbf{x}(k-1))$, the smaller $\tau_i(k-1)$. Thus, with (6.23), the node with the largest $\Delta V_i(\mathbf{x}(k-1))$'s would become the node that initiates iteration k. Note that although (6.23) tries to foster a greedy behavior, it has two limitations. First, τ_i may become smaller than t upon completion of an iteration, which is undesirable because every node's time to initiate the next iteration should be in the future, not the past. Second, it is theoretically possible that $\tau_i = \tau_j$ for some $i, j \in \mathcal{V}$, so that if nodes i and j are one-hop neighbors, then a collision would occur. To overcome these two limitations, consider the following slight modification of (6.23):

$$\tau_i(k-1) = \max\{\Phi(\Delta V_i(\mathbf{x}(k-1))), t\} + \varepsilon(\Delta V_i(\mathbf{x}(k-1))) \cdot \text{rand}(), \tag{6.24}$$

where $\varepsilon : [0, \infty) \to (0, \infty)$ is a continuous function meant to be small and positive, and each call to rand() returns a uniformly distributed random variable on the unit interval. Note from (6.24) that the $\max\{\cdot, \cdot\}$ function is intended to ensure that τ_i is never less than t. Moreover, inserting a little randomness into (6.24) reduces the probability of collision.

Observe that with each node $i \in \mathcal{V}$ utilizing the above decentralized strategy for controlling when to initiate an iteration, the resulting system becomes a simple discrete-event system, in which there are always N events scheduled, one from each node. The node $u(k)$ that initiates iteration k can therefore be determined from

$$u(k) = \arg\min_{i \in \mathcal{V}} \tau_i(k-1), \quad \forall k \in \mathbb{P}. \tag{6.25}$$

The following practical algorithm, referred to as *Controlled Hopwise Equalizing* (CHE), realizes this strategy:

Algorithm 6.3 (Controlled Hopwise Equalizing)**.**

Initialization:

1. Let time $t = 0$.
2. Each node $i \in \mathcal{V}$ transmits $|\mathcal{N}_i|$ and f_i to every node $j \in \mathcal{N}_i$.
3. Each node $i \in \mathcal{V}$ creates a function $f_{ij} : \mathcal{X} \to \mathbb{R}$, creates variables $x_{ij} \in \mathcal{X}$ $\forall j \in \mathcal{N}_i$, $\hat{x}_i \in \mathcal{X}$, $\Delta V_i \in [0, \infty)$, and $\tau_i \in (0, \infty]$, and initializes them sequentially:

   $f_{ij} \leftarrow \frac{1}{|\mathcal{N}_i|} f_i + \frac{1}{|\mathcal{N}_j|} f_j, \quad \forall j \in \mathcal{N}_i,$

   $x_{ij} \leftarrow x^*_{\{i,j\}}, \quad \forall j \in \mathcal{N}_i,$

   $\hat{x}_i \leftarrow (\sum_{j \in \mathcal{N}_i} f'_{ij})^{-1} (\sum_{j \in \mathcal{N}_i} f'_{ij}(x_{ij})),$

   $\Delta V_i \leftarrow \sum_{j \in \mathcal{N}_i} f_{ij}(\hat{x}_i) - f_{ij}(x_{ij}) - f'_{ij}(x_{ij})(\hat{x}_i - x_{ij}),$

   $\tau_i \leftarrow \max\{\Phi(\Delta V_i), t\} + \varepsilon(\Delta V_i) \cdot \text{rand}().$

Operation: At each iteration:

4. Let $i \in \arg\min_{j \in \mathcal{V}} \tau_j$ and $t = \tau_i$.
5. Node i updates x_{ij} $\forall j \in \mathcal{N}_i$, ΔV_i, and τ_i sequentially:

   $x_{ij} \leftarrow \hat{x}_i, \quad \forall j \in \mathcal{N}_i,$

$\Delta V_i \leftarrow 0$,

$\tau_i \leftarrow \infty$.

6. Node i transmits $\hat{x}_i$ to every node $j \in \mathcal{N}_i$.
7. Each node $j \in \mathcal{N}_i$ updates x_{ji}, $\hat{x}_j$, ΔV_j, and τ_j sequentially:

$x_{ji} \leftarrow \hat{x}_i$,

$\hat{x}_j \leftarrow (\sum_{\ell \in \mathcal{N}_j} f'_{j\ell})^{-1}(\sum_{\ell \in \mathcal{N}_j} f'_{j\ell}(x_{j\ell}))$,

$\Delta V_j \leftarrow \sum_{\ell \in \mathcal{N}_j} f_{j\ell}(\hat{x}_j) - f_{j\ell}(x_{j\ell}) - f'_{j\ell}(x_{j\ell})(\hat{x}_j - x_{j\ell})$,

$\tau_j \leftarrow \max\{\Phi(\Delta V_j), t\} + \varepsilon(\Delta V_j) \cdot \text{rand}()$. ■

6.5 Performance Comparison

In this section, we compare CHE with *Pairwise Equalizing* (PE), developed in Chapter 5, as well as with five existing subgradient algorithms, namely, incremental with cyclic passing of the latest estimate [43] (referred to here as NedBer'01-C), incremental with random equiprobable passing [42] (NedBer'01-R), incremental with random Markov chain-based one-hop passing [61] (RamNedVee'09), synchronous with one consensus iteration per update [46] (NedOzd'09), and synchronous with $\varphi = 5$ consensus iterations per update [27] (JKJJ'08). Due to space limitations, we omit detailed description of the implementation of these algorithms.

Suppose each node i observes a function $f_i : \mathbb{R} \to \mathbb{R}$ of the form $f_i(x) = a_i x + b_i(x - c_i)^2 + d_i(x - e_i)^4$, where $b_i, d_i \in (0, 1)$, $c_i, e_i \in (-1, 1)$, and $a_i \in (-2b_i(1 - c_i) - 4(d_i - e_i)^3, 2b_i(1 + c_i) + 4d_i(1 + e_i)^3)$, so that $x_i^* \in (-1, 1)$. For each of the subgradient algorithms, we choose a diminishing stepsize of the form $\alpha(k) = \frac{a}{k}$, where $a \in (0, \infty)$. After some fine-tuning of the value of a, we arrive at $a = 0.05$ for NedBer'01-C, $a = 0.02$ for NedBer'01-R and

RamNedVee'09, $a = 0.018$ for NedOzd'09, and $a = 0.024$ for JKJJ'08. To ensure a fair comparison, overhead transmissions are included for the following cases: for RamNedVee'09, NedOzd'09, and JKJJ'08, where a weight scheme is used, N overhead transmission are counted for each node $i \in \mathcal{V}$ to obtain $|\mathcal{N}_j|$ $\forall j \in \mathcal{N}_j$; and for PE and CHE, $5N$ overhead transmissions are counted for the transmission of a_i, b_i, c_i, d_i, and e_i. Note that for each subgradient algorithm, any overhead required to determine the value of a is excluded. Moreover, for NedBer'01-C, we use a *pseudo-Hamiltonian cycle*, or the shortest tour that visits each node of the network, to realize the cyclic passing, since a Hamiltonian cycle is not guaranteed to exist. The overhead required to construct such a tour is also excluded.

To compare these algorithms, we generate random geometric wirelessly connected graphs of $N \in \{100, 200, 300, 400, 500\}$ nodes and choose the one-hop radius such that the average number of neighbors $\frac{2L}{N} \in \{10, 20, 30, 40, 50, 60\}$. For each N and $\frac{2L}{N}$, 240 different networks were generated, and for every network, each algorithm was simulated for sufficiently many iterations to find the smallest $K \in \mathbb{P}$ such that $|\hat{x}_i(k) - x^*| \leq 0.005$ $\forall i \in \mathcal{V}$, $\forall k \geq K$. Then we record convergence as the number of real-number transmissions needed to carry out the K iterations.

Figure 6.1 shows the number of real-number transmissions needed to converge, averaged over the 240 scenarios, as a function of the number of nodes N and the average number of neighbors $\frac{2L}{N}$. Observe that, in general, the non-incremental subgradient algorithms are the least efficient, with JKJJ'08 achieving better performance than NedOzd'09, particularly in networks of larger $\frac{2L}{N}$. The next in line in terms of efficiency is PE, which also achieves better relative performance for larger $\frac{2L}{N}$. Thirdly, the random incremental subgradient algorithms (NedBer'01-R and RamNedVee'09) perform even better, especially in large networks. Finally, NedBer01-C and CHE have the best efficiency with NedBer'01-C performing slightly better than CHE for larger networks. However, NetBer'01-C requires the construction of pseudo-Hamiltonian cycles, which may be difficult to do in a distributive fashion and which is not penalized in this comparison. In contrast, CHE achieves comparable performance without needing such cycles.

6.6 Conclusion

In this chapter, we have addressed the problem of solving unconstrained, separable, convex optimization problems over networks. We have introduced a new concept in solving such problems: control of distributed convex optimization. We have developed and analyzed three algorithms—namely, HE, ICHE, and CHE—showing along the way that a common Lyapunov function, constructed based on the first-order convexity condition, can be used to incorporate the notion of greedy, decentralized, feedback iteration control, whereby individual nodes use potential drops in the value of the Lyapunov function to control when to initiate an iteration. Finally, via extensive simulation on wirelessly connected random geometric graphs, we have shown that CHE is significantly more bandwidth/energy efficient than several existing subgradient algorithms, requiring far fewer transmissions to solve a convex optimization problem.

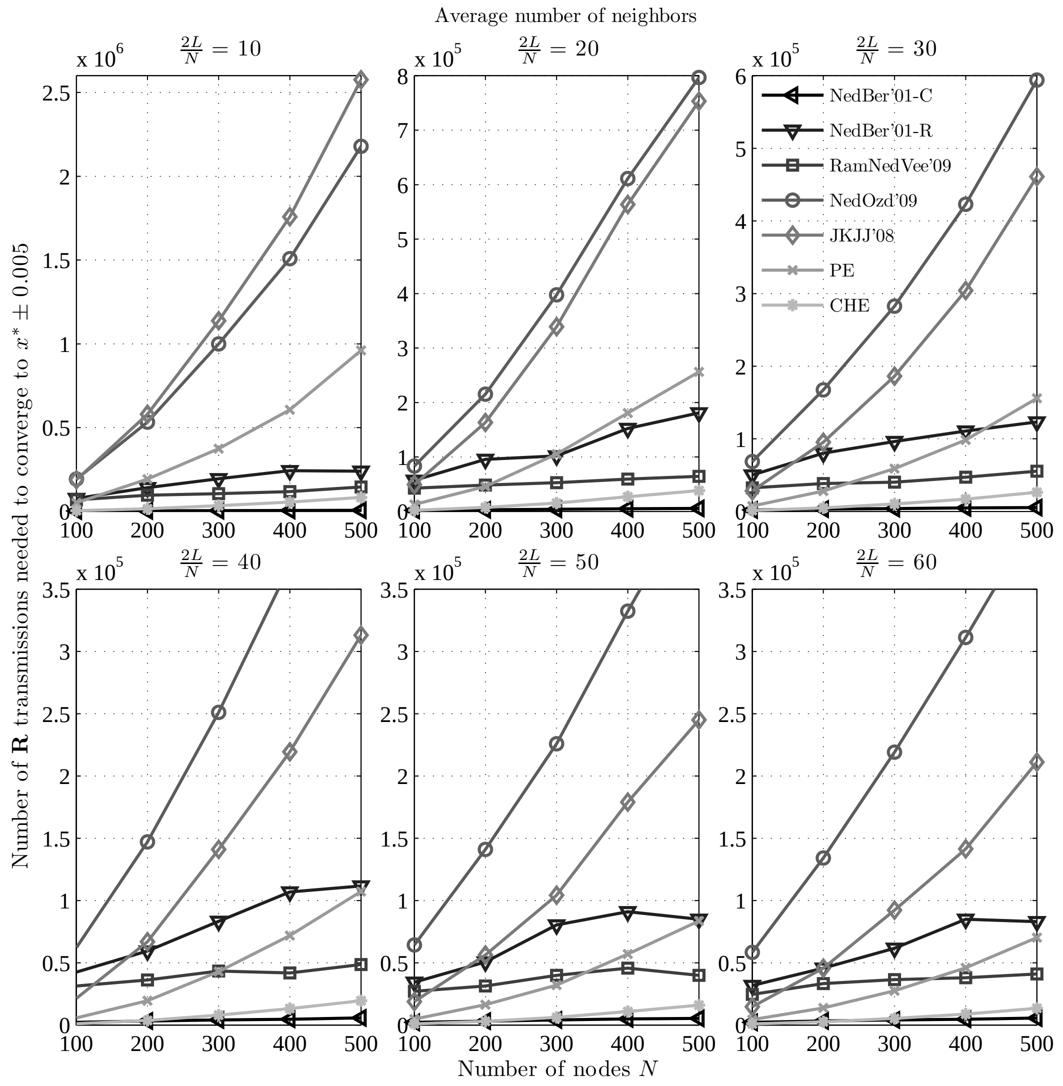


Figure 6.1: Performance comparison on random geometric wireless networks with varying number of nodes N and average number of neighbors $\frac{2L}{N}$.

Chapter 7 Zero-Gradient-Sum Algorithms for Distributed Convex Optimization: The Continuous-Time Case

7.1 Introduction

This chapter addresses the problem of solving an unconstrained, separable, convex optimization problem over an N-node multi-hop network, where each node i observes a convex function f_i, and all the N nodes wish to determine an optimizer x^* that minimizes the sum of the f_i's, i.e.,

$$x^* \in \arg\min_x \sum_{i=1}^{N} f_i(x). \tag{7.1}$$

The problem (7.1) arises in many emerging and future applications of multi-agent systems and wired/wireless/social networks, where agents or nodes often need to collaborate in order to jointly accomplish sophisticated tasks in decentralized and optimal fashions [57].

To date, a family of discrete-time *subgradient algorithms*, aimed at solving problem (7.1) under general convexity assumptions, have been reported in the literature. These subgradient algorithms may be roughly classified into two groups. The first group of algorithms [6, 28, 57, 61] are *incremental* in nature, relying on the passing of an estimate of x^* around the network to operate. The second group of algorithms [27, 47, 62] are *non-incremental*, relying instead on a combination of subgradient updates and linear consensus iterations to operate, although gossip-based updates have also been considered [60]. For each of these algorithms, a number of convergence properties have been established, including the resulting error bounds, asymptotic convergence, and convergence rates.

In Chapter 5, we introduced a gossip-style, distributed asynchronous algorithm, referred to as *Pairwise Equalizing* (PE), which solves the scalar version of problem (7.1), in a manner that is fundamentally different from the aforementioned subgradient algorithms (e.g., PE does not try to move along the gradient, nor does it require the notion of a stepsize). In Chapter 6, we showed that the two basic ideas behind PE—namely, the conservation of a certain gradient sum at zero and the use of a convexity-inspired Lyapunov function—can be extended, leading to *Controlled Hopwise Equalizing* (CHE), a distributed asynchronous algorithm that allows individual nodes to use potential drops in the value of the Lyapunov function to control, on their own, when to initiate an iteration, so that problem (7.1) may be solved efficiently. In both the chapters, problem (7.1) was studied in a discrete-time, asynchronous setting, and only the scalar version of it was considered.

In this chapter, we address problem (7.1) from a continuous-time and multi-dimensional standpoint, building upon the two basic ideas behind PE. Specifically, using the same Lyapunov function candidate as the one for PE and CHE, we first derive a family of continuous-time distributed algorithms called *Zero-Gradient-Sum* (ZGS) algorithms, with which the states of the resulting nonlinear networked dynamical systems slide along an invariant, zero-gradient-sum manifold and converge asymptotically to the unknown minimizer x^* in (7.1). We then describe a systematic way to construct ZGS algorithms and prove that a subset of them are exponentially convergent. For this subset of algorithms, we also obtain lower and upper bounds on their convergence rates as functions of the network topologies, problem characteristics, and algorithm parameters, including the algebraic connectivity, Laplacian spectral radius, and curvatures of the f_i's. As another contribution of this chapter, we show that some of the existing continuous-time distributed consensus algorithms (e.g., [24, 51, 52, 63, 66, 69]) are special cases of ZGS algorithms and are, interestingly, just a slight modification away from solving any problem of the form (7.1). In addition, the well-known result from [52], which says that the convergence rate of a linear consensus algorithm is characterized by the algebraic connectivity of the underlying graph, is a special case of Theorem 7.2 here.

The outline of this chapter is as follows: Section 7.2 provides some preliminaries. Section 7.3 formulates the problem. Sections 7.4 characterizes and constructs ZGS algorithms, and Section 7.5 analyzes their convergence rates. Finally, Section 7.6 concludes the chapter.

7.2 Preliminaries

A twice continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is *locally strongly convex* if for any convex and compact set $D \subset \mathbb{R}^n$, there exists a constant $\theta > 0$ such that the following equivalent conditions hold [23, 48]:

$$f(y) - f(x) - \nabla f(x)^T (y - x) \geq \frac{\theta}{2}\|y - x\|^2, \quad \forall x, y \in D, \tag{7.2}$$

$$(\nabla f(y) - \nabla f(x))^T (y - x) \geq \theta \|y - x\|^2, \quad \forall x, y \in D, \tag{7.3}$$

$$\nabla^2 f(x) \geq \theta I_n, \quad \forall x \in D, \tag{7.4}$$

where $\|\cdot\|$ denotes the Euclidean norm, $\nabla f : \mathbb{R}^n \to \mathbb{R}^n$ is the gradient of f, $\nabla^2 f : \mathbb{R}^n \to \mathbb{R}^{n\times n}$ is the Hessian of f, and $I_n \in \mathbb{R}^{n\times n}$ is the identity matrix.

The function f is *strongly convex* if there exists a constant $\theta > 0$ such that the equivalent conditions (7.2)–(7.4) hold for $D = \mathbb{R}^n$, in which case θ is called the *convexity parameter* of f [48]. Finally, for any twice continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, any convex set $D \subset \mathbb{R}^n$, and any constant $\Theta > 0$, the following conditions are equivalent [9, 48]:

$$f(y) - f(x) - \nabla f(x)^T(y - x) \leq \frac{\Theta}{2}\|y - x\|^2, \quad \forall x, y \in D, \tag{7.5}$$

$$(\nabla f(y) - \nabla f(x))^T(y - x) \leq \Theta\|y - x\|^2, \quad \forall x, y \in D, \tag{7.6}$$

$$\nabla^2 f(x) \leq \Theta I_n, \quad \forall x \in D. \tag{7.7}$$

7.3 Problem Formulation

Consider a multi-hop network consisting of $N \geq 2$ nodes, connected by bidirectional links in a fixed topology. The network is modeled as a connected, undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, \ldots, N\}$ represents the set of N nodes and $\mathcal{E} \subset \{\{i, j\} : i, j \in \mathcal{V}, i \neq j\}$ represents the set of links. Any two nodes $i, j \in \mathcal{V}$ are one-hop neighbors and can communicate if and only if $\{i, j\} \in \mathcal{E}$. The set of one-hop neighbors of each node $i \in \mathcal{V}$ is denoted as $\mathcal{N}_i = \{j \in \mathcal{V} : \{i, j\} \in \mathcal{E}\}$, and the communications are assumed to be delay- and error-free, with no quantization.

Suppose each node $i \in \mathcal{V}$ observes a function $f_i : \mathbb{R}^n \to \mathbb{R}$ satisfying the following assumption:

Assumption 7.1. For each $i \in \mathcal{V}$, the function f_i is twice continuously differentiable, strongly convex with convexity parameter $\theta_i > 0$, and has a locally Lipschitz Hessian $\nabla^2 f_i$.

Suppose, upon observing the f_i's, all the N nodes wish to solve the following unconstrained, separable, convex optimization problem:

$$\min_{x \in \mathbb{R}^n} F(x), \tag{7.8}$$

where the objective function $F : \mathbb{R}^n \to \mathbb{R}$ is defined as $F(x) = \sum_{i \in \mathcal{V}} f_i(x)$. The proposition below shows that F has a unique minimizer $x^* \in \mathbb{R}^n$, so that problem (7.8) is well-posed:

Proposition 7.1. *With Assumption 7.1, there exists a unique $x^* \in \mathbb{R}^n$ such that $F(x^*) \leq F(x)$ $\forall x \in \mathbb{R}^n$ and $\nabla F(x^*) = 0$.*

Proof. By Assumption 7.1, F is twice continuously differentiable and strongly convex with convexity parameter $\sum_{j \in \mathcal{V}} \theta_j > 0$. Pick any $x_o \in \mathbb{R}^n$ and define the set $D = \{x \in \mathbb{R}^n : F(x) \leq F(x_o)\}$. Since $x_o \in D$ and F is continuous, D is nonempty and closed. Pick any $y \in \mathbb{R}^n$ with $\|y\| = 1$ and consider the ray $\{x_o + \eta y \in \mathbb{R}^n : \eta \geq 0\}$. From (7.2), $F(x_o + \eta y) \geq F(x_o) + \eta \nabla F(x_o)^T y + \eta^2 \frac{\sum_{j \in \mathcal{V}} \theta_j}{2} \|y\|^2$. Since $\|y\| = 1$ and $\eta \geq 0$, $F(x_o + \eta y) \geq F(x_o) - \eta \|\nabla F(x_o)\| + \eta^2 \frac{\sum_{j \in \mathcal{V}} \theta_j}{2}$. Therefore, $\forall \eta > \frac{2\|\nabla F(x_o)\|}{\sum_{j \in \mathcal{V}} \theta_j}$, $F(x_o + \eta y) > F(x_o)$, so that $x_o + \eta y \notin D$. Hence, D is bounded and, thus, compact. Since F is continuous, there exists an $x^* \in D$ such that $F(x^*) \leq F(x)$ $\forall x \in D$. By definition of D, $F(x^*) \leq F(x)$ $\forall x \in \mathbb{R}^n$. Because F is strongly convex, x^* is unique and satisfies $\nabla F(x^*) = 0$. $\square$

Given the above network and problem, the aim of this chapter is to devise a continuous-time distributed algorithm of the form

$$\dot{x}_i(t) = \varphi_i(x_i(t), \mathbf{x}_{\mathcal{N}_i}(t); f_i, \mathbf{f}_{\mathcal{N}_i}), \quad \forall t \geq 0,\ \forall i \in \mathcal{V}, \tag{7.9}$$

$$x_i(0) = \chi_i(f_i, \mathbf{f}_{\mathcal{N}_i}), \quad \forall i \in \mathcal{V}, \tag{7.10}$$

where $t \geq 0$ denotes time; $x_i(t) \in \mathbb{R}^n$ is a state representing node i's estimate of the unknown minimizer x^* at time t; $\mathbf{x}_{\mathcal{N}_i}(t) = (x_j(t))_{j \in \mathcal{N}_i} \in \mathbb{R}^{n|\mathcal{N}_i|}$ is a vector obtained by stacking $x_j(t)$ $\forall j \in \mathcal{N}_i$; $\mathbf{f}_{\mathcal{N}_i} = (f_j)_{j\in\mathcal{N}_i} : \mathbb{R}^n \to \mathbb{R}^{|\mathcal{N}_i|}$ is a function obtained by stacking f_j $\forall j \in \mathcal{N}_i$; $\varphi_i : \mathbb{R}^n \times \mathbb{R}^{n|\mathcal{N}_i|} \to \mathbb{R}^n$ is a locally Lipschitz function of $x_i(t)$ and $\mathbf{x}_{\mathcal{N}_i}(t)$ governing the dynamics of $x_i(t)$, whose definition may depend on f_i and $\mathbf{f}_{\mathcal{N}_i}$; $\chi_i \in \mathbb{R}^n$ is a constant determining the initial state $x_i(0)$, whose value may depend on f_i and $\mathbf{f}_{\mathcal{N}_i}$; $|\cdot|$ denotes the cardinality of a set; and $x_i(t)$, f_i, φ_i, and χ_i are maintained in node i's local memory. The goal of the algorithm (7.9) and (7.10) is to steer all the estimates $x_i(t)$'s asymptotically (or, better yet, exponentially) to the unknown x^*, i.e.,

$$\lim_{t\to\infty} x_i(t) = x^*, \quad \forall i \in \mathcal{V}, \tag{7.11}$$

enabling all the nodes to cooperatively solve problem (7.8). Note that to realize (7.9) and (7.10), for each $i \in \mathcal{V}$, every node $j \in \mathcal{N}_i$ must send node i its $x_j(t)$ at each time t if φ_i does depend on $x_j(t)$, and its f_j at time $t = 0$ if φ_i or χ_i does depend on f_j.

7.4 Zero-Gradient-Sum Algorithms

In this section, we develop a family of algorithms that achieve the stated goal. To facilitate the development, we let $\mathbf{x}^* = (x^*, x^*, \ldots, x^*) \in \mathbb{R}^{nN}$ denote the vector of minimizers and $\mathbf{x}(t) = (x_1(t), x_2(t), \ldots, x_N(t)) \in \mathbb{R}^{nN}$, or simply $\mathbf{x} = (x_1, x_2, \ldots, x_N)$, denote the entire state vector.

Consider a Lyapunov function candidate $V : \mathbb{R}^{nN} \to \mathbb{R}$, defined in terms of the observed f_i's as

$$V(\mathbf{x}) = \sum_{i\in\mathcal{V}} f_i(x^*) - f_i(x_i) - \nabla f_i(x_i)^T(x^* - x_i). \tag{7.12}$$

Notice that V in (7.12) is continuously differentiable because of Assumption 7.1, and that it satisfies $V(\mathbf{x}^*) = 0$. Moreover, V is positive definite with respect to

$\mathbf{x}^*$ and is radially unbounded, which can be seen by noting that Assumption 7.1 and the first-order strong convexity condition (7.2) imply

$$V(\mathbf{x}) \geq \sum_{i\in\mathcal{V}} \frac{\theta_i}{2}\|x^* - x_i\|^2, \quad \forall \mathbf{x} \in \mathbb{R}^{nN}, \tag{7.13}$$

and (7.13) in turn implies $V(\mathbf{x}) > 0$ $\forall \mathbf{x} \neq \mathbf{x}^*$ and $V(\mathbf{x}) \to \infty$ as $\|\mathbf{x}\| \to \infty$. Therefore, V in (7.12) is a legitimate Lyapunov function candidate, which may be used to derive algorithms that ensure (7.11).

Taking the time derivative of V along the state trajectory $\mathbf{x}(t)$ of the system (7.9) and calling it $\dot{V} : \mathbb{R}^{nN} \to \mathbb{R}$, we obtain

$$\dot{V}(\mathbf{x}(t)) = \sum_{i\in\mathcal{V}} (x_i(t) - x^*)^T \nabla^2 f_i(x_i(t)) \varphi_i(x_i(t), \mathbf{x}_{\mathcal{N}_i}(t); f_i, \mathbf{f}_{\mathcal{N}_i}), \quad \forall t \geq 0. \tag{7.14}$$

Due to Assumption 7.1 and to each φ_i being locally Lipschitz, $\dot{V}$ in (7.14) is continuous. In addition, it yields $\dot{V}(\mathbf{x}^*) = 0$. Hence, if the functions φ_i $\forall i \in \mathcal{V}$ are such that $\dot{V}$ is negative definite with respect to $\mathbf{x}^*$, i.e.,

$$\sum_{i\in\mathcal{V}} (x_i - x^*)^T \nabla^2 f_i(x_i) \varphi_i(x_i, \mathbf{x}_{\mathcal{N}_i}; f_i, \mathbf{f}_{\mathcal{N}_i}) < 0, \quad \forall \mathbf{x} \neq \mathbf{x}^*, \tag{7.15}$$

the system (7.9) would have a unique equilibrium point at $\mathbf{x}^*$, which by the Barbashin-Krasovskii theorem would be globally asymptotically stable. Consequently, regardless of how the constants χ_i $\forall i \in \mathcal{V}$ in (7.10) are chosen, the goal (7.11) would be accomplished.

