



Università degli Studi di Cagliari

DOTTORATO DI RICERCA
in Ingegneria Elettronica e dell'Informazione
Ciclo XXIII

TITOLO TESI

**Consensus Algorithms for Estimation and Discrete
Averaging in Networked Control Systems.**

Settore scientifico disciplinare di afferenza
ING-INF/04 Automatica

Presentata da: Mauro Franceschelli
Coordinatore Dottorato: Prof. Alessandro Giua
Tutor: Prof. Alessandro Giua

Esame finale anno accademico 2009-2010



*Ph.D. in Electronic and Computer Engineering
Dept. of Electrical and Electronic Engineering
University of Cagliari*



Consensus Algorithms for Estimation and Discrete Averaging in Networked Control Systems.

Mauro Franceschelli

*Advisor: Prof. Alessandro Giua.
Curriculum: ING-INF/04 Automatica*

XXIII Cycle
December 2010

Dedicated to my parents

Abstract

In this thesis several topics on consensus and gossip algorithms for multi-agent systems are addressed. An agent is a dynamical system that can be fully described by a state-space representation of its dynamics. A multi-agent system is a network of agents whose pattern of interactions or couplings is described by a graph. Consensus problems in multi-agent systems consist in the study of local interaction rules between the agents such that as global emergent behavior the network converges to the so called "consensus" or "agreement" state where the value of each agent's state is the same and it is possibly a function of the initial network state, for instance the average. A consensus algorithm is thus a set of local interaction rules that solve the consensus problem under some assumptions on the network topology. A gossip algorithm is a set of local state update rules between the agents that, disregarding their objective, are supposed to be implemented in a totally asynchronous way between pairs of neighboring agents, thus resembling the act of "gossiping" in a crowd of people.

In this thesis several algorithms based on gossip that solve the consensus and other related problems are presented.

In the first part, several solutions to the consensus problem based on gossip under different sets of assumptions are proposed. In the first case, it is assumed that the state of the agents is discretized and represents a collection of tasks of different size. In the second case, under the same discretization assumptions of the first case, it is assumed that the network is represented by a Hamiltonian graph and it is shown how under this assumption the convergence speed can be improved. In the third case, a solution for the consensus problem for networks represented by arbitrary strongly connected directed graphs

is proposed, assuming that the state of the agents is a real number. In the fourth case, a coordinate-free consensus algorithm based on gossip is designed and applied to a network of vehicles able to sense the relative distance between each other but with no access to absolute position information or to a common coordinate system. The proposed algorithm is then used to build in a decentralized way a common reference frame for the network of vehicles.

In the second part, a novel local interaction rule based on the consensus equation is proposed together with an algorithm to estimate in a decentralized way the spectrum of the Laplacian matrix that encodes the network topology. As emergent behavior, each agent's state oscillates only at frequencies corresponding to the eigenvalues of the Laplacian matrix thus mapping the spectrum estimation problem into a signal processing problem solvable using the *Fourier Transform*. It is further shown that the constant component of the emergent behavior in the frequency domain solves the consensus on the average problem. The spectrum estimation algorithm is then applied to leader-follower networks of mobile vehicles to infer in a decentralized way properties such as controllability, observability and other topological features of the network such as its topology.

Finally, a fault detection and recovery technique for sensor networks based on the so called motion-probes is presented to address the inherent lack of robustness against outlier agents in networks implementing consensus algorithms to solve the distributed averaging problem.

Acknowledgements

I would like to thank all the people that in these three years supported me both from a scientific and human point of view.

First, I would like to thank my advisor Prof. Alessandro Giua and Carla Seatzu for their support along my graduate studies.

Special thanks go to Magnus Egerstedt, for his great hospitality, support and encouragement and to all my co-workers: Andrea Gasparri, Simone Martini, Cristian Mahulea, Nicola Orani, Alessandro Pisano, Elio Usai and Antonio Bicchi, for their helpful collaboration.

I thank all my friends for their invaluable friendship during this years.

Finally, I thank my parents Marcello and Aurora, for their help during all the hardest moments of my studies.

Contents

1	Introduction	1
1.1	Introduction to multi-agent systems in control	1
1.2	Overview and contributions	3
2	State of the art and related literature	7
2.1	Consensus in Continuous-time	8
2.2	Consensus in Discrete-time	12
2.3	Consensus based on Gossip	14
2.4	Quantized Consensus	18
3	Discrete Consensus on Heterogeneous Networks	21
3.1	Introduction	21
3.2	Problem description	23
3.3	Proposed algorithm	24
3.4	Convergence properties	26
3.5	Conclusions	38
4	Quantized Consensus on Hamiltonian Graphs	41
4.1	Introduction	41
4.2	Problem description	44
4.3	Proposed algorithm	46
4.4	Convergence properties	51
4.5	Conclusions	63
5	Consensus problems in directed graphs	65
5.1	Introduction	65

5.2	Proposed algorithm	69
5.3	Convergence properties	72
5.4	Simulations	78
5.5	Conclusions	91
6	Gossip based consensus in absence of common reference frames	93
6.1	Introduction	93
6.2	Problem description	95
6.3	Agreement on a common point in a 2-D space	97
6.4	Agreement on a common reference frame in a 2-D space	107
6.5	Agreement on a common point in d-D space	108
6.6	Simulations	112
6.7	Conclusions	114
7	Fault detection and recovery in consensus networks	117
7.1	Introduction	117
7.2	Problem description	118
7.3	Motion Probes	120
7.4	Fault recovery	124
7.5	Fault Diagnosis	132
7.6	Simulations	136
7.7	Conclusions	142
8	Consensus based decentralized Laplacian Spectrum estimation	143
8.1	Introduction	143
8.2	Related literature	144
8.3	Problem description	146
8.4	Proposed algorithm	148
8.5	Implementation issues of the approach	156
8.6	Conclusions	161
9	Spectrum based Controllability, observability and topology estimation	165
9.1	Introduction	165
9.2	Background on leader-follower networks	167

9.3	Spectrum based decentralized check for observability and control-	
	lability	169
9.4	Spectrum based Formation Identification	172
9.5	Conclusions	175
10	Conclusions	177
A	Appendix	185
A.1	Algebraic graph theory	185
A.2	Graph spectral analysis and Multi-agent systems	187
	Bibliography	191

Chapter 1

Introduction

1.1 Introduction to multi-agent systems in control

In the last decades, advances in computing and communication systems and a great reduction in cost for processing units has spurred innovation in many areas of information technology such as systems theory. Low-cost wireless devices are now ubiquitous and the efficient management of networks of interacting embedded systems such as sensor networks has created new challenges for the development of networked control systems. Furthermore, the reduction in cost of accurate inertial measurement systems and GPS receivers has greatly pushed forward the research in networks of autonomous mobile vehicles. All this technological advances require new ways of managing the information flow generated by the single units. In particular, the design of control systems has shifted from "centralized" approaches, where all the information available is gathered in some point of time and space and then decisions are dispatched through the network, to "decentralized" approaches, where the information locally gathered by the units (agents) is processed in locus and control decisions are taken cooperatively by the agents with no supervision. To study this kind of large scale systems in system theory, the so called "Multi-Agent" paradigm has been developed. Each agent is assumed to have some peculiar dynamics that can be fully described by its input/state map, the network or collection of agents is then described using abstractions such as graphs. One of the most common objectives is to infer the collective emergent behavior of the network from the knowledge of the agents' dynamics, their local

interaction dynamics with other agents and the network structure or topology.

A way to address the design of local interaction rules from which the desired collective behaviors emerge is to take inspiration from biological and social systems. In biology, collective emergent behaviors are observed in several different species. Ant colonies and their ability to find shortest paths between the nest and the foraging areas (1, 2) has inspired the development of algorithms to solve the Traveling Salesman Problem TSP (3). The study of the behavior of fish schools (4), flocks of birds and other animals have inspired the famous "Boyd" model and the three rules of flocking (5) that sequently have attracted a huge amount of researchers from system theory to explain the flocking dynamics or to design bio-inspired flocking algorithms for unmanned aerial vehicles (6, 7, 8). In statistical physics the study of the emerging behavior of self-propelled particle systems (9, 10) has received a significant amount of attention and several simple local interaction rules that produce flocking or swirling behaviors have been produced. Another notable example of self-propelled particle system is the one that models escape dynamics in a crowd of people (11). Evidence suggests that even when the agents are driven by complex dynamics, most emergent behaviors in nature can be modeled by simplified systems where the global dynamics are dominated by the local interactions between the agents.

In system theory, the consensus problem consists in the design of local interaction rules between the agents such that as global emergent behavior the network converges to the so called "consensus" or "agreement" state where the value of each agent's state is the same and it is a function of the initial network state. If the *consensus* state corresponds to the average of the initial network states then we refer to it as *consensus on the average*. The consensus problem has attracted a huge amount of researchers that have addressed it under different sets of assumptions on the agents dynamics and for several distinct applications. One of the firsts papers on the consensus problem dates back to 1972, where Morris H. DeGroot addressed the problem of how to make a set of agents cooperate to reach a common estimation for the probability distribution of an unknown parameter (12). Several other authors addressed related problems such as decentralized decision making (13), where a set of voters have to agree on a boolean decision or algorithms for distributed clock synchronization (14).

In the last decade there has been a surge in interest for distributed systems and

the consensus problem has evolved to address problems such as the cooperative rendezvous in a network of mobile vehicles (15, 16, 17, 18, 19), distributed clock synchronization based in consensus (20, 21, 22), the distributed average problem in sensor networks (15, 23, 24, 25, 26), load balancing on networks (27, 28, 29, 30, 31, 32, 33) and many more.

The first part of the thesis focuses on instances of consensus problems for load balancing and task assignment, distributed averaging in sensor networks, consensus in absence of a common reference frame for networks of mobile agents and fault detection and recovery for the distributed averaging problem.

The second part of the thesis focuses on a novel emerging behavior formally based on the standard continuous-time formulation of the consensus problem to estimate information about the network topology of a multi-agent system and on applications of such information to leader-follower networks.

1.2 Overview and contributions

The thesis is structured as follows:

- In **Chapter 2** the state of the art regarding consensus and gossip algorithms is presented. First, the consensus problem is stated in continuous time assuming that the dynamics of the agents is represented by a single integrator, this modeling is suitable for mobile agents. Second, the consensus problem is stated in discrete time assuming that the dynamics of the agents is a single discrete time integrator, this modeling is suitable for sensor networks. Third, the state of the art of consensus problems solved via gossip communications is presented, the use of gossiping is particularly suitable when the agents are interacting through wireless communications. Finally, the consensus problem is stated in the case of agents with a quantized state represented by integer numbers, this modeling is suitable for load balancing problems or sensor networks.
- In **Chapter 3** the framework denoted as *discrete consensus* is introduced. It is a generalization of *quantized consensus*. We assumed that a set of tasks of different weight should be assigned to nodes with different speeds with the aim of minimizing the maximum execution time. A solution based on

gossip is proposed and convergence properties are examined in detail for several network topologies.

- In **Chapter 4** the Hamiltonian Quantized Gossip Algorithm is proposed. It solves the quantized distributed average problem and the token distribution problem on Hamiltonian graphs with a greater efficiency respect to other gossip algorithms based on uniform quantization (29, 34). The main feature of the proposed algorithm is an embedded stopping criterion that blocks the algorithm once quantized consensus has been achieved. It is also shown that, if there exists a periodic interval of time where each edge along the Hamiltonian cycle is selected at least once, a finite time convergence bound can be given thus ensuring a finite and known amount of total transfers for load balancing applications and a decentralized criterion to stop averaging in sensor networks.

- In **Chapter 5** a novel gossip algorithm based on broadcasts that achieves consensus on the average on arbitrary strongly connected digraphs is proposed. The main feature this algorithm is that it converges *exactly* to the average of the initial state despite mono directional communications.

A comparison with the standard gossip algorithm based on broadcast and with the standard gossip based on pairwise averaging has also been made. Simulations show that the proposed algorithm achieves better convergence rates and energy saving than the standard gossip based on pairwise averaging if the number of neighbors of each node is sufficiently high.