As it follows from the above, the challenge lies in finding φ_i $\forall i \in \mathcal{V}$, which collectively satisfy (7.15). Such φ_i's, however, may be difficult to construct because x^* in (7.15) is unknown to any of the nodes, i.e., x^* depends on *every* f_i via (7.8), but φ_i maintained by each node $i \in \mathcal{V}$ can *only* depend on f_i and $\mathbf{f}_{\mathcal{N}_i}$. As a result, one cannot let the φ_i's depend on x^*, such as letting

$\varphi_i(x_i, \mathbf{x}_{\mathcal{N}_i}; f_i, \mathbf{f}_{\mathcal{N}_i}) = x^* - x_i \; \forall i \in \mathcal{V}$, even though this particular choice guarantees (7.15) (since each $\nabla^2 f_i(x_i)$ is positive definite, by (7.4)). Given that the required φ_i's are not readily apparent, instead of searching for them, below we present an alternative approach toward the goal (7.11), which uses the same V and $\dot{V}$ as in (7.12) and (7.14), but demands neither local nor global asymptotic stability.

To state the approach, we first introduce two definitions: let $\mathcal{A} \subset \mathbb{R}^{nN}$ represent the *agreement set* and $\mathcal{M} \subset \mathbb{R}^{nN}$ represent the *zero-gradient-sum manifold*, defined respectively as

$$\mathcal{A} = \{(y_1, y_2, \ldots, y_N) \in \mathbb{R}^{nN} : y_1 = y_2 = \cdots = y_N\}, \tag{7.16}$$

$$\mathcal{M} = \{(y_1, y_2, \ldots, y_N) \in \mathbb{R}^{nN} : \sum_{i \in \mathcal{V}} \nabla f_i(y_i) = 0\}, \tag{7.17}$$

so that $\mathbf{x} \in \mathcal{A}$ if and only if all the x_i's agree, and $\mathbf{x} \in \mathcal{M}$ if and only if the sum of all the gradients ∇f_i's, evaluated respectively at the x_i's, is zero. Notice from (7.16) that $\mathbf{x}^* \in \mathcal{A}$, from (7.17) and Proposition 7.1 that $\mathbf{x}^* \in \mathcal{M}$, and from all of them that $\mathbf{x} \in \mathcal{A} \cap \mathcal{M} \Rightarrow \mathbf{x} = \mathbf{x}^*$. Thus, $\mathcal{A} \cap \mathcal{M} = \{\mathbf{x}^*\}$. Also note from the continuity of each ∇f_i that $\mathcal{M}$ is closed and from the Implicit Function Theorem and the nonsingularity of each $\nabla^2 f_i(x) \; \forall x \in \mathbb{R}^n$ that $\mathcal{M}$ is indeed a manifold of dimension $n(N-1)$.

Having introduced $\mathcal{A}$ and $\mathcal{M}$, we now describe the approach, which is based on the following recognition: to attain the goal (7.11), condition (7.15)—which ensures that *every* trajectory $\mathbf{x}(t)$ goes to $\mathbf{x}^*$—is sufficient but not necessary. Rather, all that is needed is a *single* trajectory $\mathbf{x}(t)$, along which $\dot{V}(\mathbf{x}(t)) \leq 0 \; \forall t \geq 0$ and $\lim_{t\to\infty} V(\mathbf{x}(t)) = 0$, since the latter implies (7.11). Recognizing this, we next derive three conditions on the φ_i's and χ_i's in (7.9) and (7.10) that produce such a trajectory. Assume, for a moment, that the

χ_i's dictating the initial state $\mathbf{x}(0)$ have been decided, so that we may focus on the φ_i's that shape the trajectory $\mathbf{x}(t)$ leaving $\mathbf{x}(0)$. Observe that $\dot{V}$ in (7.14) takes the form $\dot{V}(\mathbf{x}(t)) = \Phi_1(\mathbf{x}(t)) - x^{*T}\Phi_2(\mathbf{x}(t))$ $\forall t \geq 0$, where $\Phi_1 : \mathbb{R}^{nN} \to \mathbb{R}$ and $\Phi_2 : \mathbb{R}^{nN} \to \mathbb{R}^n$. Thus, the unknown x^*—which may undesirably affect the sign of $\dot{V}(\mathbf{x}(t))$—can be eliminated by setting $\Phi_2(\mathbf{x}) = 0$ $\forall \mathbf{x} \in \mathbb{R}^{nN}$, i.e., by forcing the φ_i's to satisfy

$$\sum_{i\in\mathcal{V}} \nabla^2 f_i(x_i)\varphi_i(x_i, \mathbf{x}_{\mathcal{N}_i}; f_i, \mathbf{f}_{\mathcal{N}_i}) = 0, \quad \forall \mathbf{x} \in \mathbb{R}^{nN}. \tag{7.18}$$

With this first condition (7.18), $\dot{V}$ becomes free of x^*, reducing to

$$\dot{V}(\mathbf{x}(t)) = \sum_{i\in\mathcal{V}} x_i(t)^T \nabla^2 f_i(x_i(t))\varphi_i(x_i(t), \mathbf{x}_{\mathcal{N}_i}(t); f_i, \mathbf{f}_{\mathcal{N}_i}), \quad \forall t \geq 0. \tag{7.19}$$

Next, notice that whenever $\mathbf{x}(t)$ is in the agreement set $\mathcal{A}$, due to (7.16) and (7.18), $\dot{V}(\mathbf{x}(t))$ in (7.19) must vanish. However, whenever $\mathbf{x}(t) \notin \mathcal{A}$, there is no such restriction. Hence, any time $\mathbf{x}(t) \notin \mathcal{A}$, $\dot{V}(\mathbf{x}(t))$ can be made negative by forcing the φ_i's to also satisfy

$$\sum_{i\in\mathcal{V}} x_i^T \nabla^2 f_i(x_i)\varphi_i(x_i, \mathbf{x}_{\mathcal{N}_i}; f_i, \mathbf{f}_{\mathcal{N}_i}) < 0, \quad \forall \mathbf{x} \in \mathbb{R}^{nN} - \mathcal{A}. \tag{7.20}$$

With this additional, second condition (7.20), no matter what x^* is, $\dot{V}(\mathbf{x}(t)) \leq 0$ along $\mathbf{x}(t)$, with equality if and only if $\mathbf{x}(t) \in \mathcal{A}$. Finally, note that (7.18) and (7.9) imply

$$\frac{d}{dt}\sum_{i\in\mathcal{V}} \nabla f_i(x_i(t)) = \sum_{i\in\mathcal{V}} \nabla^2 f_i(x_i(t))\dot{x}_i(t) = 0, \quad \forall t \geq 0,$$

while (7.11), the continuity of each ∇f_i, and Proposition 7.1 imply

$$\lim_{t\to\infty}\sum_{i\in\mathcal{V}} \nabla f_i(x_i(t)) = \sum_{i\in\mathcal{V}} \nabla f_i(\lim_{t\to\infty} x_i(t)) = \sum_{i\in\mathcal{V}} \nabla f_i(x^*) = \nabla F(x^*) = 0.$$

The former says that by making the φ_i's satisfy (7.18), the gradient sum $\sum_{i\in\mathcal{V}} \nabla f_i(x_i(t))$ along $\mathbf{x}(t)$ would remain constant over time, while the latter says that to achieve $\lim_{t\to\infty} V(\mathbf{x}(t)) = 0$ or equivalently (7.11), this constant sum must be zero, i.e., $\sum_{i\in\mathcal{V}} \nabla f_i(x_i(t)) = 0\ \forall t \geq 0$. Therefore, in view of (7.10), the χ_i's must be such that

$$\sum_{i\in\mathcal{V}} \nabla f_i(\chi_i(f_i, \mathbf{f}_{\mathcal{N}_i})) = 0, \tag{7.21}$$

yielding the third and final condition.

By imposing algebraic constraints on the φ_i's and χ_i's, conditions (7.18), (7.20), and (7.21) characterize a family of algorithms. This family of algorithms share a number of properties, including one that has a nice geometric interpretation: observe from (7.21), (7.10), and (7.17) that $\mathbf{x}(0) \in \mathcal{M}$ and further from (7.18) and (7.9) that $\mathbf{x}(t) \in \mathcal{M}\ \forall t > 0$. Thus, every algorithm in the family produces a nonlinear networked dynamical system, whose trajectory $\mathbf{x}(t)$ begins on, and slides along, the zero-gradient-sum manifold $\mathcal{M}$, making $\mathcal{M}$ a positively invariant set. Due to this geometric interpretation, these algorithms are referred to as follows:

Definition 7.1. A continuous-time distributed algorithm of the form (7.9) and (7.10) is said to be a *Zero-Gradient-Sum* (ZGS) algorithm if $\varphi_i\ \forall i \in \mathcal{V}$ are locally Lipschitz and satisfy (7.18) and (7.20), and $\chi_i\ \forall i \in \mathcal{V}$ satisfy (7.21).

The following theorem lists the properties shared by ZGS algorithms, showing that every one of them is capable of asymptotically driving $\mathbf{x}(t)$ to $\mathbf{x}^*$, solving problem (7.8):

Theorem 7.1. *Consider the network modeled in Section 7.3 and the use of a ZGS algorithm described in Definition 7.1. Suppose Assumption 7.1 holds.*

Then: (i) there exists a unique solution $\mathbf{x}(t)$ $\forall t \geq 0$ *to* (7.9) *and* (7.10)*; (ii)* $\mathbf{x}(t) \in \mathcal{M}$ $\forall t \geq 0$*; (iii)* $\dot{V}(\mathbf{x}(t)) \leq 0$ $\forall t \geq 0$*, with equality if and only if* $\mathbf{x}(t) = \mathbf{x}^*$*; (iv)* $\lim_{t\to\infty} V(\mathbf{x}(t)) = 0$*; and (v)* $\lim_{t\to\infty} \mathbf{x}(t) = \mathbf{x}^*$*, i.e.,* (7.11) *holds.*

Proof. Since φ_i $\forall i \in \mathcal{V}$ are locally Lipschitz, to prove (i) it suffices to show that every solution $\mathbf{x}(t)$ of (7.9) and (7.10) lies entirely in a compact subset of $\mathbb{R}^{nN}$. To this end, let $\mathcal{B}(\mathbf{x}^*, r) \subset \mathbb{R}^{nN}$ denote the closed-ball of radius $r \in [0, \infty)$ centered at $\mathbf{x}^*$, i.e., $\mathcal{B}(\mathbf{x}^*, r) = \{\mathbf{y} \in \mathbb{R}^{nN} : \|\mathbf{y} - \mathbf{x}^*\| \leq r\}$. Note from (7.14), (7.18), and (7.20) that $\dot{V}(\mathbf{x}(t)) \leq 0$ along $\mathbf{x}(t)$. This, together with (7.13), implies that $V(\mathbf{x}(0)) \geq V(\mathbf{x}(t)) \geq \frac{\min_{i\in\mathcal{V}} \theta_i}{2}\|\mathbf{x}(t) - \mathbf{x}^*\|^2$ along $\mathbf{x}(t)$. Hence, $\mathbf{x}(t) \in \mathcal{B}(\mathbf{x}^*, \sqrt{\frac{2V(\mathbf{x}(0))}{\min_{i\in\mathcal{V}} \theta_i}})$ $\forall t \geq 0$, ensuring (i). Statement (ii) has been proven in the paragraph before Definition 7.1. To verify (iii), notice again from (7.14), (7.18), and (7.20) that $\dot{V}(\mathbf{x}(t)) = 0$ if and only if $\mathbf{x}(t) \in \mathcal{A}$. Due to (ii) and to $\mathcal{A} \cap \mathcal{M} = \{\mathbf{x}^*\}$ shown earlier, (iii) holds. To prove (iv), observe from (7.13) and (iii) that $V(\mathbf{x}(t))$ $\forall t \geq 0$ is nonnegative and non-increasing. Thus, there exists a $c \geq 0$ such that $\lim_{t\to\infty} V(\mathbf{x}(t)) = c$ and $V(\mathbf{x}(t)) \geq c$ $\forall t \geq 0$. To show that $c = 0$, assume to the contrary that $c > 0$. Then, because V in (7.12) is continuous and positive definite with respect to $\mathbf{x}^*$, there exists an $\epsilon > 0$ such that $\mathcal{B}(\mathbf{x}^*, \epsilon) \subset \{\mathbf{y} \in \mathbb{R}^{nN} : V(\mathbf{y}) < c\}$. With this ϵ, define a set $\mathcal{K} \subset \mathbb{R}^{nN}$ as $\mathcal{K} = \mathcal{M} \cap \{\mathbf{y} \in \mathbb{R}^{nN} : \epsilon \leq \|\mathbf{y} - \mathbf{x}^*\| \leq \sqrt{\frac{2V(\mathbf{x}(0))}{\min_{i\in\mathcal{V}} \theta_i}}\}$. Notice that $\mathbf{x}(t) \in \mathcal{K}$ $\forall t \geq 0$ because $\mathbf{x}(t) \in \mathcal{M}$, $V(\mathbf{x}(t)) \geq c$, and $\mathbf{x}(t) \in \mathcal{B}(\mathbf{x}^*, \sqrt{\frac{2V(\mathbf{x}(0))}{\min_{i\in\mathcal{V}} \theta_i}})$ $\forall t \geq 0$. Also note that $\mathcal{K} \subset \mathcal{M}$ but $\mathcal{K} \not\ni \mathbf{x}^*$. This, along with the properties $\mathcal{A} \cap \mathcal{M} = \{\mathbf{x}^*\}$ and $\dot{V}(\mathbf{y}) < 0$ $\forall \mathbf{y} \notin \mathcal{A}$, implies that $\dot{V}(\mathbf{y}) < 0$ $\forall \mathbf{y} \in \mathcal{K}$. Since $\dot{V}$ in (7.14) is continuous and $\mathcal{K}$ is nonempty and compact (due to $\mathcal{M}$ being a closed set), there exists an $\eta > 0$ such that $\max_{\mathbf{y}\in\mathcal{K}} \dot{V}(\mathbf{y}) = -\eta$. Since $\mathbf{x}(t) \in \mathcal{K}$ $\forall t \geq 0$,

$V(\mathbf{x}(t)) = V(\mathbf{x}(0)) + \int_0^t \dot{V}(\mathbf{x}(\tau))\,d\tau \leq V(\mathbf{x}(0)) - \eta t$. This implies $V(\mathbf{x}(t)) < c$ $\forall t > \frac{V(\mathbf{x}(0))-c}{\eta}$, which is a contradiction. Therefore, $c = 0$, establishing (iv). Finally, (v) is an immediate consequence of (7.13) and (iv). $\square$

Having established Theorem 7.1, we now present a systematic way to construct ZGS algorithms. First, to find χ_i's that meet condition (7.21), consider the following proposition, which shows that each f_i has a unique minimizer $x_i^* \in \mathbb{R}^n$:

Proposition 7.2. *With Assumption 7.1, for each $i \in \mathcal{V}$, there exists a unique $x_i^* \in \mathbb{R}^n$ such that $f_i(x_i^*) \leq f_i(x)$ $\forall x \in \mathbb{R}^n$ and $\nabla f_i(x_i^*) = 0$.*

Proof. For each $i \in \mathcal{V}$, the proof is identical to that of Proposition 7.1 with x^*, F, and $\sum_{j\in\mathcal{V}} \theta_j$ replaced by x_i^*, f_i, and θ_i, respectively. $\square$

Proposition 7.2 implies that $\sum_{i\in\mathcal{V}} \nabla f_i(x_i^*) = 0$. Hence, (7.21) can be met by simply letting

$$\chi_i(f_i, \mathbf{f}_{\mathcal{N}_i}) = x_i^*, \quad \forall i \in \mathcal{V}, \tag{7.22}$$

which is permissible since every x_i^* in (7.22) depends just on f_i. It follows that each node $i \in \mathcal{V}$ must solve a "local" convex optimization problem $\min_{x\in\mathbb{R}^n} f_i(x)$ for x_i^* before time $t = 0$, in order to execute (7.10) and (7.22).

Next, to generate locally Lipschitz φ_i's that ensure conditions (7.18) and (7.20), notice that each φ_i is premultiplied by $\nabla^2 f_i(x_i)$, which is nonsingular $\forall x_i \in \mathbb{R}^n$. Therefore, the impact of each $\nabla^2 f_i(x_i)$ can be absorbed by setting

$$\varphi_i(x_i, \mathbf{x}_{\mathcal{N}_i}; f_i, \mathbf{f}_{\mathcal{N}_i}) = (\nabla^2 f_i(x_i))^{-1}\phi_i(x_i, \mathbf{x}_{\mathcal{N}_i}; f_i, \mathbf{f}_{\mathcal{N}_i}), \quad \forall i \in \mathcal{V}, \tag{7.23}$$

where $\phi_i : \mathbb{R}^n \times \mathbb{R}^{n|\mathcal{N}_i|} \to \mathbb{R}^n$ is a locally Lipschitz function of x_i and $\mathbf{x}_{\mathcal{N}_i}$ maintained by node i. For each $i \in \mathcal{V}$, because $\nabla^2 f_i$ is locally Lipschitz (due to Assumption 7.1) and the determinant of $\nabla^2 f_i(x_i)$ for every $x_i \in \mathbb{R}^n$ is no less than a positive constant θ_i^n (due further to (7.4)), the mapping $(\nabla^2 f_i(\cdot))^{-1} : \mathbb{R}^n \to \mathbb{R}^{n\times n}$ in (7.23) is locally Lipschitz. Thus, as long as the ϕ_i's are locally Lipschitz, so would the resulting φ_i's, fulfilling the requirement. With (7.23), the dynamics (7.9) become

$$\dot{x}_i(t) = (\nabla^2 f_i(x_i(t)))^{-1}\phi_i(x_i(t), \mathbf{x}_{\mathcal{N}_i}(t); f_i, \mathbf{f}_{\mathcal{N}_i}), \quad \forall t \geq 0,\ \forall i \in \mathcal{V}, \tag{7.24}$$

and conditions (7.18) and (7.20) simplify to

$$\sum_{i\in\mathcal{V}} \phi_i(x_i, \mathbf{x}_{\mathcal{N}_i}; f_i, \mathbf{f}_{\mathcal{N}_i}) = 0, \quad \forall \mathbf{x} \in \mathbb{R}^{nN}, \tag{7.25}$$

$$\sum_{i\in\mathcal{V}} x_i^T \phi_i(x_i, \mathbf{x}_{\mathcal{N}_i}; f_i, \mathbf{f}_{\mathcal{N}_i}) < 0, \quad \forall \mathbf{x} \in \mathbb{R}^{nN} - \mathcal{A}. \tag{7.26}$$

Finally, to come up with locally Lipschitz ϕ_i's that assure conditions (7.25) and (7.26), suppose each ϕ_i is decomposed as

$$\phi_i(x_i, \mathbf{x}_{\mathcal{N}_i}; f_i, \mathbf{f}_{\mathcal{N}_i}) = \sum_{j\in\mathcal{N}_i} \phi_{ij}(x_i, x_j; f_i, f_j), \quad \forall i \in \mathcal{V}, \tag{7.27}$$

so that the dynamics (7.24) become

$$\dot{x}_i(t) = (\nabla^2 f_i(x_i(t)))^{-1} \sum_{j\in\mathcal{N}_i} \phi_{ij}(x_i(t), x_j(t); f_i, f_j), \quad \forall t \geq 0,\ \forall i \in \mathcal{V}, \tag{7.28}$$

where $\phi_{ij} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ is a locally Lipschitz function of x_i and x_j maintained by node i. Then, (7.25) can be ensured by requiring that every ϕ_{ij} and ϕ_{ji} pair be negative of each other, i.e.,

$$\phi_{ij}(y, z; f_i, f_j) = -\phi_{ji}(z, y; f_j, f_i), \quad \forall i \in \mathcal{V},\ \forall j \in \mathcal{N}_i,\ \forall y, z \in \mathbb{R}^n, \tag{7.29}$$

since $\sum_{i\in\mathcal{V}}\phi_i = \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\phi_{ij} = \sum_{\{i,j\}\in\mathcal{E}}\phi_{ij}+\phi_{ji} = 0$. With (7.27) and (7.29), the left-hand side of (7.26) turns into

$$\sum_{i\in\mathcal{V}} x_i^T\phi_i(x_i,\mathbf{x}_{\mathcal{N}_i};f_i,\mathbf{f}_{\mathcal{N}_i}) = \frac{1}{2}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}(x_i-x_j)^T\phi_{ij}(x_i,x_j;f_i,f_j), \quad \forall\mathbf{x}\in\mathbb{R}^{nN}. \tag{7.30}$$

Because the graph $\mathcal{G}$ is connected, for any $\mathbf{x}\in\mathbb{R}^{nN}-\mathcal{A}$, there exist $i\in\mathcal{V}$ and $j\in\mathcal{N}_i$ such that x_i-x_j in (7.30) is nonzero. Hence, (7.26) can be guaranteed by requiring the ϕ_{ij}'s to also satisfy

$$(y-z)^T\phi_{ij}(y,z;f_i,f_j)<0, \quad \forall i\in\mathcal{V},\ \forall j\in\mathcal{N}_i,\ \forall y,z\in\mathbb{R}^n,\ y\neq z. \tag{7.31}$$

Note that if (7.29) holds, then ϕ_{ij} satisfies the inequality in (7.31) if and only if ϕ_{ji} does. Therefore, every pair of neighboring nodes $i,j\in\mathcal{V}$ need only minimal coordination before time $t=0$ to realize the dynamics (7.28): only one of them, say, node i, needs to construct a ϕ_{ij} that satisfies the inequality in (7.31), and the other, i.e., node j, only needs to make sure that $\phi_{ji}=-\phi_{ij}$.

Examples 7.1 and 7.2 below illustrate two concrete ways to construct ϕ_{ij}'s that obey (7.29) and (7.31):

Example 7.1. Let $\phi_{ij}(y,z;f_i,f_j)=(\psi_{ij1}(y_1,z_1),\psi_{ij2}(y_2,z_2),\dots,\psi_{ijn}(y_n,z_n))$ $\forall i\in\mathcal{V}$ $\forall j\in\mathcal{N}_i$ $\forall y=(y_1,y_2,\dots,y_n)\in\mathbb{R}^n$ $\forall z=(z_1,z_2,\dots,z_n)\in\mathbb{R}^n$, where each $\psi_{ij\ell}:\mathbb{R}^2\to\mathbb{R}$ can be any locally Lipschitz function satisfying $\psi_{ij\ell}(y_\ell,z_\ell)=-\psi_{ji\ell}(z_\ell,y_\ell)$ and $(y_\ell-z_\ell)\psi_{ij\ell}(y_\ell,z_\ell)<0$ $\forall y_\ell\neq z_\ell$ (e.g., $\psi_{ij\ell}(y_\ell,z_\ell)=\tanh(z_\ell-y_\ell)$ or $\psi_{ij\ell}(y_\ell,z_\ell)=-\psi_{ji\ell}(z_\ell,y_\ell)=\frac{z_\ell-y_\ell}{1+y_\ell^2}$). Then, (7.29) and (7.31) hold. ■

Example 7.2. Let $\phi_{ij}(y,z;f_i,f_j)=\nabla g_{\{i,j\}}(z)-\nabla g_{\{i,j\}}(y)$ $\forall i\in\mathcal{V}$ $\forall j\in\mathcal{N}_i$ $\forall y,z\in\mathbb{R}^n$, where each $g_{\{i,j\}}:\mathbb{R}^n\to\mathbb{R}$ can be any twice continuously differentiable and locally strongly convex function associated with link $\{i,j\}\in\mathcal{E}$ (e.g., $g_{\{i,j\}}(y)=\frac{1}{2}y^TA_{\{i,j\}}y$, where $A_{\{i,j\}}\in\mathbb{R}^{n\times n}$ is any symmetric positive definite matrix, or $g_{\{i,j\}}(y)=f_i(y)+f_j(y)$). Then, (7.29) and (7.31) hold. ■

Examples 7.3 and 7.4 below show that some of the continuous-time distributed consensus algorithms in the literature are special cases of ZGS algorithms. In addition, they are just a slight modification away from solving general unconstrained, separable, convex optimization problems:

Example 7.3. Consider the scalar (i.e., $n = 1$) linear consensus algorithm $\dot{x}_i(t) = \sum_{j\in\mathcal{N}_i} a_{ij}(x_j(t) - x_i(t))\ \forall t \geq 0\ \forall i \in \mathcal{V}$ with symmetric parameters $a_{ij} = a_{ji} > 0\ \forall\{i,j\} \in \mathcal{E}$ and arbitrary initial states $x_i(0) = y_i\ \forall i \in \mathcal{V}$, studied in [24, 52, 63, 69]. By Definition 7.1 and Theorem 7.1, this algorithm is a ZGS algorithm that solves problem (7.8) for $f_i(x) = \frac{1}{2}(x - y_i)^2\ \forall i \in \mathcal{V}$. Moreover, the algorithm is only a Hessian inverse and an initial condition away (i.e., $\dot{x}_i(t) = (\nabla^2 f_i(x_i(t)))^{-1} \sum_{j\in\mathcal{N}_i} a_{ij}(x_j(t) - x_i(t))$ with $x_i(0) = x_i^*$) from solving *any* convex optimization problem of the form (7.8) for any $n \geq 1$. Note that the same can be said about the scalar nonlinear consensus protocol in [51]. ■

Example 7.4. Consider the multivariable (i.e., $n \geq 1$) weighted-average consensus algorithm $\dot{x}_i(t) = W_i^{-1} \sum_{j\in\mathcal{N}_i}(x_j(t) - x_i(t))\ \forall t \geq 0\ \forall i \in \mathcal{V}$ with $W_i = W_i^T > 0$ and $x_i(0) = y_i$, proposed in [66] as a step toward a distributed Kalman filter. This algorithm is a ZGS algorithm that solves problem (7.8) for $f_i(x) = \frac{1}{2}(x - y_i)^T W_i (x - y_i)\ \forall i \in \mathcal{V}$. Indeed, it came close to solving for general f_i's. ■

7.5 Convergence Rate Analysis

In this section, we derive lower and upper bounds on the exponential convergence rates of the ZGS algorithms described in (7.28) and Example 7.2, i.e.,

$$\dot{x}_i(t) = (\nabla^2 f_i(x_i(t)))^{-1} \sum_{j\in\mathcal{N}_i} \nabla g_{\{i,j\}}(x_j(t)) - \nabla g_{\{i,j\}}(x_i(t)), \quad \forall t \geq 0,\ \forall i \in \mathcal{V}, \tag{7.32}$$

which form a subset of those in Definition 7.1, but include the ones in Examples 7.3 and 7.4 as a subset. To enable the derivation, suppose an initial state $\mathbf{x}(0) \in \mathcal{M}$ is given (e.g., $\mathbf{x}(0) = (x_1^*, x_2^*, \dots, x_N^*)$ as in (7.10) and (7.22)). With this $\mathbf{x}(0)$, let $\mathcal{C}_i = \{x \in \mathbb{R}^n : f_i(x^*) - f_i(x) - \nabla f_i(x)^T(x^* - x) \leq V(\mathbf{x}(0))\}$ $\forall i \in \mathcal{V}$ and let $\mathcal{C} = \operatorname{conv} \cup_{i\in\mathcal{V}} \mathcal{C}_i$, where conv denotes the convex hull. It follows from Assumption 7.1, (7.2), (7.12), and (iii) in Theorem 7.1 that $\mathcal{C}_i$ $\forall i \in \mathcal{V}$ are compact, $\mathcal{C}$ is convex and compact, and

$$x_i(t), x^* \in \mathcal{C}_i \subset \mathcal{C}, \quad \forall t \geq 0,\ \forall i \in \mathcal{V}. \tag{7.33}$$

For each $i \in \mathcal{V}$, due to Assumption 7.1, (7.4), and $\mathcal{C}$ being compact, there exists a $\Theta_i \geq \theta_i$ such that

$$\nabla^2 f_i(x) \leq \Theta_i I_n, \quad \forall x \in \mathcal{C}. \tag{7.34}$$

Moreover, for each $\{i,j\} \in \mathcal{E}$, due to (7.3), $g_{\{i,j\}}$ being locally strongly convex, and $\mathcal{C}$ being convex and compact, there exists a $\gamma_{\{i,j\}} > 0$ such that

$$(\nabla g_{\{i,j\}}(y) - \nabla g_{\{i,j\}}(x))^T(y - x) \geq \gamma_{\{i,j\}}\|y - x\|^2, \quad \forall x, y \in \mathcal{C}. \tag{7.35}$$

Furthermore, for each $\{i,j\} \in \mathcal{E}$, due to (7.3), (7.4), (7.35), $\nabla^2 g_{\{i,j\}}$ being continuous, and $\mathcal{C}$ being convex and compact, there exists a $\Gamma_{\{i,j\}} \geq \gamma_{\{i,j\}}$ such that

$$\nabla^2 g_{\{i,j\}}(x) \leq \Gamma_{\{i,j\}} I_n, \quad \forall x \in \mathcal{C}. \tag{7.36}$$

Observe that the constants Θ_i's, $\gamma_{\{i,j\}}$'s, and $\Gamma_{\{i,j\}}$'s—unlike the convexity parameters θ_i's—depend on the initial state $\mathbf{x}(0)$ via the sets $\mathcal{C}$ and $\mathcal{C}_i$'s. Thus, the convergence rate results obtained below are dependent on $\mathbf{x}(0)$ in general. One exception is the case where the f_i's and $g_{\{i,j\}}$'s are quadratic functions, for which the θ_i's, Θ_i's, $\gamma_{\{i,j\}}$'s, and $\Gamma_{\{i,j\}}$'s may be taken as the smallest and largest eigenvalues of the Hessians of the f_i's and $g_{\{i,j\}}$'s, respectively, independent of $\mathbf{x}(0)$. Finally, for convenience, let $\theta = \min_{i\in\mathcal{V}} \theta_i$, $\Theta = \max_{i\in\mathcal{V}} \Theta_i$, $\gamma = \min_{\{i,j\}\in\mathcal{E}} \gamma_{\{i,j\}}$, and $\Gamma = \max_{\{i,j\}\in\mathcal{E}} \Gamma_{\{i,j\}}$.