- In **Chapter 6** a novel approach to the problem of decentralized agreement toward a common point in space in absence of a common reference frame is presented. In this scenario, an agent is assumed to be able to sense the distance between itself and its neighbors and the direction in which it sees its neighbors with respect to its local reference frame. The proposed approach allows to perform an agreement on the network centroid, on a common reference frame and therefore on a common heading. Using this information a global positioning system for the agents using only local measurements can be achieved. Furthermore only point-to-point asynchronous communications between neighboring agents are allowed thus achieving robustness

against random communication failures. The cases in which the agents are in a 2-D or 3-D space are addressed separately.

- In **Chapter 7** the problem of how to update the state of agents in a networked system in such a way that they do not change the desired convergence point is considered. Such movements are referred to as motion probes. We show how such motion probes can be used to identify faulty agents that do not exhibit the prescribed dynamical behavior. Moreover, once these faulty agents have been identified, we show how to nullify their impact on the behavior of the non-faulty agents.
- In **Chapter 8** a decentralized algorithm to estimate the Laplacian spectrum of a network is proposed. The key idea of the algorithm is to provide a local interaction rule among agents whose goal is to make their state oscillate at frequencies corresponding to the eigenvalues of the network topology. In this way, the problem of decentralized spectrum identification is mapped into a problem of signal processing that each agent can efficiently and independently solve by applying the *Fast Fourier Transform* algorithm to its state trajectory.
- In **Chapter 9** a decentralized method for online verification of controllability and observability of a leader-follower network is proposed. The method is based on the spectrum estimation algorithm presented in chapter 8. Furthermore, the use of the Laplacian spectrum of the network of a multi-agent system is proposed to identify when a desired formation is achieved.

Chapter 2

State of the art and related literature

In this chapter we survey many results regarding consensus problems in different frameworks.

Let us consider a multi-agent system in which the network topology is described by a graph $G = (V, E)$ where $V = \{1, \dots, n\}$ is the set of agents, represented by nodes in the graph, and $E \subseteq \{V \times V\}$ is the set of edges. An edge (i, j) exists if agent i interacts with agent j . The neighborhood of each agent i is defined as $N_i = \{j : (i, j) \in E\}$ which represents the set of agents which directly interact with it.

Each agent has dynamics

$$\dot{x}_i = f(x_i, u_i),$$

where $x_i \in \mathbb{R}^m$ is the agent's state and $u_i \in \mathbb{R}^m$ is the vector of inputs.

A local interaction protocol for agent i is defined as:

$$u_i = u(x_j : j \in N_i),$$

it represents the coupling between agent i and its neighbors.

A typical autonomous multi-agent system is thus completely defined by the network topology G , the agents' dynamics and the local interaction protocol.

Definition 2.0.1 (Consensus problem)

Consider a multi-agent system defined by the network topology $G = (V, E)$ and the agents' dynamics $\dot{x}_i = f(x_i, u_i)$ with $x_i \in \mathbb{R}^m$. A consensus problem consists in the design of a local interaction protocol $u(x_j : j \in N_i)$ such that:

$$\forall i, j \in V, \quad \lim_{t \rightarrow \infty} \|x_i - x_j\| = 0 \quad (2.1)$$

■

In the following sections several frameworks for modeling multi-agent systems will be presented, each differing in interaction protocols adopted and agents and network models.

In Section 2.1 the consensus problem for a continuous time network of single integrators is introduced (6, 15, 19, 24, 35) both for a static or switching topology. In Section 2.2 the consensus problem for discrete time single integrators is presented (23, 36). In Section 2.8 gossip algorithms are introduced and results on consensus based on gossip communications are reviewed (37, 38, 39). Finally in Section 2.4 the consensus problem under quantization constraints is presented.

2.1 Consensus in Continuous-time

Let $G = \{V, E\}$ be a graph that describes the topology of the network of agents. Assume that each agent is a single continuous time integrator with dynamics:

$$\dot{x}_i = u_i,$$

where $x_i, u_i \in \mathbb{R}^d$. This simple model is useful to describe mobile agents where the state x represent their positions in a d -dimension space and u_i is the actuated speed of the vehicle. Since most vehicles' dynamics are constrained, this model is useful when the scale of the motions are much greater than the characteristic size of the vehicle so that eventual non-holonomic constraints can be neglected.

Now let us consider the mono-dimensional case $d = 1$ and let the agents adopt the following linear interaction protocol:

$$u_i = \sum_{j \in N_i} (x_j - x_i). \quad (2.2)$$

Each agent's dynamics thus become:

$$\dot{x}_i = \sum_{j \in N_i} (x_j - x_i).$$

The global system dynamics is linear and can be compactly represented by:

$$\dot{x} = -\mathcal{L}(G)x \quad (2.3)$$

where $x \in \mathbb{R}^n$, n is the number of agents, G is the graph representing the network topology and $\mathcal{L}(G)$ is the so called $n \times n$ Laplacian matrix (40) that encodes graph G , its elements are defined as:

$$l_{ij} = \begin{cases} -1 & \text{if } (i, j) \in E \\ -\sum_{j \in N_i(t)} l_{ij} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

Example 2.1.1 *Let us consider the multi-agent system depicted in Figure ??, the network G consists of 6 agents with $V = \{1, 2, 3, 4, 5, 6\}$ and edge set*

$$E = \{(1, 2), (1, 3), (1, 6), (3, 4), (3, 5), (3, 6), (2, 5), (2, 6), (2, 1), (3, 1), (6, 1), (4, 3), (5, 3), (6, 3), (5, 2), (6, 2)\}.$$

The corresponding graph Laplacian is as follows:

$$\mathcal{L}(G) = \begin{bmatrix} 3 & -1 & -1 & 0 & 0 & -1 \\ -1 & 2 & 0 & 0 & -1 & -1 \\ -1 & 0 & 4 & -1 & -1 & -1 \\ 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & -1 & -1 & -1 & 3 & 0 \\ -1 & 0 & -1 & 0 & 0 & 2 \end{bmatrix}.$$

■

The Laplacian matrix of a graph has several interesting properties due to its structure. First, let $\mathbf{1}_n$ and $\mathbf{0}_n$ be respectively the n elements vectors of ones and zeros. For any graph G , $\mathcal{L}(G)\mathbf{1} = \mathbf{0}$ by construction. If G is balanced then $\mathbf{1}^T \mathcal{L}(G) = \mathbf{0}^T$. If G is an undirected graph then the number of null eigenvalues of \mathcal{L} is $n - c$ where c is the number of connected components of G (41). If G is a directed strongly connected graph, then $\text{rank}(\mathcal{L}) = n - 1$ (24). The $\lambda_1, \lambda_2, \dots, \lambda_n$

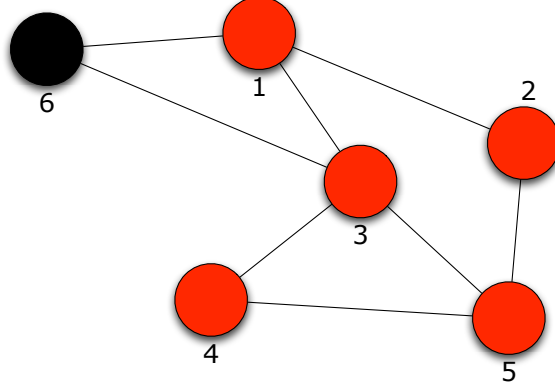


Figure 2.1: Topology of the multi-agent system in example 5.3.5.

be the n eigenvalues of $\mathcal{L}(G)$. If G is undirected then $\mathcal{L}(G)$ is symmetric, all its eigenvalues are real and the following holds (42): $0 \leq \lambda_1 \leq \lambda_2 \dots \leq \lambda_n$. Let the in-degree $\delta_{in,i}$ and out-degree $\delta_{out,i}$ of node i , be respectively the number of incoming and outgoing edges incident on i . If G is a directed graph, then we have the following result (24) originally presented for weighted digraphs:

Theorem 2.1.2 *Let $G = (V, E)$ be a digraph with Laplacian L . Denote the maximum node in-degree of the digraph G by $\delta_{max}(G) = \max_i \delta_{in,i}$. Then, all the eigenvalues of $\mathcal{L}(G)$ are located in the following disk*

$$D(G) = \{z \in \mathbb{C} : |z - \delta_{max}(G)| \leq \delta_{max}(G)\},$$

centered at $z = \delta_{max}(G + j0$ in the complex plane.

Proof: By applying the Gershgorin disk theorem to matrix \mathcal{L} we see that by construction $\delta_{in,i} = l_{ii} = -\sum_{j \in N_i} l_{ij}$. Thus each of the n Gershgorin disks is centered at $z_i = \delta_{in,i}(G + j0$ with corresponding radius

$$D_i(G) = \{z \in \mathbb{C} : |z - \delta_{in,i}(G)| \leq \delta_{in,i}\}.$$

It follows that the disk with the greatest radius δ_{max} comprises them all, thus all the eigenvalues are inside the disk

$$D(G) = \{z \in \mathbb{C} : |z - \delta_{max}(G)| \leq \delta_{max}(G)\}.$$

□

The Laplacian matrix has thus many interesting properties directly linked to the topology of the graph which is encoding. We are now ready to state the main dynamical properties of the consensus protocol (2.2) described by system 2.3.

Theorem 2.1.3 *Consider a network of single integrators $\dot{x}_i = u_i$. Let the network be described by a quasi strongly connected graph $G = (V, E)$, then protocol (2.2) with initial condition $x(0) = x_0$ solves the consensus problem*

$$\forall i, j \in V \quad \lim_{t \rightarrow \infty} \|x_i(t) - x_j(t)\| = 0,$$

and converges to $\alpha \mathbf{1}_n$ with $\alpha \in \mathbb{R}$.

If furthermore G is balanced then $\alpha = \frac{\mathbf{1}^T x(0)}{n}$.

Proof: If G is strongly connected, then $\mathcal{L}(G)$ has a single null eigenvalue to which corresponds the right eigenvector $\mathbf{1}_n$. Since the eigenvalues of $\mathcal{L}(G)$ except one have all real part greater than zero 2.1.2, it follows that system $\dot{x} = -\mathcal{L}x$ (2.3) is marginally stable and converges inside the invariant subspace $S = \{x \in \mathbb{R}^n : x = \alpha \mathbf{1}_n, \alpha \in \mathbb{R}\}$. If G is balanced then $\mathbf{1}_n^T \mathcal{L} = \mathbf{0}$, it follows that $\mathbf{1}_n^T x(t) = \mathbf{1}_n^T x(0)$, thus $\alpha \mathbf{1}_n^T \mathbf{1}_n = \mathbf{1}_n^T x(0)$ and finally $\alpha = \frac{\mathbf{1}_n^T x(0)}{n}$. \square

Many extensions have been proposed to the consensus problem, one of the most significant is the study of the behavior of the consensus protocol (2.2) for switching network topologies (43, 44, 45, 46). Let us consider a network with dynamic topology described by a time varying graph $G(t) = (V, \mathcal{E}(t))$ where $\mathcal{E}(t) : \mathbb{R} \rightarrow E$ is the time-varying edge set where any given edge $e_{i,j}$ between nodes i and j exists at time t only if $e_{i,j} \in \mathcal{E}(t)$. Under this assumption the following result holds.

Theorem 2.1.4 *Consider a network of single integrators $\dot{x}_i = u_i$. Let the network be described by a time-varying graph $G(t) = (V, \mathcal{E}(t))$ and local interaction protocol (2.2) with initial condition $x(0) = x_0$. Then, if $\forall t \geq 0$ there exists $T > 0$ such that*

$$\mathcal{G}(t+T, t) = \bigcup_{t'=t}^T G(t'),$$

is quasi-strongly connected, then

$$\forall i, j \in V \quad \lim_{t \rightarrow \infty} \|x_i(t) - x_j(t)\| = 0,$$

and the network converges to $\alpha \mathbf{1}_n$ with $\alpha \in [\min_{i \in V}(x_i), \max_{i \in V}(x_i)]$.

Proof: The result follows from the fact that $V(x) = \max(x_1, \dots, x_n) - \min(x_1, \dots, x_n)$ is a valid Lyapunov function for system (2.3). See (47) for an exhaustive proof for the more general case of weighted interaction links. \square

The most general result for agreement problems in continuous time with a switching topology is the one proposed by Z. Lin et al. (46), under the assumption that the non-linear interaction protocol is locally Lipschitz and satisfies the *strict sub-tangentiality condition*. The authors proved that a graph-theoretical necessary and sufficient condition to ensure convergence to the agreement or consensus state is that the network topology is uniformly quasi-strongly connected (UQSC).