The following theorem establishes the exponential convergence of the ZGS algorithms (7.32) and provides a lower bound ρ on their convergence rates, that they can do no worse than:

Theorem 7.2. *Consider the network modeled in Section 7.3 and the use of a ZGS algorithm described in* (7.32)*. Suppose Assumption 7.1 holds. Then,*

$$V(\mathbf{x}(t)) \le V(\mathbf{x}(0))e^{-\rho t}, \quad \forall t \ge 0, \tag{7.37}$$

$$\sum_{i\in\mathcal{V}} \theta_i \|x_i(t) - x^*\|^2 \le \sum_{i\in\mathcal{V}} \Theta_i \|x_i(0) - x^*\|^2 e^{-\rho t}, \quad \forall t \ge 0, \tag{7.38}$$

where $\rho = \sup\{\varepsilon \in \mathbb{R} : \varepsilon P \le Q\} > 0$*,* $P = [P_{ij}] \in \mathbb{R}^{N\times N}$ *is a positive semidefinite matrix given by*

$$P_{ij} = \begin{cases} (\frac{1}{2} - \frac{1}{N})\Theta_i + \frac{1}{2N^2}\sum_{\ell\in\mathcal{V}} \Theta_\ell, & \text{if } i = j, \\ -\frac{\Theta_i+\Theta_j}{2N} + \frac{1}{2N^2}\sum_{\ell\in\mathcal{V}} \Theta_\ell, & \text{otherwise,} \end{cases} \tag{7.39}$$

and $Q = [Q_{ij}] \in \mathbb{R}^{N\times N}$ *is a positive semidefinite matrix given by*

$$Q_{ij} = \begin{cases} \sum_{\ell\in\mathcal{N}_i} \gamma_{\{i,\ell\}}, & \text{if } i = j, \\ -\gamma_{\{i,j\}}, & \text{if } \{i,j\} \in \mathcal{E}, \\ 0, & \text{otherwise.} \end{cases} \tag{7.40}$$

Proof. Let $\eta(t) = \frac{1}{N}\sum_{j\in\mathcal{V}} x_j(t)$ $\forall t \ge 0$. Due to (7.33) and the convexity of $\mathcal{C}$, $\eta(t) \in \mathcal{C}$. Moreover, by Proposition 7.1, $\sum_{i\in\mathcal{V}} f_i(x^*) = F(x^*) \le F(\eta(t)) = \sum_{i\in\mathcal{V}} f_i(\eta(t))$. Observe from (7.17) and (ii) in Theorem 7.1 that

$\sum_{i\in\mathcal{V}} \nabla f_i(x_i(t)) = 0$. Thus, from (7.12), $V(\mathbf{x}(t)) \le \sum_{i\in\mathcal{V}} f_i(\eta(t)) - f_i(x_i(t)) - \nabla f_i(x_i(t))^T(\eta(t) - x_i(t))$. It follows from (7.5), (7.7), (7.34), (7.33), and (7.39) that

$$V(\mathbf{x}(t)) \le \sum_{i\in\mathcal{V}} \frac{\Theta_i}{2}\|x_i(t) - \frac{1}{N}\sum_{j\in\mathcal{V}} x_j(t)\|^2 = \mathbf{x}(t)^T(P\otimes I_n)\mathbf{x}(t), \quad \forall t \ge 0, \tag{7.41}$$

where $\otimes$ denotes the Kronecker product. Next, using (7.32), (7.9), and (7.19), we can write

$$\dot{V}(\mathbf{x}(t)) = -\frac{1}{2}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}(x_j(t) - x_i(t))^T(\nabla g_{\{i,j\}}(x_j(t)) - \nabla g_{\{i,j\}}(x_i(t))), \quad \forall t \ge 0. \tag{7.42}$$

Therefore, from (7.35), (7.33), and (7.40),

$$-\dot{V}(\mathbf{x}(t)) \ge \frac{1}{2}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\gamma_{\{i,j\}}\|x_j(t) - x_i(t)\|^2 = \mathbf{x}(t)^T(Q\otimes I_n)\mathbf{x}(t), \quad \forall t \ge 0. \tag{7.43}$$

To relate (7.41) and (7.43), notice from (7.39) and (7.40) that both P and Q are symmetric with zero row sums. Also, $\forall y = (y_1, y_2, \ldots, y_N) \in \mathbb{R}^N$, $y^TPy = \sum_{i\in\mathcal{V}} \frac{\Theta_i}{2}(y_i - \frac{1}{N}\sum_{j\in\mathcal{V}} y_j)^2 \ge 0$ and $y^TQy = \frac{1}{2}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\gamma_{\{i,j\}}(y_j - y_i)^2 \ge 0$, where the equalities hold if and only if $y_1 = y_2 = \cdots = y_N$. Hence, both P and Q are positive semidefinite with $N-1$ positive eigenvalues, one eigenvalue at 0, and $(\frac{1}{\sqrt{N}}, \frac{1}{\sqrt{N}}, \ldots, \frac{1}{\sqrt{N}})$ being its corresponding eigenvector. It follows that there exists an orthogonal $W \in \mathbb{R}^{N\times N}$ with the first column being $(\frac{1}{\sqrt{N}}, \frac{1}{\sqrt{N}}, \ldots, \frac{1}{\sqrt{N}})$, such that $W^TPW = \operatorname{diag}(0, \bar{P})$ and $W^TQW = \operatorname{diag}(0, \bar{Q})$, where $\bar{P}, \bar{Q} \in \mathbb{R}^{(N-1)\times(N-1)}$, $\bar{P} = \bar{P}^T > 0$, and $\bar{Q} = \bar{Q}^T > 0$. Note that $\forall \varepsilon \in \mathbb{R}$, $\varepsilon P \le Q \Leftrightarrow \varepsilon\bar{P} \le \bar{Q} \Leftrightarrow \varepsilon I_{N-1} \le \bar{P}^{-1/2}\bar{Q}\bar{P}^{-1/2}$, where $\bar{P}^{1/2} = (\bar{P}^{1/2})^T > 0$ is the square root of $\bar{P}$ via the spectral decomposition, i.e., $\bar{P} = \bar{P}^{1/2}\bar{P}^{1/2}$. Since $\rho = \sup\{\varepsilon \in \mathbb{R} : \varepsilon P \le Q\}$ and $\bar{P}^{-1/2}\bar{Q}\bar{P}^{-1/2} = (\bar{P}^{-1/2}\bar{Q}\bar{P}^{-1/2})^T > 0$, ρ is the smallest eigenvalue of $\bar{P}^{-1/2}\bar{Q}\bar{P}^{-1/2}$ which is positive and satisfies $\rho P \leq Q$. Therefore, $\rho(P \otimes I_n) \leq Q \otimes I_n$. This, along with (7.41) and (7.43), implies $\rho V(\mathbf{x}(t)) \leq -\dot{V}(\mathbf{x}(t))$, i.e., (7.37). Finally, due to (7.2), (7.12), (7.37), (7.34), (7.7), (7.5), and (7.33), $\sum_{i\in\mathcal{V}} \frac{\theta_i}{2}\|x_i(t) - x^*\|^2 \leq V(\mathbf{x}(t)) \leq V(\mathbf{x}(0))e^{-\rho t} \leq \sum_{i\in\mathcal{V}} \frac{\Theta_i}{2}\|x_i(0) - x^*\|^2 e^{-\rho t}$, i.e., (7.38) holds. □

The lower bound ρ in Theorem 7.2 can be calculated according to its proof: ρ is the smallest eigenvalue of $\bar{P}^{-1/2}\bar{Q}\bar{P}^{-1/2}$. The corollary below gives another lower bound, which is not as tight as ρ but is explicit in the algebraic connectivity $\lambda_2 > 0$ of the graph $\mathcal{G}$:

Corollary 7.1. *With the setup of Theorem 7.2,*

$$V(\mathbf{x}(t)) \leq V(\mathbf{x}(0))e^{-\frac{2\gamma}{\Theta}\lambda_2 t}, \quad \forall t \geq 0, \tag{7.44}$$

$$\|\mathbf{x}(t) - \mathbf{x}^*\| \leq \sqrt{\frac{\Theta}{\theta}}\|\mathbf{x}(0) - \mathbf{x}^*\|e^{-\frac{\gamma}{\Theta}\lambda_2 t}, \quad \forall t \geq 0. \tag{7.45}$$

Proof. From (7.41) and (7.43),

$$V(\mathbf{x}(t)) \leq \frac{\Theta}{2}\sum_{i\in\mathcal{V}}\|x_i(t) - \frac{1}{N}\sum_{j\in\mathcal{V}} x_j(t)\|^2 = \frac{\Theta}{2N}\mathbf{x}(t)^T(\mathcal{L}_{\bar{\mathcal{G}}} \otimes I_n)\mathbf{x}(t), \quad \forall t \geq 0, \tag{7.46}$$

$$-\dot{V}(\mathbf{x}(t)) \geq \frac{\gamma}{2}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i}\|x_j(t) - x_i(t)\|^2 = \gamma\mathbf{x}(t)^T(\mathcal{L}_{\mathcal{G}} \otimes I_n)\mathbf{x}(t), \quad \forall t \geq 0, \tag{7.47}$$

where $\mathcal{L}_{\bar{\mathcal{G}}} \in \mathbb{R}^{N\times N}$ is the Laplacian of the complete graph $\bar{\mathcal{G}}$ with vertex set $\mathcal{V}$, and $\mathcal{L}_{\mathcal{G}} \in \mathbb{R}^{N\times N}$ is the Laplacian of $\mathcal{G}$. Obviously, $\mathcal{L}_{\bar{\mathcal{G}}}$ has $N-1$ eigenvalues at N, $\mathcal{L}_{\mathcal{G}}$ has $N-1$ positive eigenvalues among which λ_2 is the smallest, and both $\mathcal{L}_{\bar{\mathcal{G}}}$ and $\mathcal{L}_{\mathcal{G}}$ have one eigenvalue at 0 with $(\frac{1}{\sqrt{N}}, \frac{1}{\sqrt{N}}, \ldots, \frac{1}{\sqrt{N}})$ being its eigenvector. Let $W \in \mathbb{R}^{N\times N}$ contain N orthonormal eigenvectors of $\mathcal{L}_{\mathcal{G}}$ in its columns. Then, $W^T\mathcal{L}_{\bar{\mathcal{G}}}W$ and $W^T\mathcal{L}_{\mathcal{G}}W$ are diagonal matrices similar to $\mathcal{L}_{\bar{\mathcal{G}}}$ and

$\mathcal{L}_{\mathcal{G}}$, and both contain the eigenvalue 0 in the same diagonal position. Hence, $\lambda_2 W^T \mathcal{L}_{\bar{\mathcal{G}}} W \leq N W^T \mathcal{L}_{\mathcal{G}} W$, so that $\lambda_2 \mathcal{L}_{\bar{\mathcal{G}}} \leq N \mathcal{L}_{\mathcal{G}}$. Applying this inequality to (7.46) and (7.47), we get $\frac{2\gamma}{\Theta}\lambda_2 V(\mathbf{x}(t)) \leq -\dot{V}(\mathbf{x}(t))$, i.e., (7.44). Finally, (7.45) follows from (7.44) the same way (7.38) does from (7.37). $\square$

Notice that in the special case where $n = 1$, $f_i(x) = \frac{1}{2}(x - x_i^*)^2 \ \forall i \in \mathcal{V}$, and $g_{\{i,j\}}(x) = \frac{1}{2}x^2 \ \forall \{i,j\} \in \mathcal{E}$, we may let the θ_i's, Θ_i's, and $\gamma_{\{i,j\}}$'s all be 1. In this case, Theorem 7.2 and Corollary 7.1 both yield $\|\mathbf{x}(t) - \mathbf{x}^*\| \leq \|\mathbf{x}(0) - \mathbf{x}^*\| e^{-\lambda_2 t} \ \forall t \geq 0$, which coincides with the well-known convergence rate result for the linear consensus algorithm $\dot{x}_i(t) = \sum_{j \in \mathcal{N}_i} x_j(t) - x_i(t) \ \forall t \geq 0 \ \forall i \in \mathcal{V}$, reported in [52]. Hence, Theorem 7.2 and Corollary 7.1 may be regarded as a generalization of such a result for distributed consensus, to distributed convex optimization.

The next theorem looks at the performance of the ZGS algorithms (7.32) from the other end, providing an upper bound $\tilde{\rho}$ on their exponential convergence rates that mirrors Theorem 7.2:

Theorem 7.3. *Consider the network modeled in Section 7.3 and the use of a ZGS algorithm described in* (7.32)*. Suppose Assumption 7.1 holds. Then,*

$$V(\mathbf{x}(t)) \geq V(\mathbf{x}(0)) e^{-\tilde{\rho} t}, \quad \forall t \geq 0, \tag{7.48}$$

$$\sum_{i \in \mathcal{V}} \Theta_i \|x_i(t) - x^*\|^2 \geq \sum_{i \in \mathcal{V}} \theta_i \|x_i(0) - x^*\|^2 e^{-\tilde{\rho} t}, \quad \forall t \geq 0, \tag{7.49}$$

where $\tilde{\rho} = \inf\{\varepsilon \in \mathbb{R} : \varepsilon \tilde{P} \geq \tilde{Q}\} > 0$, $\tilde{P} \in \mathbb{R}^{N \times N}$ is a positive definite matrix given by $\tilde{P} = \operatorname{diag}(\frac{\theta_1}{2}, \frac{\theta_2}{2}, \ldots, \frac{\theta_N}{2})$, and $\tilde{Q} = [\tilde{Q}_{ij}] \in \mathbb{R}^{N \times N}$ is a positive semidefinite matrix given by

$$\tilde{Q}_{ij} = \begin{cases} \sum_{\ell \in \mathcal{N}_i} \Gamma_{\{i,\ell\}}, & \textit{if } i = j, \\ -\Gamma_{\{i,j\}}, & \textit{if } \{i,j\} \in \mathcal{E}, \\ 0, & \textit{otherwise.} \end{cases} \tag{7.50}$$

Proof. From (7.2) and (7.12), $V(\mathbf{x}(t)) \geq \sum_{i\in\mathcal{V}} \frac{\theta_i}{2}\|x_i(t) - x^*\|^2 = (\mathbf{x}(t) - \mathbf{x}^*)^T(\tilde{P} \otimes I_n)(\mathbf{x}(t) - \mathbf{x}^*)$ $\forall t \geq 0$. From (7.42), (7.36), (7.7), (7.6), (7.33), and (7.50),

$$\begin{aligned}
-\dot{V}(\mathbf{x}(t)) &\leq \frac{1}{2}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i} \Gamma_{\{i,j\}}\|x_j(t) - x_i(t)\|^2 \\
&= \frac{1}{2}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i} \Gamma_{\{i,j\}}\|(x_j(t) - x^*) - (x_i(t) - x^*)\|^2 \\
&= (\mathbf{x}(t) - \mathbf{x}^*)^T(\tilde{Q} \otimes I_n)(\mathbf{x}(t) - \mathbf{x}^*), \quad \forall t \geq 0.
\end{aligned}$$

Like Q in (7.40), $\tilde{Q}$ in (7.50) is symmetric positive semidefinite with exactly one eigenvalue at 0. Thus, so is $\tilde{P}^{-1/2}\tilde{Q}\tilde{P}^{-1/2}$, where $\tilde{P}^{1/2} = \operatorname{diag}(\sqrt{\frac{\theta_1}{2}}, \sqrt{\frac{\theta_2}{2}}, \ldots, \sqrt{\frac{\theta_N}{2}})$ is the square root of $\tilde{P}$. Since $\tilde{\rho} = \inf\{\varepsilon \in \mathbb{R} : \varepsilon\tilde{P} \geq \tilde{Q}\}$ and $\forall \varepsilon \in \mathbb{R}$, $\varepsilon\tilde{P} \geq \tilde{Q} \Leftrightarrow \varepsilon I_N \geq \tilde{P}^{-1/2}\tilde{Q}\tilde{P}^{-1/2}$, $\tilde{\rho}$ is the largest eigenvalue of $\tilde{P}^{-1/2}\tilde{Q}\tilde{P}^{-1/2}$ which is positive and such that $\tilde{\rho}\tilde{P} \geq \tilde{Q}$. Therefore, $\tilde{\rho}V(\mathbf{x}(t)) \geq -\dot{V}(\mathbf{x}(t))$, proving (7.48). Finally, from (7.12), (7.34), (7.7), (7.5), (7.33), (7.48), and (7.2), we get (7.49). □

In contrast to ρ, the upper bound $\tilde{\rho}$ in Theorem 7.3 is the largest eigenvalue of $\tilde{P}^{-1/2}\tilde{Q}\tilde{P}^{-1/2}$. The next corollary is to Theorem 7.3 as Corollary 7.1 is to Theorem 7.2, giving another upper bound that is not as tight as $\tilde{\rho}$ but is explicit in the spectral radius $\lambda_N > 0$ of the graph Laplacian $\mathcal{L}_{\mathcal{G}}$:

Corollary 7.2. *With the setup of Theorem 7.3,*

$$V(\mathbf{x}(t)) \geq V(\mathbf{x}(0))e^{-\frac{2\Gamma}{\theta}\lambda_N t}, \quad \forall t \geq 0, \tag{7.51}$$

$$\|\mathbf{x}(t) - \mathbf{x}^*\| \geq \sqrt{\frac{\theta}{\Theta}}\|\mathbf{x}(0) - \mathbf{x}^*\|e^{-\frac{\Gamma}{\theta}\lambda_N t}, \quad \forall t \geq 0. \tag{7.52}$$

Proof. From the proof of Theorem 7.3, $\forall t \geq 0$, we have $V(\mathbf{x}(t)) \geq \sum_{i\in\mathcal{V}} \frac{\theta}{2}\|x_i(t) - x^*\|^2 = \frac{\theta}{2}\|\mathbf{x}(t) - \mathbf{x}^*\|^2$ and $-\dot{V}(\mathbf{x}(t)) \leq \frac{1}{2}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i} \Gamma\|(x_j(t) - x^*) - (x_i(t) -$

$x^*)\|^2 = \Gamma(\mathbf{x}(t) - \mathbf{x}^*)^T(\mathcal{L}_\mathcal{G} \otimes I_n)(\mathbf{x}(t) - \mathbf{x}^*) \leq \Gamma\lambda_N\|\mathbf{x}(t) - \mathbf{x}^*\|^2$. Consequently, $\frac{2\Gamma}{\theta}\lambda_N V(\mathbf{x}(t)) \geq -\dot{V}(\mathbf{x}(t))$, implying that (7.51) and (7.52) hold. $\square$

Note that for the special case below Corollary 7.1, we may let the $\Gamma_{\{i,j\}}$'s be 1, so that Theorem 7.3 and Corollary 7.2 both lead to $\|\mathbf{x}(t) - \mathbf{x}^*\| \geq \|\mathbf{x}(0) - \mathbf{x}^*\|e^{-\lambda_N t}$ $\forall t \geq 0$, which is again known. Finally, note that the above analysis provides a framework for studying the interplay among network topologies (i.e., $\mathcal{V}$ and $\mathcal{E}$), problem characteristics (i.e., the f_i's, θ_i's, and Θ_i's), and ZGS algorithm parameters (i.e., the $g_{\{i,j\}}$'s, $\gamma_{\{i,j\}}$'s, and $\Gamma_{\{i,j\}}$'s), which may be worthy of further research.

7.6 Conclusion

In this chapter, using a convexity-based Lyapunov function candidate, we have developed a set of continuous-time ZGS algorithms, which solve a class of distributed convex optimization problems over networks. We have established the asymptotic and exponential convergence of these algorithms and derived lower and upper bounds on their convergence rates. We have also shown that the ZGS algorithms for distributed convex optimization are closely related to the basic algorithms for distributed consensus, suggesting that the former may be extended in a number of directions just like the latter were, in ways that possibly parallel the latter.

Chapter 8 Conclusions

8.1 Summary

This dissertation is devoted to studying distributed computation and optimization over networks, which has become an active research area in recent years. The dissertation provides a collection of distributed algorithms that address three fundamental in-network computation and optimization problems: averaging, solving of positive definite linear equations, and convex optimization.

More specifically, a distributed asynchronous algorithm for averaging numbers across a network, referred to as CHA, is developed, which enables greedy, decentralized, feedback control of when to initiate an iteration. It is shown both analytically and numerically that the algorithm yields faster exponential convergence rates, i.e., is more efficient, than several existing averaging schemes over wireless networks.

To solve positive definite linear equations over networks, a distributed asynchronous algorithm called SE is constructed. It is capable of computing the solution over networks with time-varying node memberships, and mild sufficient conditions for its convergence are derived. In addition, the algorithm is specialized to wireless networks, leading to several distributed algorithms that are dramatically more efficient and scalable than two existing schemes.

Moreover, two gossip algorithms, PE and PB, for unconstrained, separable, convex optimization over networks with time-varying topologies are proposed. The former generalizes a well-known algorithm, while the latter relaxes a requirement of former, and both achieve asymptotic convergence to the unknown optimizer. Furthermore, the ideas of the former and the notion of feedback iteration control for CHA are tailored so as to develop another distributed asynchronous algorithm for the same problem, which uses a Bregman-divergence-based Lyapunov function to realize greedy, decentralized feedback iteration control. It is shown that this algorithm, referred to as CHE, is significantly more efficient than several existing subgradient algorithms. Finally, a family of continuous-time distributed algorithms called ZGS algorithms are constructed. Both lower and upper bounds on the exponential convergence rates of a subset of these algorithms are derived. Also, the findings on these algorithms may be regarded as a natural generalization of several classic algorithms and results for distributed consensus to distributed convex optimization.

8.2 Future Research

Several possible directions of future research on distributed computation and optimization over networks are as follows:

- *More general distributed optimization.* The distributed optimization problems addressed in this dissertation are limited to unconstrained, separable, convex optimization. Although many practical problems can be cast in this form, problems in the form of constrained, non-separable, or even non-convex optimization are also very common. Therefore, we intend to investigate how to design distributed algorithms for solving the aforementioned, more general optimization problems over networks with time-varying topologies and node memberships.

- *Decentralized feedback control.* The idea of greedy, decentralized feedback iteration control has been introduced in this dissertation, which significantly improves the bandwidth/energy efficiency of the proposed algorithms. We plan to explore the possibilities of other types of decentralized feedback control, in which every node in the network uses certain local information as feedback to control its behaviors besides when to initiate an iteration, so that algorithm performances could be enhanced.

- *Communications with delay, error, and quantization.* This dissertation assumes that all the communications are ideal, i.e., without delay, error, and quantization. We would like to study the effects caused by delay, error, and quantization on the algorithms proposed in this dissertation in future.

Bibliography

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “Wireless sensor networks: A survey,” *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[2] D. Bauso, L. Giarré, and R. Pesenti, “Non-linear protocols for optimal distributed consensus in networks of dynamic agents,” *Systems & Control Letters*, vol. 55, no. 11, pp. 918–928, 2006.

[3] L. Benmohamed and S. M. Meerkov, “Feedback control of congestion in packet switching networks: The case of a single congested node,” *IEEE/ACM Transactions on Networking*, vol. 1, pp. 693–708, 1993.

[4] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods.* Englewood Cliffs, NJ: Prentice-Hall, 1989.

[5] ——, “Some aspects of parallel and distributed iterative algorithms—a survey,” *Automatica*, vol. 27, no. 1, pp. 3–21, 1991.

[6] D. Blatt, A. O. Hero, and H. Gauchman, “A convergent incremental gradient method with a constant step size,” *SIAM Journal on Optimization*, vol. 18, no. 1, pp. 29–51, 2007.

[7] V. D. Blondel, J. M. Hendrickx, A. Olshevsky, and J. N. Tsitsiklis, “Convergence in multiagent coordination, consensus, and flocking,” in *Proc. IEEE Conference on Decision and Control and European Control Conference*, Seville, Spain, 2005, pp. 2996–3000.

[8] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, “Randomized gossip algorithms,” *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2508–2530, 2006.

[9] S. Boyd and L. Vandenberghe, *Convex Optimization.* New York, NY: Cambridge University Press, 2004.

[10] L. Bregman, "The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming," *USSR Computational Mathematics and Mathematical Physics*, vol. 7, no. 3, pp. 200–217, 1967.

[11] M. Cao, D. A. Spielman, and E. M. Yeh, "Accelerated gossip algorithms for distributed computation," in *Proc. Allerton Conference on Communication, Control, and Computing*, Monticello, IL, 2006, pp. 952–959.

[12] C. Castellano, S. Fortunato, and V. Loreto, "Statistical physics of social dynamics," *Reviews of Modern Physics*, vol. 81, no. 2, pp. 591–646, 2009.

[13] J.-Y. Chen, G. Pandurangan, and D. Xu, "Robust computation of aggregates in wireless sensor networks: Distributed randomized algorithms and analysis," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 9, pp. 987–1000, 2006.

[14] M. Chiang, S. Low, A. Calderbank, and J. Doyle, "Layering as optimization decomposition: A mathematical theory of network architectures," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 255–312, 2007.

[15] I. Chlamtac, M. Conti, and J. J.-N. Liu, "Mobile ad hoc networking: Imperatives and challenges," *Ad Hoc Networks*, vol. 1, no. 1, pp. 13–64, 2003.

[16] J. Cortés, "Finite-time convergent gradient flows with applications to network consensus," *Automatica*, vol. 42, no. 11, pp. 1993–2000, 2006.

[17] ——, "Distributed algorithms for reaching consensus on general functions," *Automatica*, vol. 44, no. 3, pp. 726–737, 2008.

[18] D. Culler, D. Estrin, and M. Srivastava, "Overview of sensor networks," *Computer*, vol. 37, no. 8, pp. 41–49, 2004.

[19] M. H. DeGroot, "Reaching a consensus," *Journal of the American Statistical Association*, vol. 69, no. 345, pp. 118–121, 1974.

[20] F. Fagnani and S. Zampieri, "Randomized consensus algorithms over large scale networks," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 4, pp. 634–649, 2008.

[21] L. Fang and P. J. Antsaklis, "On communication requirements for multi-agent consensus seeking," in *Networked Embedded Sensing and Control*, ser. Lecture Notes in Control and Information Sciences, P. J. Antsaklis

and P. Tabuada, Eds. Berlin, Germany: Springer-Verlag, 2006, vol. 331, pp. 53–67.

[22] G. J. Foschini and Z. Miljanic, "A simple distributed autonomous power control algorithm and its convergence," *IEEE Transactions on Vehicular Technology*, vol. 42, no. 4, pp. 641–646, 1993.

[23] R. Goebel and R. T. Rockafellar, "Local strong convexity and local lipschitz continuity of the gradient of convex functions," *Journal of Convex Analysis*, vol. 15, no. 2, pp. 263–270, 2008.

[24] Y. Hatano and M. Mesbahi, "Agreement over random networks," *IEEE Transactions on Automatic Control*, vol. 50, no. 11, pp. 1867–1872, 2005.

[25] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, 2003.

[26] M. Jelasity and A. Montresor, "Epidemic-style proactive aggregation in large overlay networks," in *Proc. IEEE International Conference on Distributed Computing Systems*, Tokyo, Japan, 2004, pp. 102–109.

[27] B. Johansson, T. Keviczky, M. Johansson, and K. H. Johansson, "Subgradient methods and consensus algorithms for solving convex optimization problems," in *Proc. IEEE Conference on Decision and Control*, Cancun, Mexico, 2008, pp. 4185–4190.

[28] B. Johansson, M. Rabi, and M. Johansson, "A simple peer-to-peer algorithm for distributed optimization in sensor networks," in *Proc. IEEE Conference on Decision and Control*, New Orleans, LA, 2007, pp. 4705–4710.

[29] B. Johansson, P. Soldati, and M. Johansson, "Mathematical decomposition techniques for distributed cross-layer optimization of data networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1535–1547, 2006.

[30] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *Proc. IEEE Symposium on Foundations of Computer Science*, Cambridge, MA, 2003, pp. 482–491.

[31] D. B. Kingston and R. W. Beard, "Discrete-time average-consensus under switching network topologies," in *Proc. American Control Conference*, Minneapolis, MN, 2006, pp. 3551–3556.

[32] I. Lobel and A. Ozdaglar, “Convergence analysis of distributed subgradient methods over random networks,” in *Proc. Allerton Conference on Communication, Control, and Computing*, Monticello, IL, 2008, pp. 353–360.

[33] J. Lorenz, “Continuous opinion dynamics under bounded confidence: A survey,” *International Journal of Modern Physics C*, vol. 18, no. 12, pp. 1819–1838, 2007.

[34] N. A. Lynch, *Distributed Algorithms.* San Francisco, CA: Morgan Kaufmann Publishers, 1996.

[35] S. Mascolo, D. Cavendish, and M. Gerla, “ATM rate-based congestion control using a smith predictor,” *Performance Evaluation*, vol. 31, pp. 51–65, 1997.

[36] M. Mehyar, D. Spanos, J. Pongsajapan, S. H. Low, and R. M. Murray, “Asynchronous distributed averaging on communication networks,” *IEEE/ACM Transactions on Networking*, vol. 15, no. 3, pp. 512–520, 2007.

[37] D. Mitra, “An asynchronous distributed algorithm for power control in cellular radio systems,” in *Proc. 4th WINLAB Workshop on Third Generation Wireless Information Networks*, New Brunswick, NJ, 1993, pp. 249–257.

[38] C. C. Moallemi and B. Van Roy, “Consensus propagation,” *IEEE Transactions on Information Theory*, vol. 52, no. 11, pp. 4753–4766, 2006.

[39] A. Montresor, M. Jelasity, and O. Babaoglu, “Robust aggregation protocols for large-scale overlay networks,” in *Proc. IEEE/IFIP International Conference on Dependable Systems and Networks*, Florence, Italy, 2004, pp. 19–28.

[40] L. Moreau, “Stability of multiagent systems with time-dependent communication links,” *IEEE Transactions on Automatic Control*, vol. 50, no. 2, pp. 169–182, 2005.

[41] D. Mosk-Aoyama and D. Shah, “Computing separable functions via gossip,” in *Proc. ACM Symposium on Principles of Distributed Computing*, Denver, CO, 2006, pp. 113–122.

[42] A. Nedić and D. P. Bertsekas, “Convergence rate of incremental subgradient algorithms,” in *Stochastic Optimization: Algorithms and Applications*, S. P. Uryasev and P. M. Pardalos, Eds. Norwell, MA: Kluwer Academic Publishers, 2001, pp. 223–264.

[43] ——, “Incremental subgradient methods for nondifferentiable optimization,” *SIAM Journal on Optimization*, vol. 12, no. 1, pp. 109–138, 2001.

[44] A. Nedić, D. P. Bertsekas, and V. S. Borkar, “Distributed asynchronous incremental subgradient methods,” in *Inherently Parallel Algorithms in Feasibility and Optimization and Their Applications*, D. Butnariu, Y. Censor, and S. Reich, Eds. Amsterdam, Holland: Elsevier, 2001, pp. 381–407.

[45] A. Nedić, A. Olshevsky, A. Ozdaglar, and J. N. Tsitsiklis, “Distributed subgradient methods and quantization effects,” in *Proc. IEEE Conference on Decision and Control*, Cancun, Mexico, 2008, pp. 4177–4184.

[46] A. Nedić and A. Ozdaglar, “On the rate of convergence of distributed subgradient methods for multi-agent optimization,” in *Proc. IEEE Conference on Decision and Control*, New Orleans, LA, 2007, pp. 4711–4716.

[47] ——, “Distributed subgradient methods for multi-agent optimization,” *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[48] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*. Norwell, MA: Kluwer Academic Publishers, 2004.

[49] R. Olfati-Saber, “Flocking for multi-agent dynamic systems: Algorithms and theory,” *IEEE Transactions on Automatic Control*, vol. 51, no. 3, pp. 401–420, 2006.

[50] R. Olfati-Saber, J. A. Fax, and R. M. Murray, “Consensus and cooperation in networked multi-agent systems,” *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[51] R. Olfati-Saber and R. M. Murray, “Consensus protocols for networks of dynamic agents,” in *Proc. American Control Conference*, Denver, CO, 2003, pp. 951–956.

[52] ——, “Consensus problems in networks of agents with switching topology and time-delays,” *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, 2004.

[53] A. Olshevsky and J. N. Tsitsiklis, “Convergence rates in distributed consensus and averaging,” in *Proc. IEEE Conference on Decision and Control*, San Diego, CA, 2006, pp. 3387–3392.

[54] ——, “On the nonexistence of quadratic Lyapunov functions for consensus algorithms,” *IEEE Transactions on Automatic Control*, vol. 53, no. 11, pp. 2642–2645, 2008.

[55] ——, “Convergence speed in distributed consensus and averaging,” *SIAM Journal on Control and Optimization*, vol. 48, no. 1, pp. 33–55, 2009.

[56] B. N. Oreshkin, M. J. Coates, and M. G. Rabbat, “Optimization and analysis of distributed averaging with short node memory,” *IEEE Transactions on Signal Processing*, vol. 58, no. 5, pp. 2850–2865, 2010.

[57] M. G. Rabbat and R. D. Nowak, “Distributed optimization in sensor networks,” in *Proc. International Symposium on Information Processing in Sensor Networks*, Berkeley, CA, 2004, pp. 20–27.

[58] ——, “Quantized incremental algorithms for distributed optimization,” *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 4, pp. 798–808, 2005.

[59] S. S. Ram, A. Nedić, and V. V. Veeravalli, “Stochastic incremental gradient descent for estimation in sensor networks,” in *Proc. Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, 2007, pp. 582–586.

[60] ——, “Asynchronous gossip algorithms for stochastic optimization,” in *Proc. IEEE Conference on Decision and Control*, Shanghai, China, 2009, pp. 3581–3586.

[61] ——, “Incremental stochastic subgradient algorithms for convex optimization,” *SIAM Journal on Optimization*, vol. 20, no. 2, pp. 691–717, 2009.

[62] ——, “Distributed stochastic subgradient projection algorithms for convex optimization,” *Journal of Optimization Theory and Applications*, vol. 147, no. 3, pp. 516–545, 2010.

[63] W. Ren and R. W. Beard, “Consensus seeking in multiagent systems under dynamically changing interaction topologies,” *IEEE Transactions on Automatic Control*, vol. 50, no. 5, pp. 655–661, 2005.

[64] D. S. Scherber and H. C. Papadopoulos, “Distributed computation of averages over ad hoc networks,” *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 4, pp. 776–787, 2005.

[65] S.-H. Son, M. Chiang, S. R. Kulkarni, and S. C. Schwartz, “The value of clustering in distributed estimation for sensor networks,” in *Proc. International Conference on Wireless Networks, Communications and Mobile Computing*, Maui, HI, 2005, pp. 969–974.

[66] D. P. Spanos, R. Olfati-Saber, and R. M. Murray, “Distributed sensor fusion using dynamic consensus,” in *Proc. IFAC World Congress*, Prague, Czech Republic, 2005.

[67] S. Sundaram and C. N. Hadjicostis, “Finite-time distributed consensus in graphs with time-invariant topologies,” in *Proc. American Control Conference*, New York, NY, 2007, pp. 711–716.

[68] A. Tahbaz-Salehi and A. Jadbabaie, “A one-parameter family of distributed consensus algorithms with boundary: From shortest paths to mean hitting times,” in *Proc. IEEE Conference on Decision and Control*, San Diego, CA, 2006, pp. 4664–4669.

[69] ——, “Small world phenomenon, rapidly mixing Markov chains, and average consensus algorithms,” in *Proc. IEEE Conference on Decision and Control*, New Orleans, LA, 2007, pp. 276–281.

[70] ——, “Consensus over ergodic stationary graph processes,” *IEEE Transactions on Automatic Control*, vol. 55, no. 1, pp. 225–230, 2010.

[71] B. Tavli and W. Heinzelman, *Mobile Ad Hoc Networks: Energy-Efficient Real-Time Data Communications.* Berlin, Germany: Springer-Verlag, 2006.

[72] J. N. Tsitsiklis, “Problems in decentralized decision making and computation,” Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, 1984.

[73] J. N. Tsitsiklis and M. Athans, “Convergence and asymptotic agreement in distributed decision problems,” *IEEE Transactions on Automatic Control*, vol. 29, no. 1, pp. 42–50, 1984.

[74] L. Xiao and S. Boyd, “Fast linear iterations for distributed averaging,” *Systems & Control Letters*, vol. 53, no. 1, pp. 65–78, 2004.

[75] L. Xiao, S. Boyd, and S.-J. Kim, "Distributed average consensus with least-mean-square deviation," *Journal of Parallel and Distributed Computing*, vol. 67, no. 1, pp. 33–46, 2007.

[76] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *Proc. International Symposium on Information Processing in Sensor Networks*, Los Angeles, CA, 2005, pp. 63–70.

[77] ——, "A space-time diffusion scheme for peer-to-peer least-squares estimation," in *Proc. International Conference on Information Processing in Sensor Networks*, Nashville, TN, 2006, pp. 168–176.

[78] M. Zhu and S. Martínez, "Dynamic average consensus on synchronous communication networks," in *Proc. American Control Conference*, Seattle, WA, 2008, pp. 4382–4387.

Appendix

Appendix A Proofs for Chapter 2

A.1 Proof of Theorem 2.2

To prove Theorem 2.2, we first prove the following lemma:

Lemma A.1. $V(\mathbf{x}(k)) \le \gamma \max_{i\in\mathcal{V}} \Delta V_i(\mathbf{x}(k))\ \forall k \in \mathbb{N}$, *where* γ *is as in* (2.26).

Proof. Let $k \in \mathbb{N}$. Notice from (2.14) that $\sum_{i\in\mathcal{V}} b_i = N$ and from (2.1), (2.7), (2.12), and (2.13) that $\sum_{i\in\mathcal{V}} b_i\hat{x}_i(k) = \sum_{i\in\mathcal{V}} b_i x^*$. Thus,

$$\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}} b_ib_j(\hat{x}_i(k)-\hat{x}_j(k))^2 = \sum_{j\in\mathcal{V}} b_j \sum_{i\in\mathcal{V}} b_i(\hat{x}_i(k)-x^*)^2 + \sum_{i\in\mathcal{V}} b_i \sum_{j\in\mathcal{V}} b_j(\hat{x}_j(k)-x^*)^2 \\ - 2\sum_{i\in\mathcal{V}} b_i(\hat{x}_i(k)-x^*)\sum_{j\in\mathcal{V}} b_j(\hat{x}_j(k)-x^*) = 2N\sum_{i\in\mathcal{V}} b_i(\hat{x}_i(k)-x^*)^2.$$

It follows from (2.16), (2.19), and (2.7) that

$$\begin{aligned} V(\mathbf{x}(k)) &= \frac{1}{2}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i} c_{\{i,j\}}(x_{\{i,j\}}(k)-\hat{x}_i(k))^2 + \frac{1}{2}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i} c_{\{i,j\}}(\hat{x}_i(k)-x^*)^2 \\ &\quad + \sum_{i\in\mathcal{V}}(\hat{x}_i(k)-x^*)\sum_{j\in\mathcal{N}_i} c_{\{i,j\}}(x_{\{i,j\}}(k)-\hat{x}_i(k)) \\ &= \frac{1}{2}\sum_{i\in\mathcal{V}}\Delta V_i(\mathbf{x}(k)) + \sum_{i\in\mathcal{V}} b_i(\hat{x}_i(k)-x^*)^2 \qquad\qquad (A.1) \\ &\le \tfrac{N}{2}\max_{i\in\mathcal{V}}\Delta V_i(\mathbf{x}(k)) + \frac{1}{2N}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}} b_ib_j(\hat{x}_i(k)-\hat{x}_j(k))^2 \\ &= \frac{N}{2}\max_{i\in\mathcal{V}}\Delta V_i(\mathbf{x}(k)) + \frac{1}{2N}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i} b_ib_j(\hat{x}_i(k)-\hat{x}_j(k))^2 \\ &\quad + \frac{1}{2N}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}-\mathcal{N}_i-\{i\}} b_ib_j(\hat{x}_i(k)-\hat{x}_j(k))^2. \qquad\qquad (A.2) \end{aligned}$$

Note from (2.19) that

$$\begin{aligned}
N\max_{i\in\mathcal{V}}\Delta V_i(\mathbf{x}(k)) &\geq \sum_{i\in\mathcal{V}} b_i \Delta V_i(\mathbf{x}(k))\\
&= \sum_{\{i,j\}\in\mathcal{E}} b_i c_{\{i,j\}}(\hat{x}_i(k)-x_{\{i,j\}}(k))^2 + b_j c_{\{i,j\}}(\hat{x}_j(k)-x_{\{i,j\}}(k))^2\\
&\geq \sum_{\{i,j\}\in\mathcal{E}} \frac{b_i b_j c_{\{i,j\}}}{b_i+b_j}(\hat{x}_i(k)-\hat{x}_j(k))^2.
\end{aligned}$$

Hence,

$$\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i} b_i b_j(\hat{x}_i(k)-\hat{x}_j(k))^2 \leq 2\alpha N \max_{i\in\mathcal{V}}\Delta V_i(\mathbf{x}(k)). \tag{A.3}$$

Next, it can be shown via (2.19) that $\forall i\in\mathcal{V}$ with $|\mathcal{N}_i|\geq 2$, $\forall j,\ell\in\mathcal{N}_i$ with $j\neq\ell$, $c_{\{i,j\}}c_{\{i,\ell\}}(x_{\{i,j\}}(k)-x_{\{i,\ell\}}(k))^2 \leq (c_{\{i,j\}}+c_{\{i,\ell\}})(c_{\{i,j\}}(x_{\{i,j\}}(k)-\hat{x}_i(k))^2 + c_{\{i,\ell\}}(x_{\{i,\ell\}}(k)-\hat{x}_i(k))^2) \leq (c_{\{i,j\}}+c_{\{i,\ell\}})\Delta V_i(\mathbf{x}(k))$, implying that $|x_{\{i,j\}}(k)-x_{\{i,\ell\}}(k)| \leq \big(\max_{p\in\mathcal{V}}\Delta V_p(\mathbf{x}(k))(\frac{1}{c_{\{i,j\}}}+\frac{1}{c_{\{i,\ell\}}})\big)^{\frac{1}{2}}$. In addition, $\forall i\in\mathcal{V}$, $\forall j\in\mathcal{N}_i$, $|\hat{x}_i(k)-x_{\{i,j\}}(k)| \leq \big(\frac{\max_{p\in\mathcal{V}}\Delta V_p(\mathbf{x}(k))}{c_{\{i,j\}}}\big)^{\frac{1}{2}}$ because of (2.19). For any $i,j\in\mathcal{V}$ with $i\neq j$, let the sequence $(a_1,a_2,\ldots,a_{m_{ij}})$ represent a shortest path from node i to node j, where $a_1=i$, $a_{m_{ij}}=j$, $\{a_\ell,a_{\ell+1}\}\in\mathcal{E}$ $\forall\ell\in\{1,2,\ldots,m_{ij}-1\}$, and $2\leq m_{ij}\leq D+1$. Then, it follows from (2.14), the triangle inequality, and the root-mean square-arithmetic mean-geometric mean inequality that

$$\begin{aligned}
|\hat{x}_i(k)-\hat{x}_j(k)| &\leq \big(\max_{p\in\mathcal{V}}\Delta V_p(\mathbf{x}(k))\big)^{\frac{1}{2}}\Big(\big(\frac{|\mathcal{N}_{a_1}|\cdot|\mathcal{N}_{a_2}|}{|\mathcal{N}_{a_1}|+|\mathcal{N}_{a_2}|}\big)^{\frac{1}{2}} + \sum_{\ell=2}^{m_{ij}-1}\big(\frac{|\mathcal{N}_{a_{\ell-1}}|\cdot|\mathcal{N}_{a_\ell}|}{|\mathcal{N}_{a_{\ell-1}}|+|\mathcal{N}_{a_\ell}|}\\
&\quad + \frac{|\mathcal{N}_{a_\ell}|\cdot|\mathcal{N}_{a_{\ell+1}}|}{|\mathcal{N}_{a_\ell}|+|\mathcal{N}_{a_{\ell+1}}|}\big)^{\frac{1}{2}} + \big(\frac{|\mathcal{N}_{a_{m_{ij}-1}}|\cdot|\mathcal{N}_{a_{m_{ij}}}|}{|\mathcal{N}_{a_{m_{ij}-1}}|+|\mathcal{N}_{a_{m_{ij}}}|}\big)^{\frac{1}{2}}\Big)\\
&\leq \big(m_{ij}\max_{p\in\mathcal{V}}\Delta V_p(\mathbf{x}(k))\big)^{\frac{1}{2}}\Big(\frac{|\mathcal{N}_{a_1}|+|\mathcal{N}_{a_2}|}{4} + \sum_{\ell=2}^{m_{ij}-1}\big(\frac{|\mathcal{N}_{a_{\ell-1}}|+|\mathcal{N}_{a_\ell}|}{4}+\frac{|\mathcal{N}_{a_\ell}|+|\mathcal{N}_{a_{\ell+1}}|}{4}\big)\\
&\quad + \frac{|\mathcal{N}_{a_{m_{ij}-1}}|+|\mathcal{N}_{a_{m_{ij}}}|}{4}\Big)^{\frac{1}{2}} \leq \big(m_{ij}\max_{p\in\mathcal{V}}\Delta V_p(\mathbf{x}(k))\sum_{\ell=1}^{m_{ij}}|\mathcal{N}_{a_\ell}|\big)^{\frac{1}{2}}.
\end{aligned}$$

Next, we show that $\forall i, j \in \mathcal{V}$ with $i \neq j$, each node $\ell \in \mathcal{V} - \{a_1, a_2, \ldots, a_{m_{ij}}\}$ has at most 3 one-hop neighbors in $\{a_1, a_2, \ldots, a_{m_{ij}}\}$. Clearly, this statement is true for $m_{ij} \leq 3$. For $m_{ij} \geq 4$, assume to the contrary that $\exists \ell \in \mathcal{V} - \{a_1, a_2, \ldots, a_{m_{ij}}\}$ such that $\mathcal{N}_\ell \cap \{a_1, a_2, \ldots, a_{m_{ij}}\} = \{a_{i_1}, a_{i_2}, \ldots, a_{i_n}\}$ for some $1 \leq i_1 < i_2 < \cdots < i_n \leq m_{ij}$ and $n \geq 4$. Then, $(a_1, \ldots, a_{i_1}, \ell, a_{i_n}, \ldots, a_{m_{ij}})$ is a path shorter than the shortest path $(a_1, a_2, \ldots, a_{m_{ij}})$, which is a contradiction. Therefore, the statement is true. Consequently, $\sum_{\ell=1}^{m_{ij}} |\mathcal{N}_{a_\ell}| \leq 3(N - m_{ij}) + 2(m_{ij} - 1) = 3N - m_{ij} - 2$. It follows that $\forall i, j \in \mathcal{V}$ with $i \neq j$, $(\hat{x}_i(k) - \hat{x}_j(k))^2 \leq m_{ij}(3N - m_{ij} - 2) \max_{p \in \mathcal{V}} \Delta V_p(\mathbf{x}(k))$. Since $m_{ij} \leq D + 1 \leq N$, $(\hat{x}_i(k) - \hat{x}_j(k))^2 \leq \big(3(N-1) - D\big)(D+1) \max_{p\in\mathcal{V}} \Delta V_p(\mathbf{x}(k))$. Due to this and to $\sum_{i\in\mathcal{V}} \sum_{j \in \mathcal{V} - \mathcal{N}_i - \{i\}} b_i b_j = \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{V}} b_i b_j - \beta = N^2 - \beta$, we have $\sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{V}-\mathcal{N}_i-\{i\}} b_i b_j (\hat{x}_i(k) - \hat{x}_j(k))^2 \leq (N^2 - \beta)\big(3(N-1) - D\big)(D+1) \max_{p\in\mathcal{V}} \Delta V_p(\mathbf{x}(k))$. This, along with (A.3), (A.2), and (2.26), implies $V(\mathbf{x}(k)) \leq \gamma \max_{i\in\mathcal{V}} \Delta V_i(\mathbf{x}(k))$. $\square$

Because of (2.20), (2.22), and Lemma A.1, we have $V(\mathbf{x}(k-1)) - V(\mathbf{x}(k)) \geq \frac{V(\mathbf{x}(k-1))}{\gamma}$ $\forall k \in \mathbb{P}$, which is exactly (2.23). To prove (2.24) and (2.25), note that (2.23) implies $V(\mathbf{x}(k)) \leq (1 - \frac{1}{\gamma})^k V(\mathbf{x}(0))$ $\forall k \in \mathbb{N}$. Moreover, note from (2.16) and (2.14) that $V(\mathbf{x}(k)) \geq (\min_{\{i,j\}\in\mathcal{E}} c_{\{i,j\}}) \|\mathbf{x}(k) - x^*\mathbf{1}_L\|^2$ $\forall k \in \mathbb{N}$ where $\min_{\{i,j\}\in\mathcal{E}} c_{\{i,j\}} \geq \frac{2}{\max_{i\in\mathcal{V}} |\mathcal{N}_i|}$. Furthermore, note from (A.1) and (2.14) that $V(\mathbf{x}(k)) \geq (\min_{i\in\mathcal{V}} b_i) \|\hat{\mathbf{x}}(k) - x^*\mathbf{1}_N\|^2$ $\forall k \in \mathbb{N}$ where $\min_{i\in\mathcal{V}} b_i \geq \frac{1}{2}(1 + \frac{\min_{i\in\mathcal{V}} |\mathcal{N}_i|}{\max_{i\in\mathcal{V}} |\mathcal{N}_i|})$. Thus, (2.24) and (2.25) hold. To derive the bounds on α, notice from (2.14) that $\frac{b_i + b_j}{c_{\{i,j\}}} = \frac{1}{2} + (1 + \frac{1}{2}\sum_{\ell\in\mathcal{N}_i - \{j\}} \frac{1}{|\mathcal{N}_\ell|} + \frac{1}{2}\sum_{\ell\in\mathcal{N}_j - \{i\}} \frac{1}{|\mathcal{N}_\ell|}) / (\frac{1}{|\mathcal{N}_i|} + \frac{1}{|\mathcal{N}_j|}) \leq \frac{1}{2} + (1 + \frac{\max_{\ell\in\mathcal{V}} |\mathcal{N}_\ell| - 1}{\min_{\ell\in\mathcal{V}} |\mathcal{N}_\ell|}) / (\frac{2}{\max_{\ell\in\mathcal{V}} |\mathcal{N}_\ell|}) \leq \frac{N^2 - 2N + 2}{2}$ $\forall \{i,j\} \in \mathcal{E}$. Similarly, it can be shown that $\frac{b_i + b_j}{c_{\{i,j\}}} \geq 1$ $\forall \{i,j\} \in \mathcal{E}$. Hence, $\alpha \in [1, \frac{N^2 - 2N + 2}{2}]$. To derive the bounds on β, observe that $\beta \leq \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{V}} b_i b_j = N^2$. Also, $\sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{N}_i} b_i b_j \geq 2L \cdot \big(\frac{1}{2}(1 + \frac{\min_{\ell\in\mathcal{V}} |\mathcal{N}_\ell|}{\max_{\ell\in\mathcal{V}} |\mathcal{N}_\ell|})\big)^2 \geq \frac{L}{2}(1 + \frac{1}{N-1})^2$ and $\sum_{i\in\mathcal{V}} b_i^2 \geq \frac{1}{N}(\sum_{i\in\mathcal{V}} b_i)^2 = N$.

Therefore, $\beta \in [N + \frac{L}{2}(1 + \frac{1}{N-1})^2, N^2]$. Finally, using (2.26), the bounds on α and β, and the properties $L \geq N-1$ and $\big(3(N-1)-D\big)(D+1) \leq 2N(N-1)$, we obtain $\gamma \in [\frac{N}{2}+1, N^3 - 2N^2 + \frac{N}{2} + 1]$.