2.2 Consensus in Discrete-time

Let $G = \{V, E\}$ be a graph that describes the topology of the network of agents. Assume that each agent is a single discrete time integrator with dynamics:

$$x_i(t+1) = x_i(t) + u_i,$$

where $x_i, u_i \in \mathbb{R}$. This simple model is useful to describe the state update in sensor or computer networks. The state x represents some parameter on which the network of sensors has to agree, for instance the average temperature of the environment. Now let us consider the following linear interaction protocol (15):

$$u_i = \varepsilon \sum_{j \in N_i} (x_j - x_i), \quad (2.4)$$

with $|\varepsilon| \leq \max_{i \in V} |N_i|$. Each agent's dynamics thus become:

$$x_i(t+1) = x_i(t) + \varepsilon \sum_{j \in N_i} (x_j - x_i).$$

The global system dynamics is linear and can be compactly represented by:

$$x(t+1) = Px, \quad (2.5)$$

where $P = I - \varepsilon\mathcal{L}(G)$.

If $|\varepsilon| < \frac{1}{\max_{i \in V} |N_i|} = \frac{1}{\delta_{max}(G)}$, then P is a stochastic matrix by construction. If G is balanced then P is doubly stochastic. Clearly, the dynamical properties of system (2.5) are strictly linked to the topological properties of G . These links have been explored in the theory of Markov chains, where P represents the state transition probability and the generic state $x_i(t)$ represents the probability that the state of the underlying Markov process equals x_i at time t . In our framework, though formally equivalent, we are interested to link the eigenvalues of P as function of the eigenvalues of \mathcal{L} . In particular it can be noticed that by construction both P and \mathcal{L} share the same set of eigenvectors, denoting $\lambda_1, \lambda_2, \dots, \lambda_n$ the eigenvalues of \mathcal{L} and $\mu_1, \mu_2, \dots, \mu_n$ the eigenvalues of P we have that the following relation holds:

$$\mu_i = 1 - \varepsilon\lambda_i. \quad (2.6)$$

System 2.5 has thus a structural eigenvalues $\mu_1 = 1$ (formally equivalent to the trivial unitary eigenvalue of a Markov chain), which if G is strongly connected, is unique. Since system 2.5 evolves in discrete time, it solves the consensus problem only if G is strongly connected and all the eigenvalues have module strictly less than 1 except one structural eigenvalue in 1. From (2.6) and 2.1.2 it can be shown that all the eigenvalues of P are inside the disk:

$$D(G) = \{z \in \mathbb{C} : |z - \varepsilon\delta_{max}(G)| \leq \varepsilon\delta_{max}(G)\},$$

centered at $z = 1 - \varepsilon\delta_{max}(G) + j0$. Thus all the eigenvalues are inside the unitary disk if $\varepsilon\delta_{max}(G) < 1$ which becomes $\varepsilon < \frac{1}{\delta_{max}(G)}$. Converges toward the average of the initial state is achieved if G is balanced, namely if $\mathbf{1}_n^T P = \mathbf{1}_n^T$.

Many authors proposed different protocols for consensus in discrete time. In this section we presented the one in (48)

$$x_i(t+1) = x_i(t) + \varepsilon \sum_{j \in N_i} (x_j - x_i),$$

its peculiarity is its dependence from the parameter ε and its strict link with the Laplacian matrix.

In (49) the following protocol:

$$x_i(t+1) = \frac{1}{|N_i|} \sum_{j \in N_i} (x_j - x_i),$$

was used to study formation control in multi-vehicle systems.

In (9) the following protocol was proposed to model the local interaction between self propelled particles:

$$x_i(t+1) = \frac{1}{|N_i|+1} (x_i(t) + \sum_{j \in N_i} (x_j - x_i)),$$

this protocol, in which the state represents the heading the i -th particle, was shown to produce flocking as emergent behavior if interaction noise was under some maximum threshold. In (50) convergence to a consensus state was proven, later in (51) it has been pointed out that this famous protocol is a special case of the one in (52).

2.3 Consensus based on Gossip

"Gossiping" is one of the more natural way of human communication. It is uncoordinated, unsupervised, distributed in nature and asynchronous in time. Being one of the simplest forms of communication it is one of the cheapest to mimic by wireless devices. A communication based on "Gossiping" involves short point-to-point data transfers (usually just one data packet) with no packet routing inside the network. Sensor networks and networks of mobile vehicles are the applications that best exploit the advantages of gossiping due to the higher cost of other types of communications. In these applications the network topology changes rapidly since the nodes are "moving" inside an unknown environment, the environment is changing in an unpredictable way or both. Due to fast topology changes, data routing becomes expensive since to build routing tables at the nodes the knowledge of the full topology of the network is required but frequently obsolete.

Furthermore, in sensor networks or networks of mobile vehicles, the objective is to implement distributed filters for state estimation purposes (53, 54, 55) or distributed control algorithms to achieve behaviors such as flocking (6, 7). These kinds of models, while satisfying the constraints of the distributed information

flow, are often designed as continuous or discrete time systems which require each agent to synchronously update its state as function of the state of its neighbors. This kind of synchronization requirement is the main source of unreliability in the implementation of such algorithms on wireless networks subject to packet-drops and fast topology changes, furthermore since the complexity of communication synchronization grows linearly with the network size, it further constraints the scalability of the system.

To mitigate these problems, a significant amount of attention has been devoted by the research community to distributed algorithms based on "gossip" as mean of information exchange between the nodes.

Definition 2.3.1 (Gossip Algorithm) *Let $G = (V, E)$ be an undirected graph, with set of nodes $V = \{1, \dots, n\}$ and set of edges $E \subseteq \{V \times V\}$. Let $(t) : \mathbb{R} \rightarrow (i, j) \in E$ be a time-varying edge selection process, at given time instants t_0, t_1, \dots, t_k , an edge is selected $(t) = (i, j)$. Let $S_i(t)$ represent the state of the generic node i at time t . Let \mathcal{R} be a set of pairwise local interaction rules such that*

$$\begin{cases} S_i(t_{k+1}) = \mathcal{R}(S_i(t_k), S_j(t_k)), \\ S_j(t_{k+1}) = \mathcal{R}(S_i(t_k), S_j(t_k)). \end{cases}$$

An algorithm based on gossip, is a set of rules \mathcal{R} to update the state of the nodes V of the network $G = \{V, E\}$ that are applied following the edge selection process (t) . ■

A significant amount of literature sometimes refers to "consensus algorithms based on gossip" simply as "gossip algorithms" since gossiping has been mainly exploited to solve instances of consensus problems (37, 39, 56, 57, 58, 59).

One of the most studied model for gossip based consensus was proposed in (56, 57) where the authors study the consensus on the average problem in a sensor network based on a gossip algorithm.

Let $x \in \mathbb{R}$ represent the state of a sensor network where the generic element $x_i(t)$ represent the current state of node i . At time $t = 0$ the sensor nodes make a measurement, for instance temperature, and start a consensus algorithm based on gossip to compute the average value of the measured parameter in the network.

At any given instant of time, following a random edge selection process (t) , nodes i and j interact and update their state with their average:

$$\begin{cases} x_i(t_{k+1}) = \frac{x_i(t_k) + x_j(t_k)}{2}, \\ x_j(t_{k+1}) = \frac{x_i(t_k) + x_j(t_k)}{2}. \end{cases} \quad (2.7)$$

Let e_i be the i -th column of the $n \times n$ identity matrix I . By defining

$$W_{ij} = I - \frac{(e_i - e_j)(e_i - e_j)^T}{2},$$

and

$$W(t) = \{W_{ij} : (t) = (i, j)\},$$

we can model the network state evolution as:

$$x(t_{k+1}) = W(t_k)x(t_k). \quad (2.8)$$

Each instant of time t_k , $W(t_k)$ is a doubly stochastic matrix that satisfies $W(t_k)\mathbf{1} = \mathbf{1}$ and $\mathbf{1}^T W(t_k) = \mathbf{1}^T$. Furthermore $W(t_k)$ is para-contractive, namely $\forall t_k, x \in \mathbb{R}^n$, $\|W(t_k)x\| \leq \|x\|$. The evolution of the network dynamics is thus described by:

$$x(t_k) = \prod_{\tau=1}^k W(t_\tau)x(t_\tau).$$

The network's state converges to the average consensus state for any initial condition if

$$\lim_{\tau \rightarrow \infty} \phi(t_\tau) = \lim_{k \rightarrow \infty} \prod_{\tau=1}^k W(t_\tau) = \frac{\mathbf{1}\mathbf{1}^T}{n}. \quad (2.9)$$

The general problem of determining whether the infinite product of matrices taken at random from set converges to a finite limit is still open, a number of sufficient computable conditions exist if the matrices taken in consideration have particular properties (36, 60). If the set of matrices is para-contractive then it has been proven that the series converges to the intersection of the invariant spaces of the matrices (60).

In our case, since all the matrices taken into consideration are doubly stochastic, $\phi(t_\tau)$ is doubly stochastic as well for any τ , thus the trajectories of system (2.8) are bounded. If the edge selection process (t) selects edges independently across time, assuming that each edge in E has a strictly positive probability of being chosen at each instant of time and that G is a connected undirected graph, then we know that system (2.8) converges to a finite limit

$$\lim_{k \rightarrow \infty} x(t_k) = \alpha \mathbf{1}_n, \quad (2.10)$$

with $\alpha \in [\max_i x_i(0), \min_i x_i(0)]$.

Now we study the expected evolution of system (2.8) by taking the expectation on both sides of (2.8):

$$\begin{aligned} E[x(t_{k+1})] &= E[W(t_k)x(t_k)] \\ &= E[W(t_k)]x(t_k) \\ &= \bar{W}x(t_k). \\ E[x(t_k)] &= \bar{W}^k x(t_0). \end{aligned} \quad (2.11)$$

Where $E[W(t_k)] = \bar{W} = \frac{1}{|E|} \sum_{(i,j) \in E} W_{ij}$.

Since \bar{W} is a convex combination of doubly stochastic matrices, it is doubly stochastic itself, thus $\rho(\bar{W}) = 1$ and the second largest eigenvalue of \bar{W} is $\lambda_2(\bar{W}) < 1$ if G is a connected graph. Since $W(t_k)\mathbf{1} = \mathbf{1}$, the system converges to the equilibrium space given by the right eigenvector corresponding to the unitary eigenvalue which is $\alpha \mathbf{1}_n$. Furthermore since $\mathbf{1}^T W(t_k) = \mathbf{1}^T$, we have that $\mathbf{1}^T \alpha \mathbf{1} = 1$. It follows that

$$\lim_{k \rightarrow \infty} x(t_k) = \frac{\mathbf{1}\mathbf{1}^T}{n} x(0).$$

Convergence rate

The converge rate of system (2.8) is determined by the stochastic edge selection process, here we take the definition of convergence rate proposed in (56), namely:

$$T_{ave}(\varepsilon, \bar{W}) = \sup_{x(0)} \inf \{t : Pr \left(\frac{\|x(t) - x_{ave} \mathbf{1}_n\|}{\|x(0)\|} \geq \varepsilon \right) \leq \varepsilon \}, \quad (2.12)$$

the convergence time is then the smallest time system (2.8) takes to get a ε close to $x_{ave}\mathbf{1}_n$ with high probability, disregarding the initial condition.

In (56) it was shown that for any gossip algorithm based on averaging matrices which converges in expectation, the following result holds:

$$T_{ave}(\varepsilon, \overline{W}) \leq \frac{3\log(\varepsilon^{-1})}{\log\left(\frac{1}{\lambda_2(\overline{W})}\right)} \leq \frac{3\log(\varepsilon^{-1})}{1 - \lambda_2(\overline{W})}.$$

This upperbound puts in evidence that the spectral gap $1 - \lambda_2(\overline{W})$ affects directly the expected convergence time of the gossip algorithm.

2.4 Quantized Consensus

In this section we review some consensus algorithms extended to the case in which the network state is quantized.

Any implementation of a consensus algorithm with digital processing units needs to deal with quantization. The reason is simple, all the algorithms that we have presented so far assume that no external noise or round-off errors are considered, unfortunately all the proposed systems (??) are marginally stable since all have a structural invariant subspace $\{x : x = \alpha\mathbf{1}, \forall \alpha \in \mathbb{R}\}$ in which every point is an equilibrium point. This means that whenever noise or round-off errors fall inside such subspace they effectively change the equilibrium point to which the system is converging. This is why the study of consensus in the quantized case is significant for any real implementation.

In (29), it was originally presented a solution to the quantized consensus problem taking inspiration from the literature in distributed load balancing for networks of processors (27, 28) since it turns out to be formally a similar problem.

Let us consider n agents whose state is quantized $x_i \in \mathbb{Z}$ and consider a network described by an undirected graph $G = (V, E)$. We assume a communication model based in gossip, in which an edge selection process $e(t)$ that at random selects at each instant time which edge in the network is active. In (29) a class of distributed averaging algorithms, which was defined as quantized gossip algorithms, was considered.

Say edge (i, j) is selected at time t , and let $D_{i,j}(t) = |x_i(t) - x_j(t)|$. Then, if $D_{i,j}(t) = 0$, we leave the values unchanged, $x_i(t+1) = x_i(t)$, $x_j(t+1) = x_j(t)$.

If $D_{ij}(t) \geq 1$, then it is required that the state update abide the following three rules:

1. (P1) - $x_i(t+1) + x_j(t+1) = x_i(t) + x_j(t)$,
2. (P2) - if $D_{ij}(t) > 1$ then $D_{ij}(t+1) < D_{ij}(t)$,
3. (P3) - if $D_{ij}(t) = 1$ and (without loss of generality) $x_i(t) < x_j(t)$, then $x_i(t+1) = x_j(t)$ and $x_j(t+1) = x_i(t)$. Such an update is referred to as "swap".

Let $L = \lfloor \frac{1}{n} \sum_{i=1}^n x_i \rfloor$ and $S = \sum_{i=1}^n x_i(0)$. Define the *quantized consensus state* as

$$\mathcal{S} = \{x : x_i \in [L, L+1], \forall i \in V, \sum_{i=1}^n x_i = S\}. \quad (2.13)$$

Then, the following result holds

Theorem 2.4.1 *For any given initial vector $x(0)$, if the values $x(t)$ are updated using a quantized gossip algorithm, then*

$$\lim_{t \rightarrow \infty} Pr(x(t) \in \mathcal{S}) = 1,$$

where the set \mathcal{S} is defined as in (2.13).

Proof: See (29) for a detailed proof. □

As standard example of algorithm labeled as quantized gossip algorithm we report the one in (29).

Algorithm 1 (Standard quantized gossip)

1. Let $t = 0$ and $x(0) = x_0$.
2. Select an edge $(i, j) \in E$ at random.
3. Let

$$\begin{aligned} x_i(t+1) &= \lfloor \frac{x_i(t) + x_j(t)}{2} \rfloor, \\ x_j(t+1) &= \lceil \frac{x_i(t) + x_j(t)}{2} \rceil, \end{aligned}$$

4. Let $t = t + 1$ and go back to step 2.

It is easy to show that for algorithm 1 properties (P1), (P2) and (P3) (1) hold. Particularly relevant to the convergence properties of algorithm 1 is property (P3), it holds in general if each edge has a strictly positive probability being chosen (regarding the notation of algorithm 1 it also assumes strictly positive probability of choosing either (i, j) or (j, i)). A characteristic of this algorithm is that even after the network state has reached the quantized consensus state \mathcal{S} , it does not settle in a equilibrium point. Instead, due to the dynamics of the state "swaps" between the nodes, it may persistently change while being inside \mathcal{S} .

Several authors have devised new strategies to address in the quantized consensus problem. In (26) a consensus strategy in which the state of agents is a real number but the exchanged data are symbols and not real numbers is presented, the approach is based on a logarithmic quantizer based state estimator.

In (61) a distributed algorithm in which the nodes utilize probabilistically quantized information, i.e., dithered quantization, to communicate with each other is presented. The algorithm is a dynamical system that generates sequences achieving a consensus at one of the quantization values almost surely.

Chapter 3

Discrete Consensus on Heterogeneous Networks

3.1 Introduction

In several applicative domains related to consensus problems the assumption that the state of each node is a continuous variable is clearly an oversimplified assumption, and it is necessary to explicitly take into account its discrete nature. This defines a new framework called *quantized consensus* (26, 29, 61) where the state only takes nonnegative integer values. It has been observed in (29) that reaching a consensus under this quantization constraint is equivalent to determining a balanced assignment of identical tasks to nodes. In this chapter, we generalize this approach assuming that the tasks to be assigned to nodes may not be identical. Thus we assume that the state of each node is not described by an integer number, but by a collection of objects each one with its own integer weight. We call this framework *discrete consensus*.

One of the most important classes of problems that can be formulated in terms of discrete consensus is given by *task assignment*. Tasks may be generic objects, e.g., spatial locations, network resources, or classifications. In particular, recent advances in communication and computation have allowed efficient solutions to the problem of task assignment (62). Both on-line (63, 64) and off-line (65) approaches have been proposed: on-line approaches keep into account the variations of the task environment, while off-line approaches assume that the task environ-

ment keeps constant thus a solution can be computed in advance. Processor assignment and computation of an optimal execution sequence for multiversion software is an example of this kind of problems (66).

In this chapter, whose results appeared in (67), we will focus on a problem of discrete consensus over heterogeneous networks. To provide a more intuitive interpretation to the considered physical variables, our problem formulation will be given in terms of task assignment although the approach is general. We assume that a given set of tasks, that may have different weight, should be assigned to agents (nodes). Networks are denoted as heterogeneous because nodes may have different speeds. The consensus problem for this type of nets, as far as we know, has not received much attention in the control literature.

Our goal is that of determining, using *consensus* algorithms based on *gossip* (29, 38), the solution that minimizes the maximum execution time over nodes. It is based on the recent work by Kashyap *et al.* (29) and on our previous results in (34) where homogeneous networks have been considered.

Note that due to the discrete nature of tasks and to the assumption that tasks may have arbitrary weights, the optimality of the solution is not guaranteed. As discussed in (34) this is not related to our particular approach but is intrinsic in the nature of gossip, that implements at each step a pairwise optimization, and does not always yield an optimal solution. However, we prove that there exists a bounded set that contains the optimal solution that is always reachable and we study the convergence properties and the convergence time to this bounded set.

As mentioned in the literature (29, 34), in the case of discrete consensus to ensure good convergence properties it is necessary to enrich the gossip algorithm with an appropriate *swapping rule*. Whenever a balancing between two nodes is not possible, the swap “shakes” the network configuration to redistribute the load and allows loads composed by discrete tasks to travel in the network, reaching a situation in which a new balancing may occur.

When swaps are performed, the maximum average convergence time depends on the average meeting time of agents performing a random walk in a graph. We show that this can always be computed numerically, modeling the swap process with a Markov Chain with a single absorbing state that represents the meeting of the agents. We also discuss two particular net structures (fully connected networks and networks with ring topology) for which it is possible to compute

the average meeting time analytically.

3.2 Problem description

We consider a heterogeneous network of n nodes whose connections can be described by an undirected connected graph $\mathcal{G} = (V, E)$, where $V = \{1, 2, \dots, n\}$ is the set of nodes and $E \subseteq V \times V$ is the set of edges.

We assume that K indivisible tasks should be assigned to the nodes, and an integer weight c_j , $j = 1, \dots, K$, is associated to each task. We define a weight vector $c \in \mathbb{N}^K$ whose j -th component is equal to c_j , and n binary vectors $y_i \in \{0, 1\}^K$ such that: $y_{i,j} = 1$ if the j -th task is assigned to node i , $y_{i,j} = 0$ otherwise.

To each node $i \in V$ is allocated a *load* $x_i = c^T y_i$ consisting in the sum of the costs of tasks assigned to node i that must be processed.

The *speed factor*, denoted γ_i , represents the amount of load that can be processed in a time unit by node i . In the following we denote γ_{\min} the smallest speed in the network (clearly $\gamma_{\min} > 0$), and c_{\max} the maximum weight of tasks in the network.

The task assignment we are looking for is the one that minimizes the *maximum execution time*, starting from any initial condition. Namely, if we define the load and speed vectors $x = [x_1 \ x_2 \ \dots \ x_n]^T$, $\gamma = [\gamma_1 \ \gamma_2 \ \dots \ \gamma_n]^T$ and $\Gamma = \text{diag}(\gamma)$, we would like to minimize the following objective function:

$$f(x) = \max_{i=1, \dots, n} \frac{x_i}{\gamma_i} = \|\Gamma^{-1}x\|_{\infty} \quad (3.1)$$

under the assumption that the total load remains constant, namely $\mathbf{1}^T x = \mathbf{1}^T x(0)$, where $x(0)$ represents the initial load configuration.

Denoting $Y(t) = [y_1(t) \ y_2(t) \ \dots \ y_n(t)]$ the state of the network at time t , a centralized optimal solution to this problem can be determined solving the following integer programming problem with binary variables:

$$\begin{cases} \min V & = \ \|c^T Y \Gamma^{-1}\|_{\infty} \\ \text{s.t.} & \ Y \mathbf{1} = \mathbf{1} \\ & \ y_{i,j} \in \{0, 1\} \quad \forall i = 1, \dots, n; \quad j = 1, \dots, K. \end{cases} \quad (3.2)$$

We denote Y^* (resp., V^*) the optimal solution (resp., the optimal value of the performance index) of Problem (3.2).

3.3 Proposed algorithm

We first define a task exchange process between two adjacent nodes that, while not changing the value of the objective function, modifies the load configuration.

Definition 3.3.1 (Swap) *Let us consider two nodes i and r incident on the same edge. Let $\mathcal{K}_i(t)$, resp. $\mathcal{K}_r(t)$, be the set of tasks contained in node i , resp. r , at time t .*

Let us call swap the operation that moves the tasks in $\mathcal{K}_i(t)$ to r , and the tasks in $\mathcal{K}_r(t)$ to i at time $t + 1$, reaching the distribution

$$\mathcal{K}_i(t + 1) = \mathcal{K}_r(t), \quad \text{and} \quad \mathcal{K}_r(t + 1) = \mathcal{K}_i(t),$$

provided that the objective function locally defined for the two nodes does not change, i.e.,

$$\max \left\{ \sum_{j \in \mathcal{K}_i(t+1)} \left(\frac{c_j}{\gamma_i} \right), \sum_{j \in \mathcal{K}_r(t+1)} \left(\frac{c_j}{\gamma_r} \right) \right\} = \max \left\{ \sum_{j \in \mathcal{K}_i(t)} \left(\frac{c_j}{\gamma_i} \right), \sum_{j \in \mathcal{K}_r(t)} \left(\frac{c_j}{\gamma_r} \right) \right\}.$$

■

We denote $\hat{\mathcal{K}}_{ir}(t) = \mathcal{K}_i(t) \cup \mathcal{K}_r(t)$ the set of tasks present in nodes i and r at time t . We define $\hat{c} = c \uparrow \hat{\mathcal{K}}_{ir}(t)$ the projection of c on $\hat{\mathcal{K}}_{ir}(t)$, namely a vector whose elements are the weights of the tasks present in nodes i and r at time t . Using the same notation we define two binary vectors $\hat{y}_i = y_i \uparrow \hat{\mathcal{K}}_{ir}(t)$ and $\hat{y}_r = y_r \uparrow \hat{\mathcal{K}}_{ir}(t)$, in other words each vector has a number of elements equal to the number of tasks locally present in the nodes.

Algorithm 2 (Gossip Algorithm with discrete tasks)

1. Let $t = 0$.

2. Select an edge $\{i, r\}$ at random.

3. Solve the integer programming problem (IPP):

$$\left\{ \begin{array}{l} k^* = \min k \\ \text{s.t.} \quad \frac{\hat{c}^T \hat{y}_i}{\gamma_i} \leq k \\ \frac{\hat{c}^T (\mathbf{1} - \hat{y}_i)}{\gamma_r} \leq k \\ k \in \mathbb{R}_+ \cup \{0\}, \\ \hat{y}_i \in \{0, 1\}^{|\hat{\mathcal{K}}_{ir}(t)|} \end{array} \right. \quad (3.3)$$

4. **If** $k^* < \max \left\{ \frac{\hat{c}^T \hat{y}_i(t)}{\gamma_i}, \frac{\hat{c}^T (\mathbf{1} - \hat{y}_i(t))}{\gamma_r} \right\}$ **then**

let $\hat{y}_i(t+1) = \hat{y}_i$ and $\hat{y}_r(t+1) = \mathbf{1} - \hat{y}_i$,

else execute a swap if possible.