A.2 Proof of Theorem 2.3

Lemma A.2. *$V(\mathbf{x}(k)) \leq \gamma \max_{i\in\mathcal{V}} \Delta V_i(\mathbf{x}(k))$ $\forall k \in \mathbb{N}$, where γ is as in S1 for a path graph with $N \geq 4$, S2 for a cycle graph, S3 for a K-regular graph with $K \geq 2$, and S4 for a (N,K,λ,μ)-strongly regular graph with $\mu \geq 1$.*

Proof. Let $k \in \mathbb{N}$. First, suppose $\mathcal{G}$ is a path graph with $N \geq 4$ and $\mathcal{E} = \{\{1,2\},\{2,3\},\ldots,\{N-1,N\}\}$. Note from (2.1), (2.12), (2.13), and (2.14) that $\sum_{\{i,j\}\in\mathcal{E}} \sum_{\{p,q\}\in\mathcal{E}} c_{\{i,j\}} c_{\{p,q\}} (x_{\{i,j\}}(k) - x_{\{p,q\}}(k))^2 = 2N \sum_{\{i,j\}\in\mathcal{E}} c_{\{i,j\}} (x_{\{i,j\}}(k) - x^*)^2$. This, along with (2.16) and (2.14), implies that

$$
\begin{aligned}
V(\mathbf{x}(k)) &= \frac{1}{2N} \sum_{\{i,j\}\in\mathcal{E}} \sum_{\{p,q\}\in\mathcal{E}} c_{\{i,j\}} c_{\{p,q\}} (x_{\{i,j\}}(k) - x_{\{p,q\}}(k))^2 \qquad \text{(A.4)}\\
&= \frac{1}{2N}\Big(\sum_{\{i,j\}\in\mathcal{E}'} \sum_{\{p,q\}\in\mathcal{E}'} (x_{\{i,j\}}(k) - x_{\{p,q\}}(k))^2 + 3 \sum_{\{i,j\}\in\mathcal{E}'} (x_{\{1,2\}}(k) - x_{\{i,j\}}(k))^2 \\
&\quad + 3 \sum_{\{i,j\}\in\mathcal{E}'} (x_{\{N-1,N\}}(k) - x_{\{i,j\}}(k))^2 + \frac{9}{2}(x_{\{1,2\}}(k) - x_{\{N-1,N\}}(k))^2 \Big), \qquad \text{(A.5)}
\end{aligned}
$$

where $\mathcal{E}' = \mathcal{E} - \{\{1,2\},\{N-1,N\}\}$. Observe from (2.7), (2.14), and (2.19) that $(x_{\{i-1,i\}}(k) - x_{\{i,i+1\}}(k))^2 = \frac{5}{3}\Delta V_i(\mathbf{x}(k))$ $\forall i \in \{2, N-1\}$ and $(x_{\{i-1,i\}}(k) - x_{\{i,i+1\}}(k))^2 = 2\Delta V_i(\mathbf{x}(k))$ $\forall i \in \{3,4,\ldots,N-2\}$. By the root-mean square-arithmetic mean inequality,

$$
\begin{aligned}
\sum_{\{i,j\}\in\mathcal{E}'} \sum_{\{p,q\}\in\mathcal{E}'} (x_{\{i,j\}}(k) - x_{\{p,q\}}(k))^2 &= 2 \sum_{i=2}^{N-3} \sum_{j=i+1}^{N-2} (x_{\{i,i+1\}}(k) - x_{\{j,j+1\}}(k))^2 \\
&\leq 2 \sum_{i=2}^{N-3} \sum_{j=i+1}^{N-2} (j-i) \sum_{\ell=i+1}^{j} (x_{\{\ell-1,\ell\}}(k) - x_{\{\ell,\ell+1\}}(k))^2
\end{aligned}
$$

$$= 2(N-3)\sum_{i=3}^{N-2}(N-i-1)(i-2)\Delta V_i(\mathbf{x}(k)).$$

Moreover, $3\sum_{\{i,j\}\in\mathcal{E}'}(x_{\{1,2\}}(k)-x_{\{i,j\}}(k))^2 \leq 3\sum_{i=2}^{N-2}(i-1)\sum_{j=2}^{i}(x_{\{j-1,j\}}(k)-x_{\{j,j+1\}}(k))^2 = \frac{5}{2}(N-2)(N-3)\Delta V_2(\mathbf{x}(k)) + 3\sum_{i=3}^{N-2}(N+i-4)(N-i-1)\Delta V_i(\mathbf{x}(k))$. Similarly, $3\sum_{\{i,j\}\in\mathcal{E}'}(x_{\{N-1,N\}}(k)-x_{\{i,j\}}(k))^2 \leq \frac{5}{2}(N-2)(N-3)\Delta V_{N-1}(\mathbf{x}(k)) + 3\sum_{i=3}^{N-2}(2N-i-3)(i-2)\Delta V_i(\mathbf{x}(k))$. Finally, $\frac{9}{2}(x_{\{1,2\}}(k)-x_{\{N-1,N\}}(k))^2 \leq \frac{9}{2}(N-2)\sum_{i=2}^{N-1}(x_{\{i-1,i\}}(k)-x_{\{i,i+1\}}(k))^2 = 3(N-2)\big(\frac{5}{2}\Delta V_2(\mathbf{x}(k)) + \frac{5}{2}\Delta V_N(\mathbf{x}(k)) + 3\sum_{i=3}^{N-2}\Delta V_i(\mathbf{x}(k))\big)$. Combining the above with (A.5) yields $V(\mathbf{x}(k)) \leq \gamma \max_{i\in\mathcal{V}}\Delta V_i(\mathbf{x}(k))$ where γ is as in S1.

Now suppose $\mathcal{G}$ is a cycle graph with $\mathcal{E} = \{\{1,2\},\{2,3\},\ldots,\{N-1,N\},\{N,1\}\}$. Also suppose N is odd. Let $\mathbf{y}\in\mathbb{R}^N$ be a permutation of $\mathbf{x}(k)$ such that $y_{\{N,1\}} \leq y_{\{1,2\}} \leq y_{\{N,N-1\}} \leq y_{\{2,3\}} \leq y_{\{N-1,N-2\}} \leq \cdots \leq y_{\{\frac{N-1}{2},\frac{N+1}{2}\}} \leq y_{\{\frac{N+3}{2},\frac{N+1}{2}\}}$. Then, since (A.4) holds for any graph and due to (2.14), $V(\mathbf{y}) = V(\mathbf{x}(k))$. Also, due to (2.19) and (2.14), $\max_{i\in\mathcal{V}}\Delta V_i(\mathbf{y}) \leq \max_{i\in\mathcal{V}}\Delta V_i(\mathbf{x}(k))$. For convenience, let $M = 2\max_{i\in\mathcal{V}}\Delta V_i(\mathbf{y})$ and relabel $(y_{\{N,1\}}, y_{\{1,2\}}, y_{\{N,N-1\}}, y_{\{2,3\}}, y_{\{N-1,N-2\}}, \ldots, y_{\{\frac{N-1}{2},\frac{N+1}{2}\}}, y_{\{\frac{N+3}{2},\frac{N+1}{2}\}})$ as $(z_1, z_2, \ldots, z_N)$. Then, we can write $V(\mathbf{y}) = \frac{1}{2N}\sum_{i=1}^{N}\sum_{j=1}^{N}(z_i-z_j)^2 = \frac{1}{N}(C_1+C_2)$, where $C_1 = \sum_{i=1}^{\frac{N-1}{2}}(z_1-z_{2i})^2 + (z_1-z_{2i+1})^2 + (z_{2i}-z_{2i+1})^2$ and $C_2 = \sum_{i=1}^{\frac{N-3}{2}}\sum_{j=i+1}^{\frac{N-1}{2}}(z_{2i}-z_{2j+1})^2 + (z_{2i+1}-z_{2j})^2 + (z_{2i}-z_{2j})^2 + (z_{2i+1}-z_{2j+1})^2$. Moreover, from (2.7), (2.14), and (2.19), we get $z_2 - z_1 \leq \sqrt{M}$, $z_N - z_{N-1} \leq \sqrt{M}$, and $z_{i+2} - z_i \leq \sqrt{M}$ $\forall i \in \{1,2,\ldots,N-2\}$. Due to the property $(a-b)^2 + (a-c)^2 + (b-c)^2 \leq 2(a-c)^2$ $\forall a,b,c\in\mathbb{R}$ with $a \leq b \leq c$, we have $C_1 \leq \sum_{i=1}^{\frac{N-1}{2}} 2(z_1-z_{2i+1})^2 \leq \sum_{i=1}^{\frac{N-1}{2}} 2i^2M = \frac{N(N^2-1)}{12}M$. In addition, from the property $(a-d)^2 + (b-c)^2 \leq (a-b)^2 + (a-c)^2 + (b-d)^2 + (c-d)^2$ $\forall a,b,c,d\in\mathbb{R}$, we have $C_2 \leq \sum_{i=1}^{\frac{N-3}{2}}\sum_{j=i+1}^{\frac{N-1}{2}} 2(z_{2i}-z_{2j})^2 + 2(z_{2i+1}-z_{2j+1})^2 + (z_{2i}-z_{2i+1})^2 + (z_{2j}-z_{2j+1})^2 \leq \sum_{i=1}^{\frac{N-3}{2}}\sum_{j=i+1}^{\frac{N-1}{2}}\big(4(i-j)^2M +$

$2M\big) = \frac{(N-1)(N-3)(N^2+11)}{48}M$. Combining the above, we obtain $V(\mathbf{x}(k)) = V(\mathbf{y}) \le \frac{\gamma}{2}M \le \gamma \max_{i\in\mathcal{V}} \Delta V_i(\mathbf{x}(k))$ where γ is as in S2. Next, suppose N is even. Similarly, let $\mathbf{y} \in \mathbb{R}^N$ be a permutation of $\mathbf{x}(k)$ such that $y_{\{N,1\}} \le y_{\{1,2\}} \le y_{\{N,N-1\}} \le y_{\{2,3\}} \le y_{\{N-1,N-2\}} \le \cdots \le y_{\{\frac{N}{2}-1,\frac{N}{2}\}} \le y_{\{\frac{N}{2}+2,\frac{N}{2}+1\}} \le y_{\{\frac{N}{2},\frac{N}{2}+1\}}$. Observe from (A.4), (2.14), and (2.19) that $V(\mathbf{y}) = V(\mathbf{x}(k))$ and $\max_{i\in\mathcal{V}} \Delta V_i(\mathbf{y}) \le \max_{i\in\mathcal{V}} \Delta V_i(\mathbf{x}(k))$. As before, let $M = 2\max_{i\in\mathcal{V}} \Delta V_i(\mathbf{y})$ and relabel $(y_{\{N,1\}}, y_{\{1,2\}}, y_{\{N,N-1\}}, y_{\{2,3\}}, y_{\{N-1,N-2\}}, \ldots, y_{\{\frac{N}{2}-1,\frac{N}{2}\}}, y_{\{\frac{N}{2}+2,\frac{N}{2}+1\}}, y_{\{\frac{N}{2},\frac{N}{2}+1\}})$ as $(z_1, z_2, \ldots, z_N)$. Then, $V(\mathbf{y}) = \frac{1}{2N}\sum_{i=1}^N \sum_{j=1}^N (z_i - z_j)^2 = \frac{1}{N}(C_1 + C_2 + C_3)$, where $C_1 = \sum_{i=1}^{\frac{N}{2}-1} (z_1 - z_{2i})^2 + (z_1 - z_{2i+1})^2 + (z_{2i} - z_{2i+1})^2 + (z_N - z_{2i})^2 + (z_N - z_{2i+1})^2$, $C_2 = \sum_{i=1}^{\frac{N}{2}-2} \sum_{j=i+1}^{\frac{N}{2}-1} (z_{2i} - z_{2j+1})^2 + (z_{2i+1} - z_{2j})^2 + (z_{2i} - z_{2j})^2 + (z_{2i+1} - z_{2j+1})^2$, and $C_3 = (z_1 - z_N)^2$. Moreover, $z_2 - z_1 \le \sqrt{M}$, $z_N - z_{N-1} \le \sqrt{M}$, and $z_{i+2} - z_i \le \sqrt{M}$ $\forall i \in \{1, 2, \ldots, N-2\}$. Using the above properties, it can be shown that $C_1 \le C_1 + \sum_{i=1}^{\frac{N}{2}-1} (z_{2i} - z_{2i+1})^2 \le \sum_{i=1}^{\frac{N}{2}-1} 2(z_1 - z_{2i+1})^2 + 2(z_N - z_{2i})^2 \le \sum_{i=1}^{\frac{N}{2}-1} 2i^2M + 2(\frac{N}{2}-i)^2M = \frac{N(N-1)(N-2)}{6}M$, $C_2 \le \sum_{i=1}^{\frac{N}{2}-2} \sum_{j=i+1}^{\frac{N}{2}-1} \big(4(i-j)^2M + 2M\big) = \frac{(N-2)(N-4)(N^2-2N+12)}{48}M$, and $C_3 \le \frac{N^2}{4}M$. It follows that $V(\mathbf{x}(k)) \le \gamma \max_{i\in\mathcal{V}} \Delta V_i(\mathbf{x}(k))$ where γ is as in S2.

Next, suppose $\mathcal{G}$ is a K-regular graph with $K \ge 2$. Due to (2.14) and (2.19), $\sum_{i\in\mathcal{V}} \Delta V_i(\mathbf{x}(k)) = \frac{2}{K}\sum_{\{i,j\}\in\mathcal{E}} (\hat{x}_i(k) - x_{\{i,j\}}(k))^2 + (\hat{x}_j(k) - x_{\{i,j\}}(k))^2 \ge \frac{1}{K}\sum_{\{i,j\}\in\mathcal{E}} (\hat{x}_i(k) - \hat{x}_j(k))^2$, implying that

$$\sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{N}_i} (\hat{x}_i(k) - \hat{x}_j(k))^2 \le 2K \sum_{i\in\mathcal{V}} \Delta V_i(\mathbf{x}(k)). \tag{A.6}$$

Again, because of (2.14) and (2.19), $\forall i \in \mathcal{V}$, $\forall j \in \mathcal{N}_i$, $(x_{\{i,j\}}(k) - \hat{x}_i(k))^2 \le \frac{K}{2}\max_{p\in\mathcal{V}} \Delta V_p(\mathbf{x}(k))$. Moreover, $\forall i \in \mathcal{V}$, $\forall j, \ell \in \mathcal{N}_i$ with $j \ne \ell$, $(x_{\{i,j\}}(k) - x_{\{i,\ell\}}(k))^2 \le 2\big((x_{\{i,j\}}(k) - \hat{x}_i(k))^2 + (x_{\{i,\ell\}}(k) - \hat{x}_i(k))^2\big) \le K \max_{p\in\mathcal{V}} \Delta V_p(\mathbf{x}(k))$. Via the preceding two inequalities and the root-mean square-arithmetic mean inequality, it can be shown that $\forall i \in \mathcal{V}$, $\forall j \in \mathcal{V} - \mathcal{N}_i - \{i\}$, $(\hat{x}_i(k) - \hat{x}_j(k))^2 \le$

$(D+1)(\frac{K}{2}\max_{p\in\mathcal{V}}\Delta V_p(\mathbf{x}(k))\cdot 2 + K\max_{p\in\mathcal{V}}\Delta V_p(\mathbf{x}(k))\cdot(D-1)) = KD(D+1)\max_{p\in\mathcal{V}}\Delta V_p(\mathbf{x}(k))$. It then follows from (A.2), (2.14), and (A.6) that $V(\mathbf{x}(k)) \le \gamma\max_{i\in\mathcal{V}}\Delta V_i(\mathbf{x}(k))$ where γ is as in S3.

Finally, suppose $\mathcal{G}$ is a (N,K,λ,μ)-strongly regular graph with $\mu \ge 1$, which means that it is a K-regular graph with $K \ge 2$ and with every two non-adjacent nodes having μ common neighbors. For every $i \in \mathcal{V}$ and $j \in \mathcal{V}-\mathcal{N}_i-\{i\}$, let $\{q_{ij1}, q_{ij2}, \ldots, q_{ij\mu}\} = \mathcal{N}_i \cap \mathcal{N}_j$. Then, from (2.14) and (2.19),

$$\begin{aligned}
\mu\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}-\mathcal{N}_i-\{i\}}&(\hat{x}_i(k)-\hat{x}_j(k))^2 = \sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}-\mathcal{N}_i-\{i\}}\sum_{\ell=1}^{\mu}(\hat{x}_i(k)-\hat{x}_j(k))^2\\
&\le 4\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}-\mathcal{N}_i-\{i\}}\sum_{\ell=1}^{\mu}\big((\hat{x}_i(k)-x_{\{i,q_{ij\ell}\}}(k))^2+(x_{\{i,q_{ij\ell}\}}(k)-\hat{x}_{q_{ij\ell}}(k))^2\\
&\quad+(\hat{x}_{q_{ij\ell}}(k)-x_{\{j,q_{ij\ell}\}}(k))^2+(x_{\{j,q_{ij\ell}\}}(k)-\hat{x}_j(k))^2\big)\\
&\le 2K\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}-\mathcal{N}_i-\{i\}}\big(\Delta V_i(\mathbf{x}(k))+\sum_{\ell=1}^{\mu}\Delta V_{q_{ij\ell}}(\mathbf{x}(k))+\Delta V_j(\mathbf{x}(k))\big)\\
&\le 2KN(N-K-1)(2+\mu)\max_{p\in\mathcal{V}}\Delta V_p(\mathbf{x}(k)).
\end{aligned}$$

This, along with (A.2), (2.14), and (A.6), implies that $V(\mathbf{x}(k)) \le \gamma\max_{i\in\mathcal{V}}\Delta V_i(\mathbf{x}(k))$ where γ is as in S4. □

Note that in the proof of Theorem 2.2 in Appendix A.1, Lemma A.1 is used to derive (2.23)–(2.25). In the same way, (2.23)–(2.25) can be derived using Lemma A.2, completing the proof of Theorem 2.3.

A.3 Proof of Theorem 2.4

Let γ be as in (2.26) for a general graph or as in S1–S4 for a specific graph. Note that Lemmas A.1 and A.2 are independent of $(u(k))_{k=1}^{\infty}$ and, thus, hold for CHA as well. Hence,

$$V(\mathbf{x}(k)) \le \gamma \max_{i\in\mathcal{V}} \Delta V_i(\mathbf{x}(k)), \quad \forall k \in \mathbb{N}. \tag{A.7}$$

Next, analyzing Algorithm 2.3 with $\varepsilon(v) = 0\ \forall v \in [0,\infty)$, we see that

$$V(\mathbf{x}(1)) = V(\mathbf{x}(0)) - \max_{i\in\mathcal{V}} \Delta V_i(\mathbf{x}(0)), \tag{A.8}$$

$$V(\mathbf{x}(k+1)) \le V(\mathbf{x}(k)) - \min\{\max_{i\in\mathcal{V}} \Delta V_i(\mathbf{x}(k)), \Phi^{-1}(t(k))\}, \quad \forall k \in \mathbb{P}, \tag{A.9}$$

$$t(k+1) = \max\{\Phi(\max_{i\in\mathcal{V}} \Delta V_i(\mathbf{x}(k))), t(k)\}, \quad \forall k \in \mathbb{N}. \tag{A.10}$$

With (A.7)–(A.10), we now show by induction that $\forall k \in \mathbb{P}$, $V(\mathbf{x}(k)) \le (1 - \frac{1}{\gamma})V(\mathbf{x}(k-1))$ and $t(k) \le \Phi(\frac{V(\mathbf{x}(k-1))}{\gamma})$. Let $k = 1$. Then, because of (A.7), (A.8), and (A.10) and because Φ is strictly decreasing, we have $V(\mathbf{x}(1)) \le (1 - \frac{1}{\gamma})V(\mathbf{x}(0))$ and $t(1) = \Phi(\max_{i\in\mathcal{V}} \Delta V_i(\mathbf{x}(0))) \le \Phi(\frac{V(\mathbf{x}(0))}{\gamma})$. Next, let $k \ge 1$ and suppose $V(\mathbf{x}(k)) \le (1 - \frac{1}{\gamma})V(\mathbf{x}(k-1))$ and $t(k) \le \Phi(\frac{V(\mathbf{x}(k-1))}{\gamma})$. To show that $V(\mathbf{x}(k+1)) \le (1 - \frac{1}{\gamma})V(\mathbf{x}(k))$ and $t(k+1) \le \Phi(\frac{V(\mathbf{x}(k))}{\gamma})$, consider the following two cases: (i) $\max_{i\in\mathcal{V}} \Delta V_i(\mathbf{x}(k)) < \Phi^{-1}(t(k))$ and (ii) $\max_{i\in\mathcal{V}} \Delta V_i(\mathbf{x}(k)) \ge \Phi^{-1}(t(k))$. For case (i), due to (A.7), (A.9), and (A.10), we have $V(\mathbf{x}(k+1)) \le V(\mathbf{x}(k)) - \max_{i\in\mathcal{V}} \Delta V_i(\mathbf{x}(k)) \le (1 - \frac{1}{\gamma})V(\mathbf{x}(k))$ and $t(k+1) = \Phi(\max_{i\in\mathcal{V}} \Delta V_i(\mathbf{x}(k))) \le \Phi(\frac{V(\mathbf{x}(k))}{\gamma})$. For case (ii), due to (A.9), (A.10), and the hypothesis, we have $V(\mathbf{x}(k+1)) \le V(\mathbf{x}(k)) - \Phi^{-1}(t(k)) \le V(\mathbf{x}(k)) - \frac{V(\mathbf{x}(k-1))}{\gamma} \le V(\mathbf{x}(k)) - \frac{V(\mathbf{x}(k))}{\gamma(1-\frac{1}{\gamma})} \le (1 - \frac{1}{\gamma})V(\mathbf{x}(k))$ and $t(k+1) = t(k) \le \Phi(\frac{V(\mathbf{x}(k-1))}{\gamma}) \le \Phi(\frac{V(\mathbf{x}(k))}{\gamma(1-\frac{1}{\gamma})}) \le \Phi(\frac{V(\mathbf{x}(k))}{\gamma})$. This completes the proof by induction. It follows that (2.23) and therefore (2.24) and (2.25) hold, so that Theorems 2.2 and 2.3 hold verbatim here. Next, observe from (A.10) that $(t(k))_{k=0}^{\infty}$ is non-decreasing. To show that $\lim_{k\to\infty} t(k) = \infty$, assume to the contrary that $\exists \bar{t} \in (0,\infty)$ such that $t(k) \le \bar{t}\ \forall k \in \mathbb{N}$. For each $k \in \mathbb{P}$, reconsider the above two cases. Because of (A.9) and (A.10), for case (i),

$V(\mathbf{x}(k)) - V(\mathbf{x}(k+1)) \geq \max_{i \in \mathcal{V}} \Delta V_i(\mathbf{x}(k)) = \Phi^{-1}(t(k+1)) \geq \Phi^{-1}(\bar{t})$. Similarly, for case (ii), $V(\mathbf{x}(k)) - V(\mathbf{x}(k+1)) \geq \Phi^{-1}(t(k)) \geq \Phi^{-1}(\bar{t})$. Combining these two cases, we get $V(\mathbf{x}(k+1)) \leq V(\mathbf{x}(1)) - k\Phi^{-1}(\bar{t})\ \forall k \in \mathbb{N}$. Since $\Phi^{-1}(\bar{t}) > 0$, $V(\mathbf{x}(k+1)) < 0$ for sufficiently large k, which is a contradiction. Thus, $\lim_{k\to\infty} t(k) = \infty$. Finally, from the statement shown earlier by induction, we obtain $V(\mathbf{x}(k)) \leq (1-\frac{1}{\gamma}) \cdot \gamma\Phi^{-1}(t(k)) = (\gamma-1)\Phi^{-1}(t(k))\ \forall k \in \mathbb{P}$.

Appendix B Proofs for Chapter 3

Throughout the appendix, for any $x \in \mathbb{R}^n$ and any $P \in \mathbb{S}^n_+$, we write $x^T x$ and $x^T P x$ as $\|x\|^2$ and $\|x\|^2_P$, respectively. Moreover, for any $k \in \mathbb{N}$ and any nonempty $X \subset \mathcal{M}(k)$, let

$$z_X^k = (\sum_{i \in X} Q_i(k))^{-1} (\sum_{i \in X} Q_i(k) z_i(k)). \tag{B.1}$$

Then, from (3.16),

$$z_i(k) = z^{k-1}_{\mathcal{I}(k) \cup \mathcal{L}(k)}, \quad \forall k \in \mathbb{P},\ \forall i \in \mathcal{J}(k) \cup \mathcal{I}(k). \tag{B.2}$$

B.1 Proof of Lemma 3.1

Let $\mathcal{A}$ be given. Then, due to (3.11), (3.12), (3.13), (3.14), (3.15), and (B.2),

$$\begin{aligned}
V(k) - V(k-1) &= \sum_{i \in \mathcal{J}(k) \cup \mathcal{I}(k)} z_i(k)^T Q_i(k) z_i(k) - \sum_{i \in \mathcal{I}(k) \cup \mathcal{L}(k)} z_i(k-1)^T Q_i(k-1) z_i(k-1) \\
= -\Big(& \sum_{i \in \mathcal{I}(k) \cup \mathcal{L}(k)} z_i(k-1)^T Q_i(k-1) z_i(k-1) + \sum_{i \in \mathcal{J}(k) \cup \mathcal{I}(k)} (z^{k-1}_{\mathcal{I}(k) \cup \mathcal{L}(k)})^T Q_i(k) z^{k-1}_{\mathcal{I}(k) \cup \mathcal{L}(k)} \\
& - 2 \sum_{i \in \mathcal{J}(k) \cup \mathcal{I}(k)} z_i(k)^T Q_i(k) z^{k-1}_{\mathcal{I}(k) \cup \mathcal{L}(k)} \Big) \\
= -\Big(& \sum_{i \in \mathcal{I}(k) \cup \mathcal{L}(k)} z_i(k-1)^T Q_i(k-1) z_i(k-1) + \sum_{i \in \mathcal{I}(k) \cup \mathcal{L}(k)} (z^{k-1}_{\mathcal{I}(k) \cup \mathcal{L}(k)})^T Q_i(k-1) z^{k-1}_{\mathcal{I}(k) \cup \mathcal{L}(k)} \\
& - 2 \sum_{i \in \mathcal{I}(k) \cup \mathcal{L}(k)} z_i(k-1)^T Q_i(k-1) z^{k-1}_{\mathcal{I}(k) \cup \mathcal{L}(k)} \Big)
\end{aligned}$$

$$= - \sum_{i \in \mathcal{I}(k) \cup \mathcal{L}(k)} (z_i(k-1) - z^{k-1}_{\mathcal{I}(k) \cup \mathcal{L}(k)})^T Q_i(k-1)(z_i(k-1) - z^{k-1}_{\mathcal{I}(k) \cup \mathcal{L}(k)}), \quad \forall k \in \mathbb{P}. \tag{B.3}$$

Since the right-hand side of (B.3) is nonpositive, $(V(k))_{k=0}^{\infty}$ is non-increasing.