5. Let $t = t + 1$ and goto step 2. ■

In practice IPP (3.3) provides the task assignment that minimizes the execution time at the two nodes. If the resulting assignment is better than the previous one, tasks are assigned accordingly, otherwise a swap is executed if possible.

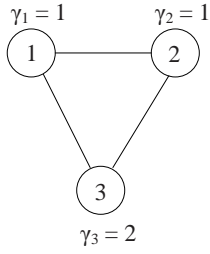
The swap allows to overcome several blocking conditions: anytime the network reaches a local minimum of the objective function the swap “shakes” the network ensuring convergence within some precise bounds (see Theorem 3.4.5).

Example 3.3.2 Let us consider the fully connected¹ net in Fig. 3.1 composed by 3 nodes with speeds $\gamma_1 = \gamma_2 = 1$ and $\gamma_3 = 2$. Assume that it contains 10 tasks whose weights are equal to $c_1 = 1$, $c_2 = c_3 = 2$, $c_4 = c_5 = 3$, $c_6 = 4$, $c_7 = 5$, $c_8 = 6$, $c_9 = 7$ and $c_{10} = 10$ and let the initial configuration be

$$\mathcal{K}_1(0) = \{6, 10\}, \quad \mathcal{K}_2(0) = \emptyset, \quad \mathcal{K}_3(0) = \{1, 2, 3, 4, 5, 7, 8, 9\}.$$

Using Algorithm 2, we obtain the optimal task assignment in four steps, as summarized in the Fig. 3.1. In particular, the first line of the table summarizes the

¹A network is *fully connected* if there is an arc from each node to any other one.



t	edge	node 1	node 2	node 3	$V(Y)$
0		4, 10		1, 2, 2, 3, 3, 5, 6, 7	14.5
1	{1, 3}	4, 10		1, 2, 2, 3, 3, 5, 6, 7	14.5
2	{2, 3}	4, 10	2, 7	1, 2, 3, 3, 5, 6	14
3	{1, 3}	5, 6	2, 7	1, 2, 3, 3, 4, 10	11.5
4	{2, 3}	5, 6	1, 2, 7	2, 3, 3, 4, 10	11

Figure 3.1: The network discussed in Example 3.3.2 and the results of Algorithm 2.

initial assignment to which it corresponds a value of the objective function that is equal to 14.5 (see the last column). The other lines point out the task assignment for $t = 1, 2, 3, 4$: in the second column we point out the selected edge, columns 3 to 5 point out the task assignment in the three nodes; finally, the last column points out the corresponding value of the objective function. ■

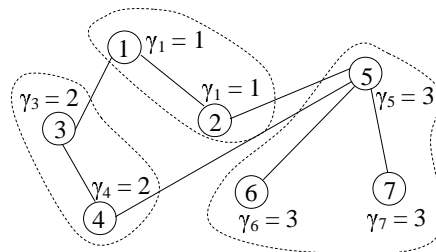
3.4 Convergence properties

The convergence properties of Algorithm 2 depend on the possibility of performing swaps.

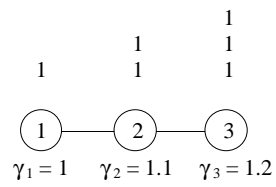
Definition 3.4.1 (Swap domain) We call swap domain $G_\gamma \subseteq G$ a connected subgraph induced by nodes with the same speed. ■

Example 3.4.2 Let us consider the network in Fig. 3.2.a that has seven nodes with three different speeds. This network can be partitioned in three different subgraphs G_1, G_2 and G_3 induced respectively by nodes $\{1, 2\}, \{3, 4\}$ and $\{5, 6, 7\}$. In this case each swap domain is connected to each other. ■

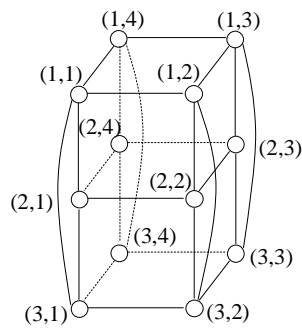
Each swap domain identifies a set of nodes where swaps may always happen. On the contrary swaps between adjacent nodes of different domains may either be possible or not, depending on the particular tasks and on the speed of nodes.



(a)



(b)



(c)

Figure 3.2: (a) The network discussed in Example 3.4.2: (b) Network in example 3.4.7 (c) A net with a generalized ring topology where $s = 3$ and $k = 4$.

As an example, if we consider the network in Example 3.3.2 a swap among nodes 1 and 3, that belong to different swap domains, is not allowed at time $t = 1$ because it would lead to an increasing of the maximum execution time at the two nodes. The following example shows a scenario in which a swap among nodes belonging to different swap domains is admissible.

Example 3.4.3 *Let us assume that two nodes 1 and 2 have speed equal to $\gamma_1 = 10$ and $\gamma_2 = 11$, respectively. Moreover, let us assume that node 1 only contains one task of weight 10, while the second node contains two tasks both of weight equal to 5. The corresponding maximum execution time is equal to 1. Now, no tasks exchange may occur that leads to a better tasks assignment, while a swap may happen keeping unaltered the maximum execution time at the two nodes. ■*

It is relevant to note that the definition of “swap domain” is embedded in the graph topology: the nodes don’t need to know in which domain they are or even that any domain exists.

Definition 3.4.4 *We call final set*

$$\tilde{\mathcal{Y}} = \{Y = [y_1 \ y_2 \ \cdots \ y_n] \mid \left| \frac{c^T y_i}{\gamma_i} - \frac{c^T y_r}{\gamma_r} \right| \leq \frac{c_{\max}}{\gamma_{\min}}, \quad \forall i, r \in \{1, \dots, n\}\} \quad (3.4)$$

i.e., the set of configurations such that, for any couple of nodes $i, r \in V$, the difference among their execution times is at most equal to the ratio c_{\max}/γ_{\min} . ■

Theorem 3.4.5 *Let $Y(t)$ be the matrix that summarizes the task assignment resulting from Algorithm 2 at the generic time t . If each swap domain is connected to each other, it holds $\lim_{t \rightarrow \infty} \Pr(Y(t) \in \tilde{\mathcal{Y}}) = 1$ where $\Pr(Y(t) \in \tilde{\mathcal{Y}})$ denotes the probability that $Y(t) \in \tilde{\mathcal{Y}}$.*

Proof: We define a Lyapunov-like function

$$V(t) = [V_1(t), V_2(t)] \quad (3.5)$$

consisting of two terms. The first one is equal to the objective function of (3.2), namely

$$V_1(t) = \|c^T Y(t) \Gamma^{-1}\|_\infty.$$

The second one is a measure of the number of nodes whose execution time is equal to

$$\|c^T Y(t) \Gamma^{-1}\|_\infty,$$

i.e.,

$$V_2(t) = \left| \arg \max_{i=1, \dots, n} \frac{c^T y_i(t)}{\gamma_i} \right|.$$

Note that we impose a lexicographic ordering on the performance index, i.e., $V = \bar{V}$ if $V_1 = \bar{V}_1$ and $V_2 = \bar{V}_2$; $V < \bar{V}$ if $V_1 < \bar{V}_1$ or $V_1 = \bar{V}_1$ and $V_2 < \bar{V}_2$.

The proof is based on three arguments.

(1) We first prove that $V(t)$ is a non increasing function of t .

This is trivially true when a swap is executed, since in such a case $V(t+1) = V(t)$.

Consider the case in which the selected nodes i and r balance their load. It holds

$$\max \left\{ \frac{c^T y_i(t+1)}{\gamma_i}, \frac{c^T y_r(t+1)}{\gamma_r} \right\} < \max \left\{ \frac{c^T y_i(t)}{\gamma_i}, \frac{c^T y_r(t)}{\gamma_r} \right\},$$

hence three different cases may happen.

(a) One of the selected nodes is the only node in the network such that its execution time is equal to $\|c^T Y \Gamma^{-1}\|_\infty$. In such a case $V_1(t+1) < V_1(t)$ hence $V(t+1) < V(t)$.

(b) One of selected nodes is such that its execution time is equal to $\|c^T Y(t) \Gamma^{-1}\|_\infty$ but there exists at least one other node in the network with the same execution time. In such a case $V_1(t+1) = V_1(t)$ and $V_2(t+1) = V_2(t) - 1$, hence $V(t+1) < V(t)$.

(c) The execution time of both the selected nodes is smaller than $\|c^T Y(t) \Gamma^{-1}\|_\infty$. In such a case $V(t+1) = V(t)$.

(2) Secondly, we observe that, if the current configuration is outside the final set $\tilde{\mathcal{Y}}$, then there exists at least one node whose execution time is equal to

$\|c^T Y(t) \Gamma^{-1}\|_\infty$ that could balance his load with (at least) one other node if they were incident on the same arc: this would reduce function $V(t)$ (see cases (a) and (b) of the previous item).

To prove this we observe that if the current configuration is outside the final set $\tilde{\mathcal{Y}}$, then there exists (at least) one couple of nodes i and r such that

$$\frac{c^T y_i(t)}{\gamma_i} - \frac{c^T y_r(t)}{\gamma_r} > \frac{c_{\max}}{\min\{\gamma_i, \gamma_r\}} \quad (3.6)$$

where $\frac{c^T y_i(t)}{\gamma_i}$ is equal to the maximum execution time. If we move a task $c_j \leq c_{\max}$ from node i to node r we have: $c^T y_i(t+1) = c^T y_i(t) - c_j$, and $c^T y_r(t+1) = c^T y_r(t) + c_j$. Now

$$\frac{c^T y_i(t+1)}{\gamma_i} = \frac{c^T y_i(t) - c_j}{\gamma_i} < \frac{c^T y_i(t)}{\gamma_i} \quad (3.7)$$

and

$$\frac{c^T y_r(t)}{\gamma_r} + \frac{c_j}{\gamma_r} \leq \frac{c^T y_r(t)}{\gamma_r} + \frac{c_{\max}}{\min\{\gamma_i, \gamma_r\}} < \frac{c^T y_i(t)}{\gamma_i}$$

where the second inequality follows from assumption (3.6); thus

$$\frac{c^T y_r(t+1)}{\gamma_r} = \frac{c^T y_r(t) + c_j}{\gamma_r} < \frac{c^T y_i(t)}{\gamma_i}. \quad (3.8)$$

By (3.7) and (3.8) it follows that

$$\max \left\{ \frac{c^T y_i(t+1)}{\gamma_i}, \frac{c^T y_r(t+1)}{\gamma_r} \right\} < \max \left\{ \frac{c^T y_i(t)}{\gamma_i}, \frac{c^T y_r(t)}{\gamma_r} \right\}.$$

(3) Finally, we observe that being each swap domain connected to each other, there exists a series of swaps that lead to a configuration in which the loads of the two nodes identified in the previous item are adjacent and the arc between them is selected. This happens with probability 1 as t goes to infinity. \square

Remark 3.4.6 Theorem 3.4.5 characterizes the convergence properties of Algorithm 2 in terms of a finite set $\tilde{\mathcal{Y}}$. This obviously does not imply that an optimal

task assignment is achieved. As shown in (34) this is not a limitation of the particular algorithm. To reach consensus an optimization involving more than two nodes at the same time may be necessary. ■

Finally the following example shows that if each swap domain is not connected to each other, then the convergence set $\tilde{\mathcal{Y}}$ may not be reached by Algorithm 2.

Example 3.4.7 Let us consider the network in Fig. 3.2.b where $\gamma_1 = 1$, $\gamma_2 = 1.1$, $\gamma_3 = 1.2$. Tasks with cost $c_1 = c_2 = \dots c_6 = 1$ are assigned such that

$$\mathcal{K}_1(0) = \{1\}, \quad \mathcal{K}_2(0) = \{1, 1\}, \quad \mathcal{K}_3(0) = \{1, 1, 1\}.$$

It can be seen that the network is in a blocking configuration where no task exchange may happen according to Algorithm 2. As a result the convergence set $\tilde{\mathcal{Y}}$ is not reached being

$$\left| \frac{c^T y_1}{\gamma_1} - \frac{c^T y_3}{\gamma_3} \right| = \left| 1 - \frac{3}{1.2} \right| = 1.5 > \frac{c_{\max}}{\gamma_{\min}} = 1.$$

On the other hand if node 3 and node 1 were connected, then node 3 could exchange one task with node 1 and achieve a configuration included in $\tilde{\mathcal{Y}}$. ■

We now show with an example the inherent limitations of pairwise balancing based on gossip.