B.2 Proof of Proposition 3.1

Let $\mathcal{A}$ be given. Also, let $\hat{P}_i \in \mathbb{S}^n_+$, $\hat{q}_i \in \mathbb{R}^n$, $\forall i \in \mathcal{F}$ and $\tilde{P}_i \in \mathbb{S}^n_+$, $\tilde{q}_i \in \mathbb{R}^n$, $\forall i \in \mathcal{F}$ be any two sets of observations P_i's and q_i's, which, respectively, generate the state variables $\hat{Q}_i(k)$'s and $\tilde{Q}_i(k)$'s. To prove the proposition, we show that if the sequence $\{\hat{Q}_i(k)\}_{k \in \mathbb{N}, i \in \mathcal{M}(k)}$ is uniformly positive definite under $\mathcal{A}$, then so is the sequence $\{\tilde{Q}_i(k)\}_{k \in \mathbb{N}, i \in \mathcal{M}(k)}$. Note from (3.23), (3.20), and (3.21) that $\forall k \in \mathbb{N}$, $\forall i \in \mathcal{M}(k)$, $\hat{Q}_i(k) = \sum_{j \in \mathcal{F}} a_{ij}(k) \hat{P}_j$ and $\tilde{Q}_i(k) = \sum_{j \in \mathcal{F}} a_{ij}(k) \tilde{P}_j$, where each $a_{ij}(k) \geq 0$ is completely determined by $\mathcal{A}$. For each $j \in \mathcal{F}$, let $\hat{\lambda}_j > 0$ be the largest eigenvalue of $\hat{P}_j$ and $\tilde{\lambda}_j > 0$ be the smallest eigenvalue of $\tilde{P}_j$. Then, let $\hat{\lambda} = \max_{j \in \mathcal{F}}\{\hat{\lambda}_j\}$ and $\tilde{\lambda} = \min_{j \in \mathcal{F}}\{\tilde{\lambda}_j\}$. Since $\{\hat{Q}_i(k)\}_{k \in \mathbb{N}, i \in \mathcal{M}(k)}$ is uniformly positive definite under $\mathcal{A}$, $\exists \hat{\alpha} > 0$ such that $\forall k \in \mathbb{N}$, $\forall i \in \mathcal{M}(k)$, $\hat{Q}_i(k) > \hat{\alpha} I$. Thus, $\sum_{j \in \mathcal{F}} a_{ij}(k) > \frac{\hat{\alpha}}{\hat{\lambda}}$ $\forall k \in \mathbb{N}$ $\forall i \in \mathcal{M}(k)$. This, along with the fact that $\tilde{Q}_i(k) \geq \tilde{\lambda} \sum_{j \in \mathcal{F}} a_{ij}(k) I$ $\forall k \in \mathbb{N}$ $\forall i \in \mathcal{M}(k)$, implies that $\tilde{Q}_i(k) > \frac{\tilde{\lambda}\hat{\alpha}}{\hat{\lambda}} I$ $\forall k \in \mathbb{N}$ $\forall i \in \mathcal{M}(k)$. Since $\frac{\tilde{\lambda}\hat{\alpha}}{\hat{\lambda}} > 0$, $\{\tilde{Q}_i(k)\}_{k \in \mathbb{N}, i \in \mathcal{M}(k)}$ is uniformly positive definite under $\mathcal{A}$.

B.3 Proof of Theorem 3.1

Let $\mathcal{A}$ be given. From (3.25), $Q_i(k) \leq \sum_{i \in \mathcal{M}(0)} P_i$ $\forall k \in \mathbb{N}$ $\forall i \in \mathcal{M}(k)$. Thus, (3.28) holds, i.e., each $Q_i(k)$ is bounded. To derive (3.29), suppose the sequence $\{Q_i(k)\}_{k \in \mathbb{N}, i \in \mathcal{M}(k)}$ is uniformly positive definite under $\mathcal{A}$ and let $\alpha > 0$ be such that $Q_i(k) - \alpha I \in \mathbb{S}^n_+$ $\forall k \in \mathbb{N}$ $\forall i \in \mathcal{M}(k)$. Then, from (3.8) and

Lemma 3.1, $\alpha \sum_{i\in\mathcal{M}(k)} \|z_i(k) - z\|^2 \leq V(k) \leq V(0)\ \forall k \in \mathbb{N}$. Therefore, (3.29) is satisfied, i.e., each $z_i(k)$ is bounded.

B.4 Proof of Corollary 3.1

Suppose the membership dynamics are ultimately static under $\mathcal{A}$. Then, by Definition 3.3, $\exists k \in \mathbb{N}$ such that $\forall \ell > k$, $\mathcal{M}(\ell) = \mathcal{M}(k)$. Due to (3.23), (3.20), and (3.21), $Q_i(\ell) \in \mathbb{S}_+^n\ \forall \ell \in [0, k]\ \forall i \in \mathcal{M}(\ell)$. Due again to (3.20), $Q_i(\ell) \in \mathbb{S}_+^n\ \forall \ell \in [k+1, \infty)\ \forall i \in \mathcal{M}(\ell)$. Hence, $\{Q_i(k)\}_{k\in\mathbb{N}, i\in\mathcal{M}(k)}$ is uniformly positive definite under $\mathcal{A}$. It follows from Theorem 3.1 that (3.28) and (3.29) hold for some $\alpha > 0$.

B.5 Proof of Lemma 3.2

Let $\mathcal{A}$ be given. Suppose the agent network is connected under $\mathcal{A}$ at some time $k \in \mathbb{N}$, i.e., $h(k) < \infty$. Clearly, (3.30) holds for $h(k) = 0$. Now suppose $h(k) \geq 1$ and consider the following:

Lemma B.1. *For any $\ell \in \mathbb{N}$, any nonempty subset X of $\mathcal{M}(\ell)$, and any $\eta \in \mathbb{R}^n$, $\sum_{i\in X} \|z_X^\ell - \eta)\|_{Q_i(\ell)}^2 \leq \sum_{i\in X} \|z_i(\ell) - \eta\|_{Q_i(\ell)}^2$.*

Proof. Let $\ell \in \mathbb{N}$, $X \subset \mathcal{M}(\ell)$, $X \neq \emptyset$, and $\eta \in \mathbb{R}^n$ be given. Due to (B.1), we have $\sum_{i\in\mathcal{X}} Q_i(\ell) z_X^\ell = \sum_{i\in\mathcal{X}} Q_i(\ell) z_i(\ell)$, implying that $\sum_{i\in\mathcal{X}} (z_X^\ell)^T Q_i(\ell)\eta = \sum_{i\in\mathcal{X}} z_i(\ell)^T Q_i(\ell)\eta$ and $\sum_{i\in\mathcal{X}} (z_X^\ell)^T Q_i(\ell) z_X^\ell = \sum_{i\in\mathcal{X}} z_i(\ell)^T Q_i(\ell) z_X^\ell$. Because of these two properties,

$$\sum_{i\in X} |z_X^\ell - \eta\|_{Q_i(\ell)}^2 - \sum_{i\in X} \|z_i(\ell) - \eta\|_{Q_i(\ell)}^2 = -\sum_{i\in X} \|z_i(\ell) - z_X^\ell\|_{Q_i(\ell)}^2 \leq 0.$$

□

Lemma B.2. *For any $\ell \in \mathbb{N}$, any nonempty subset X of $\mathcal{M}(\ell)$, and any $\eta \in \mathbb{R}^n$, $\sum_{i\in X} \|z_i(\ell) - z_X^\ell\|^2_{Q_i(\ell)} \leq \sum_{i\in X} \|z_i(\ell) - \eta\|^2_{Q_i(\ell)}$.*

Proof. Let $\ell \in \mathbb{N}$, $X \subset \mathcal{M}(\ell)$, $X \neq \emptyset$, and $\eta \in \mathbb{R}^n$ be given. Using the two properties in the proof of Lemma B.1, we have

$$\sum_{i\in X} \|z_i(\ell) - z_X^\ell\|^2_{Q_i(\ell)} - \sum_{i\in X} \|z_i(\ell) - \eta\|^2_{Q_i(\ell)} = -\sum_{i\in X} \|z_X^\ell - \eta\|^2_{Q_i(\ell)} \leq 0.$$

□

Let $\alpha > 0$ be such that $Q_i(\ell) - \alpha I \in \mathbb{S}^n_+$ $\forall \ell \in [k, k+h(k)]$ $\forall i \in \mathcal{M}(\ell)$. This, along with (3.28), implies that

$$\alpha I < Q_i(\ell) \leq \beta I, \quad \forall \ell \in [k, k+h(k)],\ \forall i \in \mathcal{M}(\ell). \tag{B.4}$$

Assume, to the contrary, that (3.30) does not hold, i.e.,

$$V(k+h(k)) > \frac{(\frac{4\beta}{\alpha})^{M-1} \cdot M \cdot M!}{(\frac{4\beta}{\alpha})^{M-1} \cdot M \cdot M! + 1} V(k).$$

For convenience, let $\epsilon > 0$ be given by

$$\epsilon = \frac{V(k)}{(\frac{4\beta}{\alpha})^{M-1} \cdot M \cdot M! + 1}. \tag{B.5}$$

Then, $V(k) - V(k+h(k)) \leq \epsilon$. It follows from Lemma 3.1 that

$$V(\ell-1) - V(\ell) \leq \epsilon, \quad \forall \ell \in [k+1, k+h(k)]. \tag{B.6}$$

Due to (B.6), (B.3), and (B.4),

$$\|z_i(\ell-1) - z^{\ell-1}_{\mathcal{I}(\ell)\cup\mathcal{L}(\ell)}\|^2 \leq \frac{\epsilon}{\alpha}, \quad \forall \ell \in [k+1, k+h(k)],\ \forall i \in \mathcal{I}(\ell) \cup \mathcal{L}(\ell). \tag{B.7}$$

Next, let $d_i(\ell) = \sum_{j\in\mathcal{C}_i(k,\ell)} \|z_j(\ell) - z^\ell_{\mathcal{C}_i(k,\ell)}\|^2_{Q_j(\ell)}$ $\forall \ell \geq k$ $\forall i \in \mathcal{M}(\ell)$. In addition, let $m(\ell)$ be the number of distinct sets in the collection $\{\mathcal{C}_i(k,\ell)\}_{i\in\mathcal{M}(\ell)}$ $\forall \ell \geq k$. Notice from (3.26) and (3.27) that $1 \leq m(\ell) \leq |\mathcal{M}(\ell)| \leq M$ $\forall \ell \geq k$ and $m(\ell) \leq m(\ell-1)$ $\forall \ell \geq k+1$. Moreover, let $B(\ell) = \{k\} \cup \{k' \in [k+1, \ell] : m(k') < m(k'-1)\}$ $\forall \ell \geq k+1$. Then, consider the following lemma:

Lemma B.3. *For each $\ell \in [k, k+h(k)]$,*

$$d_i(\ell) \le (\frac{4\beta}{\alpha})^{M-m(\ell)}(M+1-m(\ell))\Big(\prod_{k'\in B(\ell)}(M+1-m(k'))\Big)\epsilon, \quad \forall i \in \mathcal{M}(\ell). \tag{B.8}$$

Proof. By induction over $\ell \in [k, k+h(k)]$. Let $\ell = k$. For any $i \in \mathcal{M}(\ell)$, from (3.26), $\mathcal{C}_i(k,\ell) = \{i\}$, which, together with (B.1), implies that $z_i(\ell) = z^{\ell}_{\mathcal{C}_i(k,\ell)}$. Hence, $d_i(\ell) = 0$ $\forall i \in \mathcal{M}(\ell)$. Since the right-hand side of (B.8) is positive, (B.8) holds for $\ell = k$. Next, let $\ell \in [k+1, k+h(k)]$ and suppose

$$d_i(\ell-1) \le (\frac{4\beta}{\alpha})^{M-m(\ell-1)}(M+1-m(\ell-1))\Big(\prod_{k'\in B(\ell-1)}(M+1-m(k'))\Big)\epsilon, \quad \forall i \in \mathcal{M}(\ell-1). \tag{B.9}$$

Below, we show that (B.9) implies (B.8). To do so, consider the following two mutually exclusive and exhaustive cases:

Case (I): $\mathcal{I}(\ell) \cup \mathcal{L}(\ell) \subset \mathcal{C}_{i^*}(k,\ell-1)$ for some $i^* \in \mathcal{M}(\ell-1)$. Due to (3.27),

$$m(\ell) = m(\ell-1), \tag{B.10}$$

implying that

$$B(\ell) = B(\ell-1). \tag{B.11}$$

Let $i \in \mathcal{M}(\ell)$. Suppose $i \in \mathcal{M}(\ell) - (\mathcal{C}_{i^*}(k,\ell-1) \cup \mathcal{J}(\ell))$. Then, due to (3.27), (3.19), (3.20), and (3.21), $\mathcal{C}_i(k,\ell) = \mathcal{C}_i(k,\ell-1)$, $z_j(\ell) = z_j(\ell-1)$ $\forall j \in \mathcal{C}_i(k,\ell)$, and $Q_j(\ell) = Q_j(\ell-1)$ $\forall j \in \mathcal{C}_i(k,\ell)$, implying that $d_i(\ell) = d_i(\ell-1)$. Now suppose $i \in (\mathcal{C}_{i^*}(k,\ell-1) \cup \mathcal{J}(\ell)) - \mathcal{L}(\ell)$. From (3.27), $\mathcal{C}_i(k,\ell) = (\mathcal{C}_{i^*}(k,\ell-1) \cup \mathcal{J}(\ell)) - \mathcal{L}(\ell)$. Thus, from (3.15), (3.20), (3.21), (3.14), and (3.19), we have $\sum_{j\in\mathcal{C}_{i^*}(k,\ell-1)} Q_j(\ell-1) = \sum_{j\in\mathcal{C}_i(k,\ell)} Q_j(\ell)$ and $\sum_{j\in\mathcal{C}_{i^*}(k,\ell-1)} Q_j(\ell-1)z_j(\ell-1) =$

$\sum_{j\in\mathcal{C}_i(k,\ell)} Q_j(\ell)z_j(\ell)$. These and (B.1) indicate that $z^{\ell}_{\mathcal{C}_i(k,\ell)} = z^{\ell-1}_{\mathcal{C}_{i^*}(k,\ell-1)}$. It follows from (B.2), (3.15), (3.19), (3.20), (3.21), and Lemma B.1 that

$$\begin{aligned}
d_i(\ell) &= \sum_{j\in\mathcal{J}(\ell)\cup\mathcal{I}(\ell)} \|z_i(\ell) - z^{\ell-1}_{\mathcal{C}_{i^*}(k,\ell-1)}\|^2_{Q_j(\ell)} + \sum_{\substack{j\in\mathcal{C}_{i^*}(k,\ell-1)\\ -(\mathcal{I}(\ell)\cup\mathcal{L}(\ell))}} \|z_i(\ell) - z^{\ell-1}_{\mathcal{C}_{i^*}(k,\ell-1)}\|^2_{Q_j(\ell)} \\
&= \sum_{j\in\mathcal{I}(\ell)\cup\mathcal{L}(\ell)} \|z^{\ell-1}_{\mathcal{I}(\ell)\cup\mathcal{L}(\ell)} - z^{\ell-1}_{\mathcal{C}_{i^*}(k,\ell-1)}\|^2_{Q_j(\ell-1)} + \sum_{\substack{j\in\mathcal{C}_{i^*}(k,\ell-1)\\ -(\mathcal{I}(\ell)\cup\mathcal{L}(\ell))}} \|z_j(\ell-1) - z^{\ell-1}_{\mathcal{C}_{i^*}(k,\ell-1)}\|^2_{Q_j(\ell-1)} \\
&\le \sum_{j\in\mathcal{I}(\ell)\cup\mathcal{L}(\ell)} \|z_j(\ell-1) - z^{\ell-1}_{\mathcal{C}_{i^*}(k,\ell-1)}\|^2_{Q_j(\ell-1)} + \sum_{\substack{j\in\mathcal{C}_{i^*}(k,\ell-1)\\ -(\mathcal{I}(\ell)\cup\mathcal{L}(\ell))}} \|z_j(\ell-1) - z^{\ell-1}_{\mathcal{C}_{i^*}(k,\ell-1)}\|^2_{Q_j(\ell-1)} \\
&= d_{i^*}(\ell-1).
\end{aligned}$$

It follows from (B.9), (B.10), and (B.11) that (B.8) holds for Case (I).

Case (II): $\mathcal{I}(\ell)\cup\mathcal{L}(\ell) \not\subset \mathcal{C}_i(k,\ell-1)\ \forall i\in\mathcal{M}(\ell-1)$. Due to (3.27), $m(\ell) < m(\ell-1)$, which implies that

$$m(\ell-1) - m(\ell) \ge 1, \tag{B.12}$$

$$B(\ell) = B(\ell-1)\cup\{\ell\}. \tag{B.13}$$

Let $i\in\mathcal{M}(\ell)$. Suppose $i\in\mathcal{M}(\ell) - (\cup_{j\in\mathcal{I}(\ell)\cup\mathcal{L}(\ell)}\mathcal{C}_j(k,\ell-1)\cup\mathcal{J}(\ell))$. Then, observe from (3.27), (3.19), (3.20), and (3.21) that $\mathcal{C}_i(k,\ell) = \mathcal{C}_i(k,\ell-1)$, $z_j(\ell) = z_j(\ell-1)\ \forall j\in\mathcal{C}_i(k,\ell)$, and $Q_j(\ell) = Q_j(\ell-1)\ \forall j\in\mathcal{C}_i(k,\ell)$. Hence, $d_i(\ell) = d_i(\ell-1)$, which, along with (B.9), (B.12), and (B.13), implies that $d_i(\ell) \le (\frac{4\beta}{\alpha})^{M-m(\ell)}(M+1-m(\ell))\Big(\prod_{k'\in B(\ell)}(M+1-m(k'))\Big)\epsilon$. Now suppose $i\in(\cup_{j\in\mathcal{I}(\ell)\cup\mathcal{L}(\ell)}\mathcal{C}_j(k,\ell-1)\cup\mathcal{J}(\ell)) - \mathcal{L}(\ell)$. Also suppose $\{\mathcal{C}_j(k,\ell-1)\}_{j\in\mathcal{I}(\ell)\cup\mathcal{L}(\ell)} = \{\mathcal{C}_{j_1}(k,\ell-1), \mathcal{C}_{j_2}(k,\ell-1), \ldots, \mathcal{C}_{j_p}(k,\ell-1)\}$, where, due to (3.27), $2\le p\le m(\ell-1)$ and $j_q\in\mathcal{C}_{j_q}(k,\ell-1)\subset\cup_{j\in\mathcal{I}(\ell)\cup\mathcal{L}(\ell)}\mathcal{C}_j(k,\ell-1)\ \forall q\in\{1,2,\ldots,p\}$. Then, from (3.27),

$$\mathcal{C}_i(k,\ell) = \Big(\cup_{q=1}^{p}\mathcal{C}_{j_q}(k,\ell-1)\cup\mathcal{J}(\ell)\Big) - \mathcal{L}(\ell). \tag{B.14}$$

Let $s_q \in \mathcal{C}_{j_q}(k, \ell-1) \cap (\mathcal{I}(\ell) \cup \mathcal{L}(\ell))\ \forall q \in \{1, 2, \ldots, p\}$. Then, because of (B.4), (B.14), Lemma B.2, (3.19), (B.2), the triangle inequality, (B.7), (B.9), (B.12), and (B.13),

$$
\begin{aligned}
d_i(\ell) &\leq \beta \sum_{\substack{j \in (\cup_{q=1}^{p} \mathcal{C}_{j_q}(k,\ell-1) \\ \cup \mathcal{J}(\ell)) - \mathcal{L}(\ell)}} \|z_j(\ell) - z^{\ell-1}_{\mathcal{I}(\ell)\cup\mathcal{L}(\ell)}\|^2 = \beta \sum_{q=1}^{p} \sum_{\substack{j \in \mathcal{C}_{j_q}(k,\ell-1) \\ -(\mathcal{I}(\ell)\cup\mathcal{L}(\ell))}} \|z_j(\ell-1) - z^{\ell-1}_{\mathcal{I}(\ell)\cup\mathcal{L}(\ell)}\|^2 \\
&\leq \beta \sum_{q=1}^{p} \sum_{\substack{j \in \mathcal{C}_{j_q}(k,\ell-1) \\ -(\mathcal{I}(\ell)\cup\mathcal{L}(\ell))}} (\|z_j(\ell-1) - z_{s_q}(\ell-1)\| + \|z_{s_q}(\ell-1) - z^{\ell-1}_{\mathcal{I}(\ell)\cup\mathcal{L}(\ell)}\|)^2 \\
&\leq \beta \sum_{q=1}^{p} \sum_{\substack{j \in \mathcal{C}_{j_q}(k,\ell-1) \\ -(\mathcal{I}(\ell)\cup\mathcal{L}(\ell))}} 2\Big((\|z_j(\ell-1) - z^{\ell-1}_{\mathcal{C}_{j_q}(k,\ell-1)}\| + \|z^{\ell-1}_{\mathcal{C}_{j_q}(k,\ell-1)} - z_{s_q}(\ell-1)\|)^2 \\
&\quad + \|z_{s_q}(\ell-1) - z^{\ell-1}_{\mathcal{I}(\ell)\cup\mathcal{L}(\ell)}\|^2\Big) \\
&\leq \beta \sum_{q=1}^{p} \sum_{\substack{j \in \mathcal{C}_{j_q}(k,\ell-1) \\ -(\mathcal{I}(\ell)\cup\mathcal{L}(\ell))}} 2\Big(2(\|z_j(\ell-1) - z^{\ell-1}_{\mathcal{C}_{j_q}(k,\ell-1)}\|^2 + \|z_{s_q}(\ell-1) - z^{\ell-1}_{\mathcal{C}_{j_q}(k,\ell-1)}\|^2) + \frac{\epsilon}{\alpha}\Big) \\
&\leq \beta \sum_{q=1}^{p} \sum_{\substack{j \in \mathcal{C}_{j_q}(k,\ell-1) \\ -(\mathcal{I}(\ell)\cup\mathcal{L}(\ell))}} 2(\frac{2}{\alpha} d_{j_q}(\ell-1) + \frac{\epsilon}{\alpha}) \\
&\leq |C_i(k,\ell)| \Big((\frac{4\beta}{\alpha})^{M-m(\ell-1)+1} (M+1-m(\ell-1)) \Big(\prod_{k' \in B(\ell-1)} (M+1-m(k')) \Big) \epsilon + \frac{2\beta}{\alpha}\epsilon \Big) \\
&\leq |C_i(k,\ell)| (\frac{4\beta}{\alpha})^{M-m(\ell)} (M+1-m(\ell)) \Big(\prod_{k' \in B(\ell-1)} (M+1-m(k')) \Big) \epsilon \\
&= |C_i(k,\ell)| (\frac{4\beta}{\alpha})^{M-m(\ell)} \Big(\prod_{k' \in B(\ell)} (M+1-m(k')) \Big) \epsilon.
\end{aligned}
$$

This, along with the fact that $|\mathcal{C}_i(k,\ell)| \leq M+1-m(\ell)$, implies that $d_i(\ell) \leq (\frac{4\beta}{\alpha})^{M-m(\ell)}(M+1-m(\ell))\Big(\prod_{k'\in B(\ell)}(M+1-m(k'))\Big)\epsilon$. Therefore, (B.8) holds for Case (II). □

Since $\mathcal{C}_i(k, k+h(k)) = \mathcal{M}(k+h(k))\ \forall i \in \mathcal{M}(k+h(k))$, we have $m(k+h(k)) = 1$. Also, note that $\Pi_{k'\in B(k+h(k))}(M+1-m(k')) \leq M!$. Furthermore, note from (3.5), (3.24), and (3.25) that $z = z^{\ell}_{\mathcal{M}(\ell)}$ $\forall \ell \in \mathbb{N}$, implying that $d_i(k + h(k)) = V(k + h(k))$ $\forall i \in \mathcal{M}(k + h(k))$. It follows from Lemma B.3 and (B.5) that $V(k+h(k)) \leq (\frac{4\beta}{\alpha})^{M-1} \cdot M \cdot M! \cdot \epsilon \leq \frac{(\frac{4\beta}{\alpha})^{M-1} \cdot M \cdot M!}{(\frac{4\beta}{\alpha})^{M-1} \cdot M \cdot M! + 1} V(k)$, which contradicts the assumption that (3.30) is violated. Therefore, (3.30) must hold.

B.6 Proof of Theorem 3.2

Let $\mathcal{A}$ be given. Suppose the agent network is connected under $\mathcal{A}$, i.e., $h(k) < \infty$ $\forall k \in \mathbb{N}$, and suppose the sequence $\{Q_i(k)\}_{k\in\mathbb{N}, i\in\mathcal{M}(k)}$ is uniformly positive definite under $\mathcal{A}$. Let $\alpha > 0$ be such that $Q_i(k) - \alpha I \in \mathbb{S}^n_+$ $\forall k \in \mathbb{N}$ $\forall i \in \mathcal{M}(k)$. Then, (3.31) holds if and only if $\lim_{k\to\infty} V(k) = 0$. Hence, we only need to show $\lim_{k\to\infty} V(k) = 0$. From (3.8) and Lemma 3.1, $(V(k))_{k=0}^{\infty}$ is nonnegative and non-increasing. Thus, $\exists c \geq 0$ such that $\lim_{k\to\infty} V(k) = c$. To show that c must be zero, assume, to the contrary, that $c > 0$. Let $\epsilon = \frac{c}{(\frac{4\beta}{\alpha})^{M-1} \cdot M \cdot M!}$. Then, $\exists k \in \mathbb{N}$ such that $c \leq V(\ell) < c + \epsilon$ $\forall \ell \geq k$. However, by Lemma 3.2, we have $V(k+h(k)) < \frac{(\frac{4\beta}{\alpha})^{M-1} \cdot M \cdot M!}{(\frac{4\beta}{\alpha})^{M-1} \cdot M \cdot M! + 1}(c+\epsilon) = c$, which contradicts the inequality $c \leq V(\ell)$. Therefore, $c = 0$, i.e., $\lim_{k\to\infty} V(k) = 0$, implying that (3.31) holds.

B.7 Proof of Corollary 3.2

Suppose the agent network is connected under $\mathcal{A}$ and suppose the membership dynamics are ultimately static under $\mathcal{A}$. As is shown in the proof of Corollary 3.1 in B.4, $\{Q_i(k)\}_{k\in\mathbb{N}, i\in\mathcal{M}(k)}$ is uniformly positive definite under $\mathcal{A}$. Thus, from Theorem 3.2, (3.31) holds.

B.8 Proof of Theorem 3.3

Let $\mathcal{A}$ be given. Suppose the agent network is uniformly connected under $\mathcal{A}$, i.e., $h^* < \infty$, and suppose the sequence $\{Q_i(k)\}_{k\in\mathbb{N}, i\in\mathcal{M}(k)}$ is uniformly positive definite under $\mathcal{A}$. Let $\alpha > 0$ be such that $Q_i(k) - \alpha I \in \mathbb{S}_+^n$ $\forall k \in \mathbb{N}$ $\forall i \in \mathcal{M}(k)$. Then, it follows from Lemmas 3.1 and 3.2 that $\forall \ell \in \mathbb{N}$, $V((\ell + 1)h^*) \leq V(\ell h^* + h(\ell h^*)) \leq \frac{(\frac{4\beta}{\alpha})^{M-1}\cdot M \cdot M!}{(\frac{4\beta}{\alpha})^{M-1}\cdot M\cdot M!+1} V(\ell h^*)$, which implies that $V(\ell h^*) \leq \left(\frac{(\frac{4\beta}{\alpha})^{M-1}\cdot M\cdot M!}{(\frac{4\beta}{\alpha})^{M-1}\cdot M\cdot M!+1}\right)^{\ell} V(0)$. Also, from (3.8), $\alpha\|z_j(\ell h^*) - z\|^2 \leq V(\ell h^*)$ $\forall \ell \in \mathbb{N}$ $\forall j \in \mathcal{M}(\ell h^*)$. It follows that (3.32) holds.

B.9 Proof of Corollary 3.3

Suppose the agent network is uniformly connected under $\mathcal{A}$ and suppose the membership dynamics are ultimately static under $\mathcal{A}$. As is shown in the proof of Corollary 3.1 in B.4, $\{Q_i(k)\}_{k\in\mathbb{N}, i\in\mathcal{M}(k)}$ is uniformly positive definite under $\mathcal{A}$. Thus, from Theorem 3.3, (3.32) holds for some $\alpha > 0$.

Appendix C Proofs for Chapter 4

C.1 Proof of Theorem 4.1

In this section, we use $\mathbf{P}(A)$ to represent the probability of event A. Suppose PE is utilized. Then, for any $k \in \mathbb{P}$, $\mathcal{I}(k) = \{i, j\} \in \mathcal{E}$ with probability $\frac{1}{L}$, and $\mathcal{J}(k) = \mathcal{L}(k) = \emptyset$. Thus, $\forall \{i, j\} \in \mathcal{E}$, $\forall k \in \mathbb{P}$,

$$\mathbf{P}(\mathcal{I}(\ell) \neq \{i, j\},\ \forall \ell \geq k) = \Pi_{\ell=k}^{\infty}(1 - \frac{1}{L}) = 0. \tag{C.1}$$

For any given $\{i, j\} \in \mathcal{E}$, consider the following probability:

$$\begin{aligned}
&\mathbf{P}(\{i, j\} \text{ is selected to be } \mathcal{I}(k) \text{ infinitely many times}) \\
&\quad = 1 - \mathbf{P}(\{i, j\} \text{ is selected to be } \mathcal{I}(k) \text{ finite times}) \\
&\quad = 1 - \mathbf{P}(\cup_{k=1}^{\infty}\{\mathcal{I}(\ell) \neq \{i, j\},\ \forall \ell \geq k\}) \\
&\quad \geq 1 - \sum_{k=1}^{\infty} \mathbf{P}(\mathcal{I}(\ell) \neq \{i, j\},\ \forall \ell \geq k).
\end{aligned}$$

This, along with (C.1), implies that

$$\mathbf{P}(\{i, j\} \text{ is selected to be } \mathcal{I}(k) \text{ infinitely many times}) \geq 1.$$

Since any probability cannot exceed 1,

$$\mathbf{P}(\{i, j\} \text{ is selected to be } \mathcal{I}(k) \text{ infinitely many times}) = 1.$$

Then, because the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is connected, we know that with probability 1, the network is connected under $\mathcal{A}$, by Definition 1 in Chapter 3. Since

$A_i \in \mathbb{S}_+^n$, $\forall i \in \mathcal{V}$, it follows from Theorem 2 in Chapter 3 that with probability 1, $\lim_{k\to\infty} \hat{x}_i(k) = x$, $\forall i \in \mathcal{V}$.

C.2 Proof of Theorem 4.2

This proof is similar to that of Theorem 4.1.

C.3 Proof of Theorem 4.3

This proof also is similar to that of Theorem 4.1.

C.4 Proof of Theorem 4.4

From (4.9) and Lemma 1 in Chapter 3, the sequence $\{V(\mathbf{x}(k))\}_{k=0}^{\infty}$ is nonnegative and non-increasing. Thus, $\exists c \geq 0$ such that $\lim_{k\to\infty} V(\mathbf{x}(k)) = c$. To show that c must be zero, assume, to the contrary, that $c > 0$. Let

$$\epsilon = \frac{c}{\frac{4\beta}{\alpha} N^2 L}, \tag{C.2}$$

where α and β are defined in (4.16). Then, $\exists K \in \mathbb{N}$ such that

$$c \leq V(\mathbf{x}(k)) < c + \epsilon, \quad \forall k \geq K, \tag{C.3}$$

implying that

$$V(\mathbf{x}(k)) - V(\mathbf{x}(k+1)) < \epsilon, \quad \forall k \geq K. \tag{C.4}$$

With ICHE, there are N candidates for $\mathcal{I}(k)$, $\forall k \in \mathbb{P}$, denoted as $\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_N$ and defined as $\mathcal{I}_i = \{\{i, j\} \in \mathcal{E}\}$, so that at iteration k, if node i is selected, then $\mathcal{I}(k) = \mathcal{I}_i$.

Now suppose that $\mathcal{I}(K+1) = \mathcal{I}_p$. Then from (4.10) and (C.4),

$$\begin{aligned}
&\epsilon > V(\mathbf{x}(K)) - V(\mathbf{x}(K+1)) = \sum_{\{p,j\}\in\mathcal{I}_p} \|x_{\{p,j\}}(K) - \hat{x}_p(K)\|^2_{A_{\{p,j\}}} \\
&\quad \geq \sum_{\{q,j\}\in\mathcal{I}_q} \|x_{\{q,j\}}(K) - \hat{x}_q(K)\|^2_{A_{\{q,j\}}}, \quad \forall q \in \mathcal{V}.
\end{aligned}$$

Then $\forall q \in \mathcal{V}, \forall i \in \mathcal{N}_q$, we have $\|x_{\{q,i\}}(K) - \hat{x}_q(K)\|^2 < \frac{\epsilon}{\alpha}$, and from the triangle inequality, $\|x_{\{q,i\}}(K) - x_{\{q,j\}}(K)\| < 2\sqrt{\frac{\epsilon}{\alpha}}$, $\forall i, j \in \mathcal{N}_q$, implying that

$$\|x_{\{i,j\}}(K) - x_{\{p,q\}}(K)\| \leq 2(N-2)\sqrt{\frac{\epsilon}{\alpha}}, \ \forall \{i,j\}, \{p,q\} \in \mathcal{E}. \tag{C.5}$$

Let $\{p,q\} \in \mathcal{E}$. From Lemma 4 in Chapter 3,

$$\begin{aligned}
V(\mathbf{x}(K)) &\leq \sum_{\{i,j\}\in\mathcal{E}} \|x_{\{i,j\}}(K) - x_{\{p,q\}}(K)\|^2_{A_{\{i,j\}}} \\
&\leq \beta \sum_{\{i,j\}\in\mathcal{E}\setminus\{p,q\}} \|x_{\{i,j\}}(K) - x_{\{p,q\}}(K)\|^2.
\end{aligned}$$

Then, from (C.5), we obtain

$$V(\mathbf{x}(K)) \leq \frac{4\beta}{\alpha}(N-2)^2(L-1)\epsilon. \tag{C.6}$$

Substituting (C.2) into the right-hand side of (C.6) gives $V(K) < c$, which is a contradiction to (C.3). Therefore, we must have $\lim_{k\to\infty} V(\mathbf{x}(k)) = 0$. This, along with the fact that $A_{\{i,j\}} \in \mathbb{S}^n_+$, $\forall \{i,j\} \in \mathcal{E}$, implies that $\lim_{k\to\infty} x_{\{i,j\}}(k) = x$, $\forall \{i,j\} \in \mathcal{E}$. Finally, due to (4.8), $\lim_{k\to\infty} \hat{x}_i(k) = x$, $\forall i \in \mathcal{V}$.

C.5 Proof of Theorem 4.5

With ICHE, for any $k \in \mathbb{P}$, we have

$$V(\mathbf{x}(k-1)) - V(\mathbf{x}(k)) = \max_{i\in\mathcal{V}} \Delta V_i(\mathbf{x}(k-1)). \tag{C.7}$$

Let $V(\mathbf{x}(k-1))$ be given and suppose $V(\mathbf{x}(k-1)) = d \neq 0$. Then, (C.7) can be rewritten as follows:

$$V(\mathbf{x}(k)) = V(\mathbf{x}(k-1)) \cdot \Big(1 - \frac{1}{d} \max_{i \in \mathcal{V}} \Delta V_i(\mathbf{x}(k-1))\Big),$$

implying that $V(\mathbf{x}(k)) \leq \rho V(\mathbf{x}(k-1))$, where

$$\rho = 1 - \frac{1}{d} \min_{\mathbf{x}(k-1)} \max_{i \in \mathcal{V}} \Delta V_i(\mathbf{x}(k-1)).$$

Let $\omega_1 = d(1-\rho)$. Then, ω_1 is the optimal value of the following problem:

$$\begin{array}{ll} \text{minimize}_{\mathbf{y} \in \mathbb{R}^{Ln}} & \max_{i \in \mathcal{V}} \Delta V_i(\mathbf{y}) \\ \text{subject to} & \Big(\sum\limits_{\{i,j\} \in \mathcal{E}} A_{\{i,j\}}\Big)^{-1} \Big(\sum\limits_{\{i,j\} \in \mathcal{E}} A_{\{i,j\}} y_{\{i,j\}}\Big) = x \\ & V(\mathbf{y}) = d, \end{array} \tag{C.8}$$

where x is defined in (4.1).

Next, we prove the following lemma.

Lemma C.1. *Problem* (C.8) *is equivalent to the following problem:*

$$\begin{array}{ll} \text{minimize}_{\mathbf{z} \in \mathbb{R}^{Ln}} & \max_{i \in \mathcal{V}} \Delta V_i(\mathbf{z}) \\ \text{subject to} & \Big(\sum\limits_{\{i,j\} \in \mathcal{E}} A_{\{i,j\}}\Big)^{-1} \Big(\sum\limits_{\{i,j\} \in \mathcal{E}} A_{\{i,j\}} z_{\{i,j\}}\Big) = 0 \\ & V(\mathbf{z}) = 1. \end{array} \tag{C.9}$$

Moreover, if ω_2 is the optimal value of problem (C.9)*, then $\omega_1 = d\omega_2$.*

Proof. For any feasible point $\mathbf{y} \in \mathbb{R}^{Ln}$ of problem (C.8), let

$$z_{\{i,j\}} = \frac{1}{\sqrt{d}}(y_{\{i,j\}} - x), \quad \forall \{i,j\} \in \mathcal{E}.$$

Then, the first constraint in (C.8) can be rewritten as

$$\begin{aligned} 0 &= \Big(\sum_{\{i,j\} \in \mathcal{E}} A_{\{i,j\}}\Big)^{-1} \Big(\sum_{\{i,j\} \in \mathcal{E}} A_{\{i,j\}} \frac{1}{\sqrt{d}}(\mathbf{y}_{\{i,j\}} - x)\Big), \\ &= \Big(\sum_{\{i,j\} \in \mathcal{E}} A_{\{i,j\}}\Big)^{-1} \Big(\sum_{\{i,j\} \in \mathcal{E}} A_{\{i,j\}} z_{\{i,j\}}\Big), \end{aligned}$$

implying that $\mathbf{z}$ satisfies the first constraint in problem (C.9). Also, since $\frac{1}{d}V(\mathbf{y}) = \sum_{\{i,j\}\in\mathcal{E}} \|\frac{1}{\sqrt{d}}(y_{\{i,j\}} - x)\|^2_{A_{\{i,j\}}} = 1$, we have $V(\mathbf{z}) = 1$. Thus, $\mathbf{z}$ satisfies the two constraints in problem (C.9), implying that $\mathbf{z}$ is feasible for problem (C.9). On the other hand, observe that the converse is also true. Therefore, $\mathbf{z}$ is feasible for problem (C.9) if and only if $\mathbf{y}$ is feasible for problem (C.8). Furthermore,

$$\begin{aligned}
&\max_{i\in\mathcal{V}} \Delta V_i(\mathbf{y}) \\
&= d \max_{i\in\mathcal{V}} \sum_{j\in\mathcal{N}_i} \Big\| \frac{y_{\{i,j\}} - x}{\sqrt{d}} - (\sum_{j\in\mathcal{N}_i} A_{\{i,j\}})^{-1} (\sum_{j\in\mathcal{N}_i} A_{\{i,j\}} \frac{y_{\{i,j\}} - x}{\sqrt{d}}) \Big\|^2_{A_{\{i,j\}}} \\
&= d \max_{i\in\mathcal{V}} \Delta V_i(\mathbf{z}). \qquad \text{(C.10)}
\end{aligned}$$

Suppose $\mathbf{y}^*$ is the optimizer of problem (C.8). Then, for any feasible point $\mathbf{y}$ of problem (C.8), we have

$$\max_{i\in\mathcal{V}} \Delta V_i(\mathbf{y}^*) \leq \max_{i\in\mathcal{V}} \Delta V_i(\mathbf{y}). \qquad \text{(C.11)}$$

Let $z^*_{\{i,j\}} = \frac{1}{\sqrt{d}}(y^*_{\{i,j\}} - x)$, $\forall \{i,j\} \in \mathcal{E}$. Then, $\mathbf{z}^*$ is feasible for problem (C.9). Next, divide both sides of (C.11) by d. Then, from (C.10), we have $\max_{i\in\mathcal{V}} \Delta V_i(\mathbf{z}^*) \leq \max_{i\in\mathcal{V}} \Delta V_i(\mathbf{z})$. Since $\mathbf{z}$ can be any feasible point of problem (C.9), $\mathbf{z}^*$ is the optimizer of problem (C.9), implying that problems (C.8) and (C.9) are equivalent. From (C.10), we know that $\omega_1 = d\omega_2$. $\square$

Now we show that problems (C.8) and (4.14) are equivalent. From Lemma C.1, we know that $\min_{\mathbf{y}\in\mathbb{R}^{Ln}} \max_{i\in\mathcal{V}} \Delta V_i(\mathbf{y})$ is proportional to $V(\mathbf{y})$ and independent of x. Thus we can let $x = \mathbf{0}$. If we let $\max_{i\in\mathcal{V}} \Delta V_i(\mathbf{x}) = 1$, then problem (C.8) is converted to maximizing $V(\mathbf{x})$, while satisfying

$$\Big(\sum_{\{i,j\}\in\mathcal{E}} A_{\{i,j\}}\Big)^{-1} \Big(\sum_{\{i,j\}\in\mathcal{E}} A_{\{i,j\}} x_{\{i,j\}}\Big) = \mathbf{0},$$

which is problem (4.14). Note that the optimal value of problem (4.14) is the reciprocal of ω_2. Moreover, $\omega_2 = \frac{\omega_1}{d} = 1 - \rho$ by Lemma C.1. Hence, $\frac{1}{1-\rho}$ is the optimal value of problem (4.14).

Finally, from problem (C.9), we know that $\omega_2 \leq 1$ because V is always nonnegative. This implies that $\rho \geq 0$. Also, since $L \geq 2$, the optimal value of problem (4.14) is positive, implying that $\rho < 1$. Therefore, $0 \leq \rho < 1$.

C.6 Proof of Theorem 4.6

Let K be such that $\Delta V_i(K) \leq \gamma$, $\forall i \in \mathcal{V}$. Due to (4.10), $\|x_{\{i,j\}}(K) - \hat{x}_i(K)\| \leq \sqrt{\frac{\gamma}{\alpha}}$, $\forall i \in \mathcal{V}$, $\forall j \in \mathcal{N}_i$. By the triangle inequality, we have $\|\hat{x}_i(K) - \hat{x}_j(K)\| \leq 2\sqrt{\frac{\gamma}{\alpha}}$, $\forall \{i,j\} \in \mathcal{E}$. Also, notice that whenever $|\mathcal{N}_i| = 1$, $i \in \mathcal{V}$, we have $\|x_{\{i,j\}}(K) - \hat{x}_i(K)\| = 0$ and thus $\|\hat{x}_i(K) - \hat{x}_j(K)\| \leq \sqrt{\frac{\gamma}{\alpha}}$, $\forall j \in \mathcal{N}_i$. Therefore,

$$\|\hat{x}_i(K) - \hat{x}_j(K)\| \leq 2(N-2)\sqrt{\frac{\gamma}{\alpha}}, \quad \forall i,j \in \mathcal{V}. \tag{C.12}$$

Note that from (4.8), the solution x in (4.1) can be rewritten as follows:

$$\begin{aligned} x &= (2\sum_{\{i,j\}\in\mathcal{E}} A_{\{i,j\}})^{-1}(2\sum_{\{i,j\}\in\mathcal{E}} A_{\{i,j\}}x_{\{i,j\}}(K)) \\ &= (\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i} A_{\{i,j\}})^{-1}(\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i} A_{\{i,j\}}x_{\{i,j\}}(K)) \\ &= (\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i} A_{\{i,j\}})^{-1}(\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i} A_{\{i,j\}}\hat{x}_i(K)). \end{aligned} \tag{C.13}$$

Let $p \in \mathcal{V}$. From (C.12), (C.13), and Lemma 4 in Chapter 3, we have

$$\begin{aligned} \alpha'\sum_{i\in\mathcal{V}}\|\hat{x}_i(K) - x\|^2 &\leq \sum_{i\in\mathcal{V}}\|\hat{x}_i(K) - x\|^2_{\sum_{j\in\mathcal{N}_i}A_{\{i,j\}}} \leq \sum_{i\in\mathcal{V}}\|\hat{x}_i(K) - \hat{x}_p(K)\|^2_{\sum_{j\in\mathcal{N}_i}A_{\{i,j\}}} \\ &\leq \beta'\sum_{i\in\mathcal{V}-\{p\}}\|\hat{x}_i(K) - \hat{x}_p(K)\|^2 \leq \frac{4\beta'}{\alpha}(N-2)^2(N-1)\gamma, \end{aligned}$$

implying that (4.18) holds.

Appendix D Proofs for Chapter 5

D.1 Proof of Theorem 5.1

Suppose Assumption 5.1 holds and let $(u(k))_{k=1}^{\infty}$ satisfying Assumption 5.2 be given. Consider the following lemmas:

Lemma D.1. *Suppose Assumption 5.1 holds. Then, $\forall [a,b] \subset \mathcal{X}$, there exists a continuous and strictly increasing function $\gamma : [0,\infty) \to [0,\infty)$ satisfying $\gamma(0) = 0$ and $\lim_{d\to\infty} \gamma(d) = \infty$, such that $\forall \eta > 0$, $\forall i \in \mathcal{V}$, $\forall (x,y) \in [a,b]^2$, $f_i(y) - f_i(x) - f_i'(x)(y-x) \le \eta$ implies $|y-x| \le \gamma^{-1}(\eta)$.*

Proof. Let $[a,b] \subset \mathcal{X}$. For each $i \in \mathcal{V}$, define $g_i : [a,b]^2 \to \mathbb{R}$ as $g_i(x,y) = f_i(y) - f_i(x) - f_i'(x)(y-x)$. Due to Assumption 5.1 and (5.18), $g_i(x,y) \ge 0$ $\forall (x,y) \in [a,b]^2$, where the equality holds if and only if $x = y$. Moreover, since f_i' is strictly increasing and $g_i(x,y)$ can be written as $g_i(x,y) = \int_x^y (f_i'(t) - f_i'(x))dt$, $g_i(x,y)$ is strictly increasing with $|y-x|$ for each fixed $x \in [a,b]$. Furthermore, because f_i and f_i' are continuous, g_i is continuous. Next, for each $d \in [0, b-a]$, let $\mathcal{K}(d) = \{(x,y) \in [a,b]^2 : |y-x| = d\}$. Also, for each $i \in \mathcal{V}$, define $\gamma_i : [0, b-a] \to \mathbb{R}$ as $\gamma_i(d) = \min_{(x,y)\in\mathcal{K}(d)} g_i(x,y)$. Due to the compactness of $\mathcal{K}(d)$ $\forall d \in [0, b-a]$ and the continuity of g_i, γ_i is well-defined and continuous. In addition, since $g_i(x,y) = 0$ $\forall (x,y) \in \mathcal{K}(0)$, $\gamma_i(0) = 0$. Now pick any d_1 and d_2 such that $0 \le d_1 < d_2 \le b-a$. Let $(x_2,y_2) \in \mathcal{K}(d_2)$ be such that $\gamma_i(d_2) = g_i(x_2,y_2)$. If $y_2 > x_2$, then $y_2 - x_2 = d_2$. In this case, $\exists y_1 \in [x_2, y_2)$

such that $y_1 - x_2 = d_1$. Since $g_i(x_2, y)$ is strictly increasing with y for $y \geq x_2$, we have $\gamma_i(d_1) \leq g_i(x_2, y_1) < g_i(x_2, y_2) = \gamma_i(d_2)$. Similarly, if $y_2 < x_2$, we also have $\gamma_i(d_1) < \gamma_i(d_2)$. Hence, γ_i is strictly increasing. Finally, define $\gamma : [0, \infty) \to [0, \infty)$ as $\gamma(d) = \left\{ \begin{smallmatrix} \min_{i \in \mathcal{V}} \gamma_i(d) & \text{if } d \in [0, b-a] \\ \min_{i \in \mathcal{V}} \gamma_i(b-a) + d - (b-a) & \text{if } d \in (b-a, \infty) \end{smallmatrix} \right.$. Note that $\gamma(0) = 0$ since $\gamma_i(0) = 0 \ \forall i \in \mathcal{V}$, and that $\lim_{d \to \infty} \gamma(d) = \infty$. Moreover, since γ_i is continuous and strictly increasing $\forall i \in \mathcal{V}$, so is γ on $[0, b-a]$. Also, observe that γ is continuous and strictly increasing on $[b-a, \infty)$. Thus, γ is continuous and strictly increasing. Now let $\eta > 0$, $i \in \mathcal{V}$, and $(x, y) \in [a, b]^2$. Suppose $g_i(x, y) \leq \eta$. If $\eta \leq \gamma(b-a)$, then $|y - x| \leq \gamma^{-1}(\eta)$ because $\gamma(|y-x|) \leq \gamma_i(|y - x|) \leq g_i(x, y) \leq \eta$. If $\eta > \gamma(b - a)$, then $|y - x| \leq b - a < \gamma^{-1}(\eta)$. $\square$

Lemma D.2. *Suppose Assumption 5.1 holds. Then, $\forall [a, b] \subset \mathcal{X}$, $\exists \beta \in (0, \infty)$ such that $\forall i \in \mathcal{V}$, $\forall (x, y) \in [a, b]^2$, $f_i(y) - f_i(x) - f_i'(x)(y - x) \leq \beta |y - x|$.*

Proof. Let $[a, b] \subset \mathcal{X}$ and $\beta = 1 + 2 \max_{j \in \mathcal{V}} |f_j'(b)|$. Obviously, $\beta > 0$, and by Assumption 5.1, $\beta < \infty$. Let $i \in \mathcal{V}$ and $(x, y) \in [a, b]^2$. Since f_i is continuously differentiable, by the Mean Value Theorem, $\exists c$ between x and y such that $f_i(y) - f_i(x) = f_i'(c)(y - x)$. This, along with the triangle inequality and the fact that f_i' is strictly increasing, implies that $f_i(y) - f_i(x) - f_i'(x)(y - x) = (f_i'(c) - f_i'(x))(y - x) \leq |f_i'(c) - f_i'(x)| \cdot |y - x| \leq (|f_i'(c)| + |f_i'(x)|)|y - x| \leq 2|f_i'(b)| \cdot |y - x| \leq \beta |y - x|$. $\square$

Let $a = \min_{i \in \mathcal{V}} \hat{x}_i(0)$ and $b = \max_{i \in \mathcal{V}} \hat{x}_i(0)$. Then, it follows from Proposition 5.2 that $\hat{x}_i(k) \in [a, b] \subset \mathcal{X} \ \forall k \in \mathbb{N} \ \forall i \in \mathcal{V}$ and from (5.4) and Lemma 5.1 that $x^* \in [a, b]$. By Lemma D.1, there exists a continuous and strictly increasing function $\gamma : [0, \infty) \to [0, \infty)$ satisfying $\gamma(0) = 0$ and $\lim_{d \to \infty} \gamma(d) = \infty$, such that $\forall \eta > 0$, $\forall i \in \mathcal{V}$, $\forall (x, y) \in [a, b]^2$, $f_i(y) - f_i(x) - f_i'(x)(y - x) \leq \eta$ implies $|y - x| \leq \gamma^{-1}(\eta)$. Also, by Lemma D.2, $\exists \beta \in (0, \infty)$ such that $\forall i \in \mathcal{V}$,

$\forall (x,y) \in [a,b]^2$, $f_i(y) - f_i(x) - f_i'(x)(y-x) \leq \beta|y-x|$. From Lemma 5.2, $(V(\mathbf{x}(k)))_{k=0}^{\infty}$ is nonnegative and non-increasing. Thus, $\exists c \geq 0$ such that $\lim_{k\to\infty} V(\mathbf{x}(k)) = c$. To show that c must be zero, assume, to the contrary, that $c > 0$. Let $\epsilon > 0$ be given by $\epsilon = \gamma(\frac{c}{4\beta N^2})$. Then, $\exists k_1 \in \mathbb{N}$ such that

$$c \leq V(\mathbf{x}(k)) < c + \epsilon, \quad \forall k \geq k_1. \tag{D.1}$$

Due to (D.1), $V(\mathbf{x}(k-1)) - V(\mathbf{x}(k)) < \epsilon \ \forall k \geq k_1 + 1$. Hence, from (5.18) and (5.21), $f_i(\hat{x}_i(k)) - f_i(\hat{x}_i(k-1)) - f_i'(\hat{x}_i(k-1))(\hat{x}_i(k) - \hat{x}_i(k-1)) < \epsilon \ \forall k \geq k_1 + 1$ $\forall i \in u(k)$. As a result, $|\hat{x}_i(k) - \hat{x}_i(k-1)| \leq \gamma^{-1}(\epsilon) \ \forall k \geq k_1 + 1 \ \forall i \in u(k)$. Because of this and (5.14),

$$|\hat{x}_i(k) - \hat{x}_j(k)| \leq 2\gamma^{-1}(\epsilon), \quad \forall k \geq k_1, \ \forall i,j \in u(k+1). \tag{D.2}$$

Now suppose $\max_{i\in\mathcal{V}} \hat{x}_i(k_1) - \min_{i\in\mathcal{V}} \hat{x}_i(k_1) > 2(N-1)\gamma^{-1}(\epsilon)$. Then, $\exists p,q \in \mathcal{V}$ such that $\hat{x}_q(k_1) - \hat{x}_p(k_1) > 2\gamma^{-1}(\epsilon)$ and $\mathcal{C}_1 \cup \mathcal{C}_2 = \mathcal{V}$, where $\mathcal{C}_1 = \{i \in \mathcal{V} : \hat{x}_i(k_1) \leq \hat{x}_p(k_1)\}$ and $\mathcal{C}_2 = \{i \in \mathcal{V} : \hat{x}_i(k_1) \geq \hat{x}_q(k_1)\}$. Next, we show by induction that $\forall k \geq k_1$, $\hat{x}_i(k) \leq \hat{x}_p(k_1) \ \forall i \in \mathcal{C}_1$ and $\hat{x}_i(k) \geq \hat{x}_q(k_1) \ \forall i \in \mathcal{C}_2$. Clearly, the statement is true for $k = k_1$. For $k \geq k_1 + 1$, suppose $\hat{x}_i(k-1) \leq \hat{x}_p(k_1) \ \forall i \in \mathcal{C}_1$ and $\hat{x}_i(k-1) \geq \hat{x}_q(k_1) \ \forall i \in \mathcal{C}_2$. Then, due to (D.2), $\forall i \in \mathcal{C}_1$, $\forall j \in \mathcal{C}_2$, $\{i,j\} \neq u(k)$, i.e., $u(k) \subset \mathcal{C}_1$ or $u(k) \subset \mathcal{C}_2$. It follows from (5.17) and Lemma 5.1 that $\hat{x}_i(k) \leq \hat{x}_p(k_1) \ \forall i \in \mathcal{C}_1$ and $\hat{x}_i(k) \geq \hat{x}_q(k_1) \ \forall i \in \mathcal{C}_2$, completing the induction. Due again to (D.2), we have $\forall i \in \mathcal{C}_1$, $\forall j \in \mathcal{C}_2$, $\{i,j\} \neq u(k) \ \forall k \geq k_1 + 1$, which violates Assumption 5.2. Consequently, $\max_{i\in\mathcal{V}} \hat{x}_i(k_1) - \min_{i\in\mathcal{V}} \hat{x}_i(k_1) \leq 2(N-1)\gamma^{-1}(\epsilon)$. It follows from (5.4) and Lemma 5.1 that $|x^* - \hat{x}_i(k_1)| \leq \max_{j\in\mathcal{V}} \hat{x}_j(k_1) - \min_{j\in\mathcal{V}} \hat{x}_j(k_1) \leq 2(N-1)\gamma^{-1}(\epsilon)$ $\forall i \in \mathcal{V}$. Hence, $V(\mathbf{x}(k_1)) \leq \beta \sum_{i\in\mathcal{V}} |x^* - \hat{x}_i(k_1)| \leq \beta \cdot N \cdot 2(N-1)\gamma^{-1}(\epsilon) < c$, which contradicts (D.1). Therefore, $c = 0$, i.e., (5.20) holds, implying that (5.3) is satisfied.