Example 3.4.8 Let us consider a three node fully connected network. The initial configuration of the network is

$$\mathcal{K}_1(0) = \{1\}, \quad \mathcal{K}_2(0) = \{2, 3, 4\}, \quad \mathcal{K}_3(0) = \{5, 6\}$$

where $c_1 = 7, c_2 = c_3 = c_4 = 3, c_5 = c_6 = 5$. An optimal configuration in terms of load balancing is

$$\mathcal{K}_1^* = \{1, 2\}, \quad \mathcal{K}_2^* = \{3, 5\}, \quad \mathcal{K}_3^* = \{4, 6\}.$$

However, it cannot be reached, because neither a swap nor a load balancing is possible between any two nodes. ■

The previous example highlights an important general property: an optimal load balancing with non-unitary tasks cannot always be achieved by greedy gossip algorithms that balance the load between two nodes at each step, even on a fully connected network. In fact, to reach consensus an optimization involving more than two nodes at the same time may be necessary.

Convergence time of Algorithm 2

The *convergence time* is a random variable defined for a given initial task assignment $Y(0) = Y$ as: $T_{conv}(Y) = \inf \{t \mid \forall t' \geq t, Y(t') \in \tilde{\mathcal{Y}}\}$. Thus, $T_{conv}(Y)$ represents the number of steps required at a certain execution of Algorithm 2 to reach the convergence set $\tilde{\mathcal{Y}}$ starting from a given tasks distribution. Let us firstly introduce the following notation.

- N_{\max} is the maximum number of improvements of $V(t)$ defined as in (4.7), needed by any realization of Algorithm 2 to reach the set $\tilde{\mathcal{Y}}$, starting from a given configuration.
- T_{\max} is the maximum average time between two consecutive improvements of $V(t)$ defined as in (4.7), needed by any realization of Algorithm 2, starting from a given configuration.

Using the previous notation, it follows that the *expected convergence time* is

$$\mathcal{E}[T_{conv}(Y)] \leq N_{\max} \cdot T_{\max}. \quad (3.9)$$

The following proposition provides a topology independent upper bound on N_{\max} .

Proposition 3.4.9 *Let us consider a net with n nodes and let γ be the corresponding speed vector. Let $x(0)$ be the vector representative of the initial amount of load at nodes. It holds:*

$$N_{max} \leq (n - 1) \cdot \varrho \cdot (M - m) \quad (3.10)$$

where

$$M = \|\Gamma^{-1}x(0)\|_{\infty}, \quad m = \frac{\sum_{i=1}^n x_i(0)}{\sum_{i=1}^n \gamma_i} = \frac{\mathbf{1}^T x(0)}{\mathbf{1}^T \Gamma \mathbf{1}}, \quad (3.11)$$

$$\varrho = \max_{\{i,r\} \in E} lcm\{\gamma_i, \gamma_r\},$$

and lcm denotes the least common multiple.

Proof: By definition the maximum number of improvements of $V_1 = f$ needed by any realization of Algorithm 2 to reach the set $\tilde{\mathcal{Y}}$ is smaller or equal to the ratio between the global improvement of f needed before reaching the convergence set $\tilde{\mathcal{Y}}$ starting from $x(0)$, and its minimum admissible improvement.

By Step 5 of Algorithm 2 the task assignment is updated if and only if leads to an improvement of the objective function, otherwise a swap is executed. Thus, the largest value of $f(x)$ occurs at the initial configuration and is equal to

$$M = f(x(0)) = \|\Gamma^{-1}x(0)\|_{\infty}.$$

The minimum value of $f(x)$ corresponds to the case of perfect task assignment, that in general is not achievable in the discrete case. However, a lower estimate of it is given by its optimal value in the case of infinitely divisible tasks, namely by $f(x^*)$ where

$$x^* = \alpha\gamma \quad \text{and} \quad \alpha = \frac{\mathbf{1}x(0)}{\mathbf{1}^T\Gamma\mathbf{1}}$$

. Thus, if we define $m = f(x^*) = \alpha$, then for any task assignment x it holds $m \leq f(x)$.

We also observe that the minimum load exchange is equal to 1 since all tasks have an integer weight.

Now, if we consider the generic edge $\{i, r\}$, we know that the minimum improvement of f that we may obtain when balancing this edge is equal to $1/\text{lcm}\{\gamma_i, \gamma_r\}$. As a consequence the minimum improvement of f at a generic step of Algorithm 2 is equal to $1/\varrho = 1/\max_{\{i,r\} \in E} \text{lcm}\{\gamma_i, \gamma_r\}$, where E is the set of edges.

Thus, we may conclude that the largest number of improvements of f before reaching the convergence set $\tilde{\mathcal{Y}}$ starting from $x(0)$ is at most equal to $\varrho \cdot (M - m)$.

Finally, in the worst case $n - 1$ consecutive balancing may occur before having an improvement of f , namely $n - 1$ consecutive reductions of V_2 may occur before having a reduction of $V_1 = f$. In particular, this case may happen if $n - 1$ nodes have the same execution time that is equal to the maximum one. In this case, a first balancing may occur between the only “different” node and any of the other ones. Then, a new balancing may occur between any of the remaining $n - 2$ nodes with the maximum execution time and one with a smaller execution time, and so on. \square

We now focus on T_{\max} . Evaluating T_{\max} , and hence the average convergence time (3.9), is in general a difficult issue because it is strictly related to the particular topology of the net.

In the following we consider two cases: *fully connected* networks and *generalized ring topology* nets. Similar approaches based on Markov chains can always be used to evaluate numerically an upper bound on T_{\max} for a particular net example.

Fully connected networks

Proposition 3.4.10 *Let us consider a fully connected network, and let n be the number of nodes.*

It holds

$$T_{\max} = \frac{n(n-1)}{2}. \quad (3.12)$$

Proof: The maximum average time between two consecutive balancing occurs when only one balancing is possible. Thus, if N is the number of arcs of the net, then the probability of selecting the only arc whose incident nodes may balance their load is equal to $p = 1/N$, while the average time needed to select it is equal to N . Since the network is fully connected, if n is the number of nodes, the number of arcs is $N = n(n-1)/2$ and so $T_{\max} = n(n-1)/2$. \square

Proposition 3.4.11 *If a net is fully connected, the average convergence time of Algorithm 2 is*

$$\mathcal{E}[T_{\text{conv}}(Y)] \leq \varrho \cdot (M - m) \cdot \frac{n(n-1)^2}{2} = \mathcal{O}(n^3).$$

Proof: Follows from equation (3.9) and Propositions 4.4.7 and 4.4.8. \square

Generalized ring topology

Definition 3.4.12 (Generalized ring topology) *A graph $\mathcal{G} = (E, V)$ has a generalized ring topology if it satisfies the following assumptions.*

- It is composed by s rings, each one with k nodes. The generic j -th ring R_j is a graph $R_j = (V_j, E_j)$ with $V_j = \{1, \dots, k\}$ and $E_j = \{\{i, r\} \in E \mid r = i + 1, \forall i = 1, \dots, k - 1\} \cup \{k, 1\}$.

- The same speed is associated to all nodes in the same ring, while nodes of different rings have different speeds. Thus each ring defines a different swap domain.

- Let (i, j) , with $i = 1, \dots, k$ and $j = 1, \dots, s$, be the i -th node of ring R_j . Let $\Sigma_i = \{(i, j) \in V, j = 1, \dots, s\}$ be the set of the nodes of index i in all rings. All nodes in Σ_i are fully connected, i.e., for all $i = 1, \dots, k$, there exists an edge in E that connects each node in Σ_i with any other node in Σ_i . ■

An example of a net with a generalized ring topology is reported in Fig. 3.2.c: here $s = 3$, $k = 4$.

Note that such a topology well fits with our problem for two main reasons. Firstly, it is scalable both in the number of nodes in the rings and in the number of rings (namely in the number of swap domains). Secondly, the diameter of the net, namely the maximum distance among nodes that may balance, increases with the number of nodes in the ring.

Proposition 3.4.13 *Let us consider a net with a generalized ring topology. Let s be the number of rings and $n = k \cdot s$ be the total number of nodes in the net. It holds*

$$T_{\max} \leq \frac{n^2(s+1)}{32 \cdot s} \cdot \left(\frac{n}{s} + 16\right) = \frac{k^2 s(s+1)}{32} \cdot (k+16). \quad (3.13)$$

Proof: We first observe that, due to the gossip nature of Algorithm 2 and to the random rule used to select the edges, the problem of evaluating an upper bound on T_{\max} can be formulated as the problem of finding the average meeting time of two agents walking on a graph executing a random walk. In fact, the average meeting time of the two agents may be thought as the average time of selecting an edge whose incident nodes may balance their load. Note that in general more than two edges may balance their load, thus assuming that only two agents are walking on the graph provides us an upper bound on the value of T_{\max} . In particular, the worst case in terms of meeting time occurs when the two agents are on different rings.

In the following we compute the average meeting time using discrete Markov chains assuming that the two agents walk on different rings (worst case). For the sake of simplicity, we assume that the number of nodes k in each ring is even¹.

We call distance between two agents in nodes (i, j) and (i', j') , with $j \neq j'$, $d_{i,i'} = 1 + \min\{|i - i'|, k - |i - i'|\}$, namely the number of arcs in the shortest path connecting node i with node i' . In simple words the above distance is equal to the distance between the two agents, computed as if they were in the same ring, plus 1 due to the fact that they are on different rings. This is consistent with the assumption that, in a generalized ring topology net, any node with a given index in a certain ring is connected to all the other nodes having the same index in different rings. Therefore nodes with a unitary distance are nodes within the same section Σ . Under the assumption that k is even, the maximum distance between the two agents is equal to $D = k/2 + 1$.

The Markov chain relative to a net with an even value of k is shown in Fig. 3.3, thus it is a particular birth-death process. Each node (apart from the first one, named A) is characterized by an integer number that denotes the distance between the two nodes. Let us now discuss the weight of the arcs in the Markov chain.

— The weight of the arcs going from nodes i to $i + 1$, and viceversa, for $i = 2, \dots, D - 1$ is equal to $2/N$ where $N = ks(s + 1)/2$ is the number of arcs². This follows from the fact that if a net has N arcs the probability of selecting a generic edge is equal to $1/N$; moreover, if the distance between the two agents is $i = 1, \dots, D - 1$, two are the edges whose selection leads to an increasing or decreasing of their distance. The same reasoning explains the weight of the arc going from $D - 1$ to D and the weight of the arc going from 2 to 1.

— If the distance between the two agents is unitary (the state of the Markov chain is 1) being by assumption the two agents on different rings, it means that they are on the same section. Two different cases may occur: either we select an edge that leads to a distance equal to 2, or the edge incident on the nodes containing the agents is selected. The first case occurs with a probability equal

¹The case of rings with an odd number of nodes k is upper bounded by the case of rings with $k + 1$ nodes.

²The number of arcs of a ring topology net is equal to k times the number of arcs of each section Σ , plus k times the number of arcs of each ring. Being each Σ a fully connected graph with s nodes, its number of arcs is equal to $s(s - 1)/2$. Therefore, $N = ks(s + 1)/2 + ks = ks(s + 1)/2$.

to $4/N$; the second case occurs with a probability equal to $1/N$ and leads to the absorbing state A .

— Now, assume that the distance between the agents is equal to D . Since by assumption the two agents walk on different rings, in such a case the selection of 4 different arcs may lead to a decreasing of their distance. Therefore the arc of the Markov chain going from node D to node $D - 1$ has a weight equal to $4/N$.

— Finally, the weights of all self-loops are due to the fact that the sum of the weights of arcs exiting a node is equal to 1 in a discrete Markov chain.

Given the Markov chain in Fig. 3.3 it is easy to compute the average hitting time of the absorbing state from any admissible distance. This can be done solving analytically the following linear system of equations:

$$(I - P') \tau = \mathbf{1} \quad (3.14)$$

where I is the D -dimensional identity matrix; P' has been obtained by the probability matrix P of the Markov chain in Fig. 3.3 removing the row and the column relative to the absorbing state¹; τ is the D -dimensional vector of unknowns: its i -th component $\tau(i)$ is equal to the hitting time of the absorbing state starting from an initial distance equal to i , for $i = 1, \dots, D$; finally, $\mathbf{1}$ is the D -dimensional column vector of ones. We found out that the worst case in terms of hitting time occurs when the two agents are at their maximum distance, i.e., for $i = D$. In particular it is $\tau(D) = \frac{n^2(s+1)}{32 \cdot s} \cdot \left(\frac{n}{s} + 16\right) = \frac{k^2 s(s+1)}{32} \cdot (k+16)$ where the last equality follows from the fact that $n = ks$. This proves the statement being $T_{\max} \leq \tau(D)$. \square

Proposition 3.4.14 *If a net has a generalized ring topology, then the average convergence time of Algorithm 2 in terms of the number of nodes n is*

$$\mathcal{E}[T_{\text{conv}}(Y)] \leq \varrho \cdot (M - m) \cdot \frac{n^2(s+1)}{32 \cdot s} \cdot \left(\frac{n}{s} + 16\right) \cdot (n - 1) = \mathcal{O}(n^4)$$

or, in terms of the net parameters k and s

¹It obviously holds that the hitting time of the absorbing state is null from the absorbing state itself.

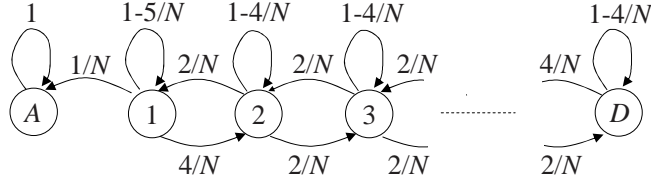


Figure 3.3: The Markov chain associated to a generalized ring topology net with an even value of k .

$$\mathcal{E}[T_{conv}(Y)] \leq \varrho \cdot (M - m) \cdot \frac{k^2 s(s + 1)}{32} \cdot (k + 16) \cdot (k s - 1) = \mathcal{O}(k^4 s^3).$$

Proof: Follows from equation (3.9) and Propositions 4.4.7 and 3.4.13. □

Remark 3.4.15 We want to motivate why generalized ring topology nets are significant within this framework, even if this class may appear restrictive.

Proposition 3.4.13, is based on the observation that the problem of computing an upper bound on the maximum average time between two consecutive balancing of Algorithm 2 can be formulated as the problem of finding the average meeting time of two agents walking on a graph executing a random walk. Such an average meeting time can obviously be computed using Markov chains, regardless of the structure of the net. However, while solving a linear system of the form (3.14) is easy for a given network, computing an analytical solution for a generic class of networks is an open problem.

In the case of generalized ring topology nets, the structure of the coefficient matrix of the linear system (3.14) is such that, through triangularization, we have been able to compute the meeting time analytically.

3.5 Conclusions

In this chapter we introduced a framework denoted as *discrete consensus*, that is a generalization of *quantized consensus*. We assumed that a set of tasks of different weight should be assigned to nodes with different speeds with the aim

of minimizing the maximum execution time. A solution based on gossip has been proposed and convergence properties have been examined in detail.

Chapter 4

Quantized Consensus on Hamiltonian Graphs

4.1 Introduction

A fair effort has been devoted to the problem of quantized consensus, i.e., the consensus problem over a network of agents with quantized state variables (25, 26, 29, 34), as a practical implementation of the continuous one (7, 23, 24, 50, 68). Such problem has relevant applications such as sensor networks, task assignment and token distribution over networks (a simplified load balancing problem) (27, 28, 69, 70). In the case of sensor networks, the quantized distributed average problem arises from the fact that sensor measurements are inevitably quantized given the finite amount of bits used to represent variables and the finite amount of bandwidth of the communication links between the nodes. Some approaches (71) deal with quantization by adding a quantization noise in the communication links to model such effect and study the resulting convergence properties without modifying the algorithms. Other approaches propose probabilistic quantization (25, 72) to ensure that after a certain amount of time each node has exactly the same value, even though it might be slightly different from the actual initial average of the measurements.

Some years ago in (29) it was originally proposed an algorithm to solve the distributed average problem with uniformly quantized measurements. Such an algorithm guarantees that almost surely the state of all the agents ($x_i, i = 1, \dots, n$)

will reach a value that is either equal to the floor of the average of the net (L), or the ceil ($L+1$), i.e., it ensures that the net will almost surely reach the convergence set

$$\mathcal{S} \triangleq \{\mathbf{x} : \{x_i\}_1^N \in \{L, L+1\}, L = \lfloor N^{-1} \sum_{i=1}^N x_i \rfloor\}.$$

However, a stopping criterion is missing, i.e., load transfers may occur even if the convergence set \mathcal{S} is reached.

Several works followed the pioneering work in (29). In (73) three quantized consensus algorithms are proposed which achieve a comparable performance respect to the one in (29). In (74) quantized consensus over random and switching graphs is addressed and polynomial upper-bounds to the convergence time are provided. In (75, 76) quantized gossip algorithms are investigated in the case of edges with different weights corresponding to different probabilities of being chosen.

In this chapter we propose an algorithm to solve the quantized distributed average problem using a gossip algorithm (56). Our algorithm can be applied to the token distribution problem, i.e., the problem of evenly distribute a set of tokens among the agents (29). We investigated the extension of this problem to the distribution of tokens of arbitrary size (34). Our algorithm presents two main advantages with respect to other applications and approaches in the literature:

1. A decentralized stopping criterion.
2. An average convergence time reduced with respect to (29, 34).

Moreover, let us observe that in our approach tokens may have different size. However, in the particular case of tokens with the same size our convergence set coincides with the convergence set in (29), defined as *quantized consensus*.

Our work has three main differences with respect to (73, 74, 75, 76). First, we consider tokens with arbitrary and possibly different size or cost as in (34, 77). Second, we consider hamiltonian graphs, i.e., graphs in which an hamiltonian cycle exists. Third, we propose a novel interaction rule to be applied when no averaging due to quantization issues can be applied, that improves the convergence time of the algorithm by reducing the average meeting time of two random walks in graph. Since the convergence time of all the quantized gossip and consensus algorithms proposed in (73, 74, 75, 76) depend upon the average meeting time of

two random walks in a graph, an interesting direction of research is to improve the convergence times of such algorithms with the ideas proposed in this chapter.

We remark that the issue of providing a stop criterion has already been solved by other authors using non uniform quantization, e.g., probabilistic or logarithmic quantization (25, 26). However, uniform quantization is surely easier to implement and less cost consuming than the other types of quantization. Moreover, in (25, 26) a convergence set is not defined, and the convergence properties are given in terms of probability.

Finally, our algorithm is based on gossip, i.e., only adjacent nodes asynchronously exchange information to achieve a global objective. In particular, one edge is selected at each iteration, and only the nodes incident on this edge may communicate and redistribute their tokens. Thus, no time synchronization is required nor information exchange between distant agents may occur. This clearly reduces significantly the implementation complexity and cost of the procedure. Note that parallel communications between disjoint sets of nodes are allowed as in (28). Nevertheless the convergence time is expressed as total number of updates to allow a straightforward comparison to other gossip algorithms.

The content of this chapter can be found in (34, 78). We provide both a convergence proof for the case in which edges are selected at random and a proof for the case in which there exists a periodic interval of time in which each link is selected at least once.

Algorithm Applications

The proposed Hamiltonian Quantized Consensus (HQC) algorithm may be applied in several application domains. The most significant ones are discussed in the following items.

- *Token distribution over networks.* The token distribution problem is a static variant of the load balancing problem (30, 69, 70, 79, 80, 81, 82) where K indivisible tokens of possibly different size should be uniformly distributed over N parallel processors.
- *Sensor networks.* The case in which tokens are indivisible and of unitary size is equivalent to the case in which a network of agents need to agree on the average of integer state variables.

- *Token Ring/IEEE 802.5 networks.* Our proposed algorithm well applies to all those application domains where the communication architecture is based on a Token Ring network which has an embedded Hamiltonian cycle.

4.2 Problem description

In this section we recall the results we presented in (34).

Let us consider a network of n agents whose connections can be described by an undirected connected graph $\mathcal{G} = (V, E)$, where V is the set of nodes (agents) and E is the set of edges.

Assume that K indivisible tokens should be assigned to the nodes, where the size of the generic j -th token is denoted as c_j , $j = 1, \dots, K$. Notice that assuming unitary size for all tokens is equivalent to the problem of quantized consensus with integer state variables (29).

Our goal is that of achieving a globally balanced state, starting from any initial condition, such that the total number of tokens weighted by their sizes in each node is as close as possible, in the least-square sense, to the best possible token distribution

$$\bar{c} = \frac{1}{n} \sum_{j=1}^K c_j. \quad (4.1)$$

In the token distribution problem no token enters nor leaves the network thus the total amount of tokens is preserved during the iterations. This assumption is helpful in abstracting the convergence properties of the network that depend on the topology and on the actual token distribution. In the following we will refer to the total size of the tokens in the generic node as the load of such a node.

We define a cost vector $c \in \mathbb{N}^K$ whose j -th component is equal to c_j , and n binary vectors $y_i \in \{0, 1\}^K$ such that

$$y_{i,j} = \begin{cases} 1 & \text{if the } j\text{-th token is assigned to node } i \\ 0 & \text{otherwise.} \end{cases} \quad (4.2)$$

In the following, given a generic node i , we denote $\mathcal{K}_i(t)$ the set of indices of tokens assigned to i at time t , where $\sum_{j \in \mathcal{K}_i} c_j = c^T y_i$.

The optimal token distribution corresponds to any distribution such that the following performance index

$$V_1(Y) = \sum_{i=1}^n (c^T y_i - \bar{c})^2, \quad (4.3)$$

is minimum, where

$$Y(t) = [y_1(t) \ y_2(t) \ \dots \ y_n(t)] \quad (4.4)$$

denotes the state of the network at time t and Y^* (resp., V_1^*) is the optimal token distribution (resp. optimal value of the performance index). Finally, we denote

$$c_{\max} = \max_{j=1, \dots, K} c_j \quad c_{\min} = \min_{j=1, \dots, K} c_j \quad (4.5)$$

respectively the maximum and the minimum size of tokens in the network.

An interesting class of decentralized algorithms for load balancing or averaging networks is given by *gossip-based* algorithms that can be summarized as follows (29, 34).

Algorithm 3 (Quantized Gossip Algorithm)

1. Let $t = 0$.
2. Select an edge $e_{i,r}$.
3. Perform a local balancing between nodes i and r using a suitable rule such that the difference between their loads is reduced.
If such a balancing is not possible execute a swap among the loads in i and r .
4. Let $t := t + 1$ and goto Step 2. ■

A *swap* is an operation between two communicating nodes that, while not reducing nor increasing their load difference, it modifies the token distribution.

Definition 4.2.1 (34) [Swap] *Let us consider two nodes i and r incident on the same edge and let $\mathcal{J}_i \subseteq \mathcal{K}_i(t)$ and $\mathcal{J}_r \subseteq \mathcal{K}_r(t)$ be two subsets of their tokens.*

We call swap the operation that moves the tokens in \mathcal{J}_i to r , and the tokens in \mathcal{J}_r to i at time $t + 1$, reaching the distribution

$$\begin{aligned}\mathcal{K}_i(t + 1) &= \mathcal{J}_r \cup (\mathcal{K}_i(t) \setminus \mathcal{J}_i), \\ \mathcal{K}_r(t + 1) &= \mathcal{J}_i \cup (\mathcal{K}_r(t) \setminus \mathcal{J}_r)\end{aligned}$$

provided the absolute value of the load difference between the two nodes does not change.

In particular, we say that a total swap occurs if $\mathcal{J}_i = \mathcal{K}_i(t)$ and $\mathcal{J}_r = \mathcal{K}_r(t)$. ■

In the following section we provide an algorithm that is still based on the notion of swap. However, the main difference with respect to Algorithm 3 is that in Algorithm 3 swaps are executed following a random process, while in the proposed algorithm we exploit the existence of an hamiltonian cycle in the graph so that they can be executed following an appropriate criterion. As discussed in detail in the rest of the chapter, this leads to two main advantages. First, if the average out-degree of the nodes is not high, it results in a smaller convergence time. Secondly, our algorithm has a stopping criterion, while Algorithm 3 indefinitely iterates even if no further improvement can be obtained.