D.2 Proof of Theorem 5.2

Suppose Assumption 5.1 holds and let the random sequence $(u(k))_{k=1}^{\infty}$ specified by $p_{\{i,j\}}(k)\ \forall k \in \mathbb{P}\ \forall \{i,j\} \in \mathcal{E}(k)$ satisfy Assumption 5.3. Let $\{i,j\} \in \tilde{\mathcal{E}}_{\infty}$. Then, from the definition of $\tilde{\mathcal{E}}_{\infty}$ in (5.23), $\exists \varepsilon > 0$ such that $\forall k \in \mathbb{P}$, $p_{\{i,j\}}(\ell) \geq \varepsilon$ for some $\ell > k$. For each $k \in \mathbb{P}$, let $A_{\{i,j\}}(k) = \{\ell \geq k : \{i,j\} \in \mathcal{E}(\ell)$ and $p_{\{i,j\}}(\ell) \geq \varepsilon\}$ and $B_{\{i,j\}}(k) = \{\ell \geq k : \{i,j\} \in \mathcal{E}(\ell)$ and $p_{\{i,j\}}(\ell) < \varepsilon\}$. Then,

$$
\begin{aligned}
\mathrm{P}\{u(\ell) \neq \{i,j\}\ \forall \ell \geq k\} &= \prod_{\substack{\ell=k \\ \{i,j\}\in\mathcal{E}(\ell)}}^{\infty} (1 - p_{\{i,j\}}(\ell)) \\
&= \prod_{\ell \in A_{\{i,j\}}(k)} (1 - p_{\{i,j\}}(\ell)) \cdot \prod_{\ell \in B_{\{i,j\}}(k)} (1 - p_{\{i,j\}}(\ell)) \\
&\leq \prod_{\ell \in A_{\{i,j\}}(k)} (1 - \varepsilon) \cdot \prod_{\ell \in B_{\{i,j\}}(k)} 1 = 0, \quad \forall k \in \mathbb{P},
\end{aligned}
$$

where the last step is due to $A_{\{i,j\}}(k)$ having infinitely many elements $\forall k \in \mathbb{P}$, as a result of (5.23). Hence,

$$
\mathrm{P}\{u(\ell) \neq \{i,j\}\ \forall \ell \geq k\} = 0, \quad \forall k \in \mathbb{P}. \tag{D.3}
$$

Next, notice that

$$
\begin{aligned}
&\mathrm{P}\{u(k) = \{i,j\} \text{ for infinitely many } k \in \mathbb{P}\} \\
&\quad = 1 - \mathrm{P}\{\cup_{k=1}^{\infty}\{u(\ell) \neq \{i,j\}\ \forall \ell \geq k\}\} \geq 1 - \sum_{k=1}^{\infty} \mathrm{P}\{u(\ell) \neq \{i,j\}\ \forall \ell \geq k\}.
\end{aligned}
$$

Thus, from (D.3),

$$
\mathrm{P}\{u(k) = \{i,j\} \text{ for infinitely many } k \in \mathbb{P}\} = 1. \tag{D.4}
$$

Now consider the set $\mathcal{E}_{\infty}$ defined by (5.22). Since $(u(k))_{k=1}^{\infty}$ appears in (5.22) and is a random sequence, the set $\mathcal{E}_{\infty}$ is a random variable taking values in the power set of the set $\{\{i, j\} : i, j \in \mathcal{V}, i \neq j\}$. From (D.4), we have $\mathrm{P}\{\{i, j\} \in \mathcal{E}_\infty\} = 1$. Since $\{i, j\}$ is an arbitrary element in $\tilde{\mathcal{E}}_\infty$, $\mathrm{P}\{\{i', j'\} \in \mathcal{E}_\infty \ \forall \{i', j'\} \in \tilde{\mathcal{E}}_\infty\} = 1$. Thus, $\mathrm{P}\{\tilde{\mathcal{E}}_\infty \subset \mathcal{E}_\infty\} = 1$. This, along with Assumption 5.3, implies that $\mathrm{P}\{\text{the graph } (\mathcal{V}, \mathcal{E}_\infty) \text{ is connected}\} = 1$. Therefore, it follows from the proof of Theorem 5.1 that (5.20) and (5.3) hold with probability 1.

D.3 Proof of Theorem 5.3

The proof is similar to that of Theorem 5.1. Let a, b, γ, and β be as defined in Appendix D.1. Then, due to (5.12), (5.28), (5.4), and Lemma 5.1, we have $\hat{x}_i(k) \in [a, b] \ \forall k \in \mathbb{N} \ \forall i \in \mathcal{V}$ and $x^* \in [a, b]$. From Lemma 5.3, $\lim_{k\to\infty} V(\mathbf{x}(k)) = c$ for some $c \geq 0$. To show that $c = 0$, assume to the contrary that $c > 0$ and let ϵ be as defined in D.1. Then, (D.1) holds for some $k_1 \in \mathbb{N}$. It follows from the proof of Lemma 5.3 that $f_i(\hat{x}_i(k)) - f_i(\hat{x}_i(k-1)) - f_i'(\hat{x}_i(k-1))(\hat{x}_i(k) - \hat{x}_i(k-1)) \leq V(\mathbf{x}(k-1)) - V(\mathbf{x}(k)) < \epsilon \ \forall k \geq k_1 + 1 \ \forall i \in u(k)$. Thus, $|\hat{x}_i(k) - \hat{x}_i(k-1)| \leq \gamma^{-1}(\epsilon) \ \forall k \geq k_1 + 1 \ \forall i \in u(k)$. This, along with (5.29) and the fact that $R \in \mathbb{P}$, implies $|\hat{x}_i(k) - \hat{x}_j(k)| \leq \frac{2\gamma^{-1}(\epsilon)}{1 - \frac{1}{2^R}} \leq 4\gamma^{-1}(\epsilon)$ $\forall k \geq k_1 \ \forall i, j \in u(k+1)$. Then, using the same idea as in D.1, it can be shown that $\max_{i\in\mathcal{V}} \hat{x}_i(k_1) - \min_{i\in\mathcal{V}} \hat{x}_i(k_1) \leq 4(N-1)\gamma^{-1}(\epsilon)$. This leads to $V(\mathbf{x}(k_1)) < c$, which contradicts (D.1). Therefore, (5.20) and (5.3) hold.

Appendix E Proofs for Chapter 6

E.1 Proof of Theorem 6.1

Let the sequence $(u(k))_{k=1}^{\infty}$ such that each $i \in \mathcal{V}$ appears infinitely often in it be given. Consider the following lemmas (see Appendix D.1 for proofs):

Lemma E.1. *For any given $[a,b] \subset \mathcal{X}$, $\exists$ a continuous and strictly increasing function $\gamma : [0,\infty) \to [0,\infty)$ satisfying $\gamma(0) = 0$ and $\lim_{d\to\infty} \gamma(d) = \infty$, such that $\forall \eta > 0$, $\forall \{i,j\} \in \mathcal{E}$, $\forall (x,y) \in [a,b]^2$, $f_{\{i,j\}}(y) - f_{\{i,j\}}(x) - f'_{\{i,j\}}(x)(y-x) \le \eta$ implies $|y-x| \le \gamma^{-1}(\eta)$.*

Lemma E.2. *For any given $[a,b] \subset \mathcal{X}$, $\exists \beta \in (0,\infty)$ such that $\forall \{i,j\} \in \mathcal{E}$, $\forall (x,y) \in [a,b]^2$, $f_{\{i,j\}}(y) - f_{\{i,j\}}(x) - f'_{\{i,j\}}(x)(y-x) \le \beta |y-x|$.*

Let $a = \min_{\{i,j\}\in\mathcal{E}} x_{\{i,j\}}(0)$ and $b = \max_{\{i,j\}\in\mathcal{E}} x_{\{i,j\}}(0)$. Then, it follows from (6.10) and (6.13) that

$$x_{\{i,j\}}(k) \in [a,b], \quad \forall k \in \mathbb{N},\ \forall \{i,j\} \in \mathcal{E}, \tag{E.1}$$

$$\hat{x}_i(k) \in [a,b], \quad \forall k \in \mathbb{N},\ \forall i \in \mathcal{V}. \tag{E.2}$$

In addition, notice from (6.4) and (6.3) that $\sum_{\{i,j\}\in\mathcal{E}} f_{\{i,j\}} = F$. It follows from (6.5) and (6.11) that $0 = \sum_{\{i,j\}\in\mathcal{E}} f'_{\{i,j\}}(x^*) = \sum_{\{i,j\}\in\mathcal{E}} f'_{\{i,j\}}(x_{\{i,j\}}(0)) = \sum_{\{i,j\}\in\mathcal{E}} f'_{\{i,j\}}(x_{\{i,j\}}(k))$. Thus, due to Lemma 6.1,

$$x^* \in [\min_{\{i,j\}\in\mathcal{E}} x_{\{i,j\}}(k), \max_{\{i,j\}\in\mathcal{E}} x_{\{i,j\}}(k)] \subset [a,b],\ \forall k \in \mathbb{N}. \tag{E.3}$$

By Lemma E.1, $\exists$ a continuous and strictly increasing function $\gamma : [0,\infty) \to [0,\infty)$ satisfying $\gamma(0) = 0$ and $\lim_{d\to\infty} \gamma(d) = \infty$, such that

$$\forall \eta > 0,\ \forall \{i,j\} \in \mathcal{E},\ \forall (x,y) \in [a,b]^2, f_{\{i,j\}}(y) - f_{\{i,j\}}(x) - f'_{\{i,j\}}(x)(y-x) \le \eta \\ \Rightarrow |y-x| \le \gamma^{-1}(\eta). \tag{E.4}$$

By Lemma E.2, $\exists \beta \in (0,\infty)$ such that

$$\forall \{i,j\} \in \mathcal{E},\ \forall (x,y) \in [a,b]^2,\ f_{\{i,j\}}(y) - f_{\{i,j\}}(x) - f'_{\{i,j\}}(x)(y-x) \le \beta |y-x|. \tag{E.5}$$

We first show that (6.17) holds. Due to (6.16), (6.15), and Lemma 6.2, $(V(\mathbf{x}(k)))_{k=0}^{\infty}$ is nonnegative and non-increasing. Thus, $\exists c \ge 0$ such that $\lim_{k\to\infty} V(\mathbf{x}(k)) = c$. To show that c must be zero, assume, to the contrary, that $c > 0$. Let $\epsilon > 0$ be given by $\epsilon = \gamma(\frac{c}{\beta N L})$. Then, $\exists k_1 \in \mathbb{N}$ such that

$$c \le V(\mathbf{x}(k)) < c + \epsilon, \quad \forall k \ge k_1. \tag{E.6}$$

Due to (E.6), $V(\mathbf{x}(k-1)) - V(\mathbf{x}(k)) < \epsilon\ \forall k \ge k_1 + 1$, which, along with (6.19) and (6.15), implies that $f_{\{u(k),j\}}(\hat{x}_{u(k)}(k-1)) - f_{\{u(k),j\}}(x_{\{u(k),j\}}(k-1)) - f'_{\{u(k),j\}}(x_{\{u(k),j\}}(k-1))(\hat{x}_{u(k)}(k-1) - x_{\{u(k),j\}}(k-1)) < \epsilon\ \forall j \in \mathcal{N}_{u(k)}$. Thus, due to (6.11), (6.12), (E.1), (E.2), and (E.4),

$$|x_{\{u(k),j\}}(k) - x_{\{u(k),j\}}(k-1)| = |\hat{x}_{u(k)}(k-1) - x_{\{u(k),j\}}(k-1)| \le \gamma^{-1}(\epsilon), \\ \forall k \ge k_1 + 1,\ \forall j \in \mathcal{N}_{u(k)}. \tag{E.7}$$

Next, consider the following three definitions and a lemma: first, $\forall k \ge k_1$, $\forall i \in \mathcal{V}$, let $C_i(k) \subset \mathcal{V}$ be defined recursively as follows: for $k = k_1$, let $C_i(k) = \emptyset\ \forall i \in \mathcal{V}$. For each $k \ge k_1 + 1$, let

$$C_i(k) = \begin{cases} (\cup_{j\in\mathcal{N}_{u(k)}} C_j(k-1)) \cup \{u(k)\}, & \text{if } i \in (\cup_{j\in\mathcal{N}_{u(k)}} C_j(k-1)) \cup \{u(k)\}, \\ C_i(k-1), & \text{otherwise,} \end{cases} \\ \forall i \in \mathcal{V}. \tag{E.8}$$

Second, $\forall X \subset \mathcal{V}$, let $\mathcal{E}_X = \{\{i,j\} \in \mathcal{E} : i \in X \text{ or } j \in X\}$. Third, $\forall k \geq k_1$, $\forall \ell \in \mathcal{V}$, let $d_\ell(k) = \max_{\{i,j\}\in\mathcal{E}_{C_\ell(k)}} x_{\{i,j\}}(k) - \min_{\{i,j\}\in\mathcal{E}_{C_\ell(k)}} x_{\{i,j\}}(k)$, where max and min, when taken over an empty set, are assumed to be 0.

Lemma E.3. *For each $k \geq k_1$,*

$$d_\ell(k) \leq ||C_\ell(k)| - 1|\gamma^{-1}(\epsilon), \quad \forall \ell \in \mathcal{V}. \tag{E.9}$$

Proof. By induction over k. Let $k = k_1$. For each $\ell \in \mathcal{V}$, since $C_\ell(k) = \emptyset$, we have $\mathcal{E}_{C_\ell(k)} = \emptyset$, so that $d_\ell(k) = 0\ \forall \ell \in \mathcal{V}$. Therefore, (E.9) holds for $k = k_1$. Next, let $k \geq k_1 + 1$ and suppose

$$d_\ell(k-1) \leq ||C_\ell(k-1)| - 1|\gamma^{-1}(\epsilon), \quad \forall \ell \in \mathcal{V}. \tag{E.10}$$

We will show that (E.10) implies (E.9). To do so, consider the following two mutually exclusive and exhaustive cases:

Case (I): $C_{u(k)}(k-1) \neq \emptyset$. It can be seen from (E.8) that

$$\mathcal{E}_{\{u(k)\}} \subset \mathcal{E}_{C_{u(k)}(k-1)}, \tag{E.11}$$

$$C_\ell(k) = C_\ell(k-1), \quad \forall \ell \in \mathcal{V}. \tag{E.12}$$

Because of (6.11) and Lemma 6.1, $\min_{\{i,j\}\in\mathcal{E}_{\{u(k)\}}} x_{\{i,j\}}(k-1) \leq x_{\{u(k),\ell\}}(k) \leq \max_{\{i,j\}\in\mathcal{E}_{\{u(k)\}}} x_{\{i,j\}}(k-1)\ \forall \ell \in \mathcal{N}_{u(k)}$. Also, due to (6.11), $x_{\{i,j\}}(k) = x_{\{i,j\}}(k-1)\ \forall \{i,j\} \in \mathcal{E} - \mathcal{E}_{\{u(k)\}}$. It follows from (E.8), (E.11), and (E.12) that $\forall \ell \in \mathcal{V}$, $\min_{\{i,j\}\in\mathcal{E}_{C_\ell(k-1)}} x_{\{i,j\}}(k-1) \leq \min_{\{i,j\}\in\mathcal{E}_{C_\ell(k)}} x_{\{i,j\}}(k) \leq \max_{\{i,j\}\in\mathcal{E}_{C_\ell(k)}} x_{\{i,j\}}(k) \leq \max_{\{i,j\}\in\mathcal{E}_{C_\ell(k-1)}} x_{\{i,j\}}(k-1)$. This implies that $d_\ell(k) \leq d_\ell(k-1)\ \forall \ell \in \mathcal{V}$. Therefore, from (E.10) and (E.12), (E.9) holds for this Case (I).

Case (II): $C_{u(k)}(k-1) = \emptyset$. Consider two subcases:

Subcase (i): $\ell \in \mathcal{V} - ((\cup_{j\in\mathcal{N}_{u(k)}} C_j(k-1)) \cup \{u(k)\})$. From (E.8) and (6.11), $C_\ell(k) = C_\ell(k-1)$ and $x_{\{i,j\}}(k) = x_{\{i,j\}}(k-1)\ \forall \{i,j\} \in \mathcal{E}_{C_\ell(k)}$. Thus,

$d_\ell(k) = d_\ell(k-1)$, which, together with (E.10), implies that in this Subcase (i), $d_\ell(k) \leq ||C_\ell(k)| - 1|\gamma^{-1}(\epsilon)$.

Subcase (ii): $\ell \in (\cup_{j \in \mathcal{N}_{u(k)}} C_j(k-1)) \cup \{u(k)\}$. From (E.8), $u(k) \notin C_j(k-1) \; \forall j \in \mathcal{N}_{u(k)}$, implying that

$$|C_\ell(k)| = | \cup_{j \in \mathcal{N}_{u(k)}} C_j(k-1)| + 1 \geq 1. \tag{E.13}$$

Suppose $\cup_{j \in \mathcal{N}_{u(k)}} C_j(k-1) = \emptyset$, i.e., $C_j(k-1) = \emptyset \; \forall j \in \mathcal{N}_{u(k)}$. Then, from (E.8), $C_\ell(k) = \{u(k)\}$. This, along with (6.11), implies that $\min_{\{i,j\} \in \mathcal{E}_{C_\ell(k)}} x_{\{i,j\}}(k) = \max_{\{i,j\} \in \mathcal{E}_{C_\ell(k)}} x_{\{i,j\}}(k)$. Thus,

$$d_\ell(k) = 0 = ||C_\ell(k)| - 1|\gamma^{-1}(\epsilon). \tag{E.14}$$

Now suppose $\cup_{j \in \mathcal{N}_{u(k)}} C_j(k-1) \neq \emptyset$, and pick any $\ell' \in \mathcal{N}_{u(k)}$ with $C_{\ell'}(k-1) \neq \emptyset$, i.e., $|C_{\ell'}(k-1)| \geq 1$. From (E.8), $\mathcal{E}_{C_{\ell'}(k-1)} \cap \mathcal{E}_{\{u(k)\}} \neq \emptyset$, because it contains $\{u(k), \ell'\}$. Thus, due to (6.11), $|x_{\{i,j\}}(k) - x_{\{p,q\}}(k)| \leq |x_{\{i,j\}}(k-1) - x_{\{p,q\}}(k)|$ $\forall \{i,j\} \in \mathcal{E}_{C_{\ell'}(k-1)}$ $\forall \{p,q\} \in \mathcal{E}_{C_{\ell'}(k-1)} \cap \mathcal{E}_{\{u(k)\}}$, because $x_{\{i,j\}}(k) = x_{\{p,q\}}(k)$ if $\{i,j\} \in \mathcal{E}_{C_{\ell'}(k-1)} \cap \mathcal{E}_{\{u(k)\}}$ and $x_{\{i,j\}}(k) = x_{\{i,j\}}(k-1)$ if $\{i,j\} \in \mathcal{E}_{C_{\ell'}(k-1)} - \mathcal{E}_{\{u(k)\}}$. It follows from the triangle inequality, (E.10), and (E.7) that

$$\begin{aligned}
|x_{\{i,j\}}(k) - x_{\{p,q\}}(k)| &\leq |x_{\{i,j\}}(k-1) - x_{\{p,q\}}(k-1)| + |x_{\{p,q\}}(k-1) - x_{\{p,q\}}(k)| \\
&\leq d_{\ell'}(k-1) + \gamma^{-1}(\epsilon) \\
&\leq |C_{\ell'}(k-1)|\gamma^{-1}(\epsilon), \quad \forall \{i,j\} \in \mathcal{E}_{C_{\ell'}(k-1)},\ \forall \{p,q\} \in \mathcal{E}_{C_{\ell'}(k-1)} \cap \mathcal{E}_{\{u(k)\}}.
\end{aligned} \tag{E.15}$$

In addition, from (6.11), $|x_{\{i,j\}}(k) - x_{\{p,q\}}(k)| = |x_{\{i,j\}}(k-1) - x_{\{p,q\}}(k-1)| \leq d_{\ell'}(k-1) \; \forall \{i,j\}, \{p,q\} \in \mathcal{E}_{C_{\ell'}(k-1)} - \mathcal{E}_{\{u(k)\}}$. This, along with (E.15) and (E.10), implies that

$$\max_{\{i,j\} \in \mathcal{E}_{C_{\ell'}(k-1)}} x_{\{i,j\}}(k) - \min_{\{i,j\} \in \mathcal{E}_{C_{\ell'}(k-1)}} x_{\{i,j\}}(k) \leq |C_{\ell'}(k-1)|\gamma^{-1}(\epsilon). \tag{E.16}$$

Note from (6.11) and (E.8) that $\{x_{\{i,j\}}(k) : \{i,j\} \in \mathcal{E}_{C_\ell(k)}\} = \{x_{\{i,j\}}(k) : \{i,j\} \in \mathcal{E}_{\cup_{j \in \mathcal{N}_{u(k)}} C_j(k-1)}\}$. Thus, there exist $\ell_1, \ell_2 \in \mathcal{N}_{u(k)}$, $\{i_1, j_1\} \in \mathcal{E}_{C_{\ell_1}(k-1)}$, and $\{i_2, j_2\} \in \mathcal{E}_{C_{\ell_2}(k-1)}$ such that $x_{\{i_1,j_1\}}(k) = \min_{\{i,j\} \in \mathcal{E}_{C_\ell(k)}} x_{\{i,j\}}(k)$ and $x_{\{i_2,j_2\}}(k) = \max_{\{i,j\} \in \mathcal{E}_{C_\ell(k)}} x_{\{i,j\}}(k)$. If $C_{\ell_1}(k-1) = C_{\ell_2}(k-1)$, then because of (E.13) and (E.16),

$$d_\ell(k) \le |C_{\ell_1}(k-1)|\gamma^{-1}(\epsilon) \le ||C_\ell(k)| - 1|\gamma^{-1}(\epsilon). \tag{E.17}$$

Otherwise, i.e., $C_{\ell_1}(k-1)$ and $C_{\ell_2}(k-1)$ are distinct, pick any $\{p_1, q_1\} \in \mathcal{E}_{C_{\ell_1}(k-1)} \cap \mathcal{E}_{\{u(k)\}}$ and any $\{p_2, q_2\} \in \mathcal{E}_{C_{\ell_2}(k-1)} \cap \mathcal{E}_{\{u(k)\}}$. Note from (E.8) that $\{p_1, q_1\}$ and $\{p_2, q_2\}$ exist. Also note from (6.11) that $x_{\{p_1,q_1\}}(k) = x_{\{p_2,q_2\}}(k)$. Thus, from (E.13) and (E.15), $d_\ell(k) = x_{\{i_2,j_2\}}(k) - x_{\{p_2,q_2\}}(k) + x_{\{p_1,q_1\}}(k) - x_{\{i_1,j_1\}}(k) \le |C_{\ell_2}(k-1)|\gamma^{-1}(\epsilon) + |C_{\ell_1}(k-1)|\gamma^{-1}(\epsilon) \le ||C_\ell(k)| - 1|\gamma^{-1}(\epsilon)$. Combining this, (E.14), and (E.17), we see that in this Subcase (ii), $d_\ell(k) \le ||C_\ell(k)| - 1|\gamma^{-1}(\epsilon)$.

Therefore, (E.9) holds for Case (II). This completes the proof by induction. $\square$

Since each $i \in \mathcal{V}$ appears infinitely often in $(u(k))_{k=1}^\infty$, $\exists k_2 \ge k_1 + 1$ such that $\forall i \in \mathcal{V}$, $u(k) = i$ for some $k \in [k_1 + 1, k_2]$. Due to (E.8) and the connectedness of $\mathcal{G}$, $C_i(k_2) = \mathcal{V}$ $\forall i \in \mathcal{V}$. This, along with the property $\mathcal{E}_\mathcal{V} = \mathcal{E}$ and Lemma E.3 that $\max_{\{i,j\} \in \mathcal{E}} x_{\{i,j\}}(k_2) - \min_{\{i,j\} \in \mathcal{E}} x_{\{i,j\}}(k_2) = d_\ell(k_2) \le (N-1)\gamma^{-1}(\epsilon)$ $\forall \ell \in \mathcal{V}$. It follows from (6.16), (E.1), (E.3), (E.5), and the value of ϵ that $V(\mathbf{x}(k_2)) \le \beta \sum_{\{i,j\} \in \mathcal{E}} |x^* - x_{\{i,j\}}(k_2)| \le \beta \sum_{\{i,j\} \in \mathcal{E}} (\max_{\{p,q\} \in \mathcal{E}} x_{\{p,q\}}(k_2) - \min_{\{p,q\} \in \mathcal{E}} x_{\{p,q\}}(k_2)) \le \beta L(N-1)\gamma^{-1}(\epsilon) = \frac{N-1}{N} c < c$, which contradicts (E.6). Therefore, $c = 0$, i.e., (6.17) holds, implying that (6.18) holds. Furthermore, due to (6.12), (6.14) is satisfied.

E.2 Proof of Theorem 6.2

Similar to the proof of Theorem 6.1, to show that (6.17) holds, we assume to the contrary that $\lim_{k\to\infty} V(\mathbf{x}(k)) = c > 0$. Let $\epsilon > 0$ be given by $\epsilon = \gamma(\frac{c}{\beta L(N+1)})$. Then, $\exists k_1 \in \mathbb{N}$ such that (E.6) holds, implying that $V(\mathbf{x}(k_1)) - V(\mathbf{x}(k_1+1)) < \epsilon$. It follows from (6.21), (6.20), (6.22), and (6.15) that $f_{\{i,j\}}(\hat{x}_i(k_1)) - f_{\{i,j\}}(x_{\{i,j\}}(k_1)) - f'_{\{i,j\}}(x_{\{i,j\}}(k_1))(\hat{x}_i(k_1) - x_{\{i,j\}}(k_1)) < \epsilon$ $\forall i \in \mathcal{V}\ \forall j \in \mathcal{N}_i$. This, along with (E.1), (E.2), and (E.4), implies that $|\hat{x}_i(k_1) - x_{\{i,j\}}(k_1)| \le \gamma^{-1}(\epsilon)$ $\forall i \in \mathcal{V}\ \forall j \in \mathcal{N}_i$. Thus, by the triangle inequality, $\max_{\{i,j\}\in\mathcal{E}} x_{\{i,j\}}(k_1) - \min_{\{i,j\}\in\mathcal{E}} x_{\{i,j\}}(k_1) \le N\gamma^{-1}(\epsilon)$. It follows from (6.16), (E.1), (E.3), (E.5), and the value of ϵ that $V(\mathbf{x}(k_1)) \le \beta \sum_{\{i,j\}\in\mathcal{E}} |x^* - x_{\{i,j\}}(k_1)| \le \beta L N \gamma^{-1}(\epsilon) = \frac{N}{N+1}c < c$. This is a contradiction to (E.6). Therefore, (6.17) holds, implying that (6.18) and (6.14) hold.