4.3 Proposed algorithm

Our idea is based on the notion of Hamiltonian cycle, and our assumption is that the considered nets are represented by Hamiltonian graphs, i.e., they have a Hamiltonian cycle.

Definition 4.3.1 *A Hamiltonian cycle is a cycle in an undirected graph that visits each vertex exactly once and returns to the starting vertex. ■*

Given a network represented by graph $\mathcal{G} = \{V, E\}$ we label the nodes $V = 1, \dots, n$ along the Hamiltonian cycle in increasing order such that node i is connected to node $i + 1$ and node n is connected to node 1. According to this, we define the set of edges belonging to the Hamiltonian cycle as $\mathcal{H} = \{e_{i,i+1} =$

$\{V_i, V_{i+1}\}, \quad i = 1, \dots, n - 1\} \cup \{e_{n,1}\}$. It follows that if \mathcal{G} is Hamiltonian then $\mathcal{H} \subseteq E$.

In such a Hamiltonian cycle we label edge $e_{n,1}$ as e_{ae} and call it *absorbing edge*.

In the literature the question of how common Hamiltonian cycles are in arbitrary graphs is still an open issue even if many results exist in this framework. In particular it is known that if the number of nodes and arcs is sufficiently high then almost surely a Hamiltonian cycle exists (83?).

Finding a Hamiltonian cycle in a graph is an NP-complete problem (84). On the other hand, many algorithms can be formulated to design a network such that a Hamiltonian cycle is embedded in it by construction (85) or to find it in a distributed way (86?). Furthermore there exist communication architectures where a Hamiltonian cycle is embedded in their structure. A famous example of such a communication architecture is the Token Ring network (87).

Note that the proposed algorithm is “distributed”. Indeed the agents need not to know the network topology nor the number of agents. The agents only know who are the next and previous agents on the directed Hamiltonian cycle and whether one of their incident edges is the absorbing edge. The assignment of increasing integer numbers as labels to the nodes is an arbitrary choice we have done for simplicity of presentation.

Notice that the network can be *arbitrarily* connected as long as it contains a Hamiltonian cycle.

In the following we denote the total amount of load in the generic node i at time t as $x_i(t) = c^T y_i(t)$. The optimal assignment of tokens \bar{y}_i, \bar{y}_r at time t between two different nodes with respect to (4.3) is the one that minimizes the following quantity:

$$(\bar{y}_i, \bar{y}_r) = \arg \min_{y_i, y_r} |x_i(t) - x_r(t)|$$

given the set of tasks $\mathcal{K}_i(t) \cup \mathcal{K}_r(t)$.

The following algorithm assumes that a Hamiltonian cycle is determined before its initialization.

Algorithm 4 (Hamiltonian Quantized Consensus (HQC) Algorithm)

1. Let $t = 0$.

2. An edge $e_{i,r}$ is selected at random.
3. If $x_i(t) \neq x_r(t)$ (the load balancing among the two nodes may potentially be improved)
 - (a) Let \bar{x}_i, \bar{x}_r and respectively \bar{y}_i, \bar{y}_r , be the optimal assignment of tokens with indices in $\mathcal{K}_i(t) \cup \mathcal{K}_r(t)$
 - (b) If $|\bar{x}_i - \bar{x}_r| < |x_i(t) - x_r(t)|$, then

$$\begin{aligned} y_i(t+1) &= \bar{y}_i, \\ y_r(t+1) &= \bar{y}_r; \end{aligned}$$

and goto step 6.

4. If $e_{i,r} \notin \mathcal{H}$ or $e_{i,r} \equiv e_{ae}$ then

$$\begin{aligned} y_i(t+1) &= y_i(t), \\ y_r(t+1) &= y_r(t); \end{aligned}$$

else if $e_{i,r} \in \mathcal{H} \setminus \{e_{ae}\}$,

if $x_r(t) \equiv x_{i+1}(t) > x_i(t)$ then execute a swap such that

$$x_i(t+1) > x_{i+1}(t+1),$$

else

$$\begin{aligned} y_i(t+1) &= y_i(t), \\ y_r(t+1) &= y_r(t); \end{aligned}$$

and goto step 6.

5. If $x_i(t) = x_r(t)$, then

$$\begin{aligned} y_i(t+1) &= y_i(t), \\ y_r(t+1) &= y_r(t); \end{aligned}$$

6. Let $t = t + 1$ and go back to Step 2.

■

Explanation of the algorithm

In simple words, at each time t an edge is arbitrary selected. If the two nodes incident on the edge have different loads we look for a better load balancing (that may potentially occur only if their loads differ of more than one unit). If the edge belongs to the Hamiltonian cycle but it is not the absorbing edge, then the larger loads are moved toward nodes with smaller index and the smaller loads to nodes with higher index. Thus, the largest and smallest loads eventually meet at the absorbing edge where they can eventually be balanced.

Remark 4.3.2 *We point out that in general if the tokens are not of unitary size it is not guaranteed that the final load configuration is optimal. The following Theorem 4.4.1 characterizes the convergence properties of the algorithm and shows that disregarding the network topology, the number of tokens and the number of nodes, the maximum distance of the final tokens distribution from the optimal one depends only on the token sizes. ■*

As it will be formally proved in the following section, while preserving the asynchrony of the local updates, the simple notion of a "preferred" direction produces several important advantages. Firstly, it reduces the convergence time; then, it makes finite the total number of tokens exchanges between the nodes to achieve the global tokens distribution; finally, it makes the algorithm stop once a balanced state is reached¹ to allow a change of mode of operation (e.g., take a new measurement in the case of a sensor network or proceed with task execution in the case of multi agent systems).

Let us conclude this section with an important remark.

Remark 4.3.3 *Algorithm 4 does not contain an explicit stopping criterion. What happens in practice is that, after a certain number of iterations, no load can be further balanced nor swapped. However, the communication among nodes continues indefinitely.*

¹We point out that some algorithms in the literature (29) achieve quantized consensus asymptotically, without actually terminating. This is a relevant issue in the case of load balancing and tasks assignment. In wireless sensor networks such an improvement also allows to save power by avoiding averaging indefinitely after having reached a satisfactory agreement.

To impose a stopping criterion on communications, we may assume that the edge selection is implemented in a distributed fashion as follows: any node may asynchronously start a communication request with one of its neighbors. After a node has already tested all its possible communications and no balancing or swap was possible, it will enter a “sleeping” state in which it will wait for communication requests but it will not start any new communication. If a sleeping node receives a communication request and as a result its load changes, then it leaves the sleeping state. This ensures that once the network reaches a configuration from which no evolution is possible, each node, after having tested all its link, will reach a sleeping state and all communications will eventually stop. ■

A numerical example

Let us consider the network in Fig. 4.1(a). It consists of six nodes whose connections allow the existence of a Hamiltonian cycle. By assumption arcs are undirected. The direction given to the edges in the Hamiltonian cycle is only introduced to better explain the steps of the algorithm. Assume that the initial token distribution is that in Fig. 4.1(a): here the integer numbers upon nodes denote the size of tokens in their inside. Finally, $e_{ae} = e_{6,1}$ is the absorbing edge.

We now run Algorithm 4. In Table ?? the evolution of the network is shown. As it can be seen, when Algorithm 4 can not locally balance the loads, it moves the largest load toward nodes with smaller index and the smallest one to nodes with higher index. This behavior makes the largest load move toward node V_1 and the smallest one to V_6 . In Fig. 4.1(b) is shown the token distribution at time $t = 1$. Here the thick dashed edge denotes the selected edge. In Fig. 4.1(c) is shown the final token distribution reached at time $t = 10$.

Let us finally observe that all the updates are decentralized and asynchronous, i.e., the order in which edges are selected is not relevant to the algorithm convergence properties. After $t = 10$ local updates of the network is in a globally balanced configuration: due to the token quantization a better distribution is not reachable.

Moreover, starting from the last configuration no further load transfer is allowed because every node is locally balanced with its neighbors and the loads are in descending order starting from node V_1 to node V_6 . This is a great advantage

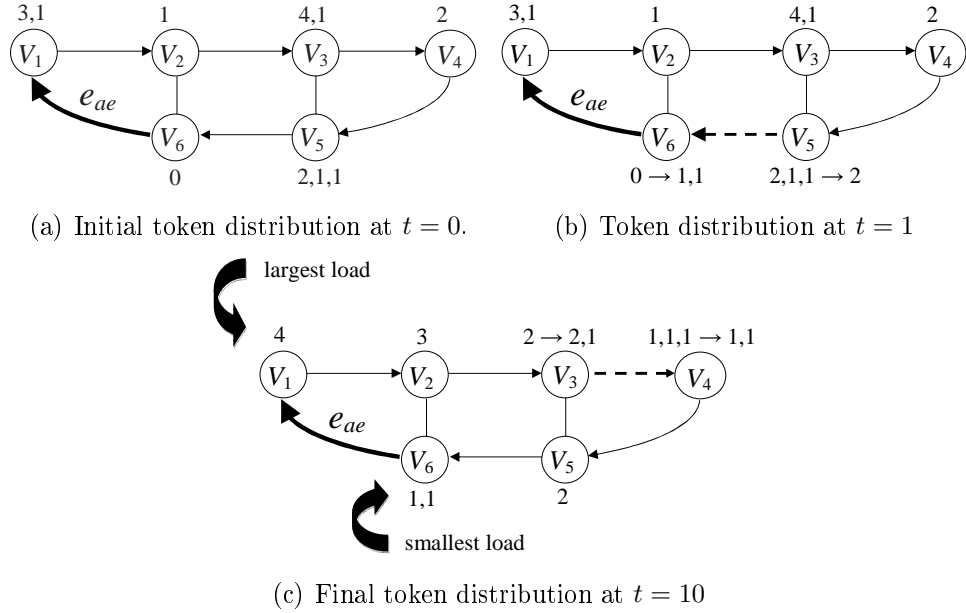


Figure 4.1: The network considered in Subsection 5.3.5.

with respect to other randomized algorithms which keep on swapping loads even after the best load configuration achievable is reached (29, 34).

4.4 Convergence properties

The convergence properties of Algorithm 4 are stated by the following theorem. In particular, Theorem 4.4.1 claims that using Algorithm 4 the net distribution will almost surely converge to a given set \mathcal{Y} defined as in the following equation (4.6).

Theorem 4.4.1 *Let us consider*

$$\mathcal{Y} = \{Y = [y_1 \ y_2 \ \cdots \ y_n] \mid |c^T y_i - c^T y_r| \leq c_{\max}, \quad \forall i, r \in \{1, \dots, n\}\}. \quad (4.6)$$

Let $Y(t)$ be the matrix that summarizes the token distribution resulting from Algorithm 4 at the generic time t .

It holds

$$\lim_{t \rightarrow \infty} \Pi(Y(t) \in \mathcal{Y}) = 1$$

Time	Edge\Node	V_1	V_2	V_3	V_4	V_5	V_6
0		3, 1	1	4, 1	2	2, 1, 1	0
1	$e_{5,6}$	3, 1	1	4, 1	2	2	1, 1
2	$e_{3,5}$	3, 1	1	4	2	2, 1	1, 1
3	$e_{2,3}$	3, 1	4	1	2	2, 1	1, 1
4	$e_{1,6}$	3	4	1	2	2, 1	1, 1, 1
5	$e_{4,5}$	3	4	1	2, 1	2	1, 1, 1
6	$e_{1,2}$	4	3	1	2, 1	2	1, 1, 1
7	$e_{3,4}$	4	3	2	1, 1	2	1, 1, 1
8	$e_{5,6}$	4	3	2	1, 1	2, 1	1, 1
9	$e_{4,5}$	4	3	2	1, 1, 1	2	1, 1
10	$e_{3,4}$	4	3	2, 1	1, 1	2	1, 1

Table 4.1: The results of the numerical example in Subsection 5.3.5.

where $\Pi(Y(t) \in \mathcal{Y})$ denotes the probability that $Y(t) \in \mathcal{Y}$.

Proof. We define a Lyapunov-like function

$$V(t) = [V_1(t), V_2(t)] \quad (4.7)$$

consisting of two terms. The first one is:

$$V_1(Y(t)) = \sum_{i=1}^n (x_i(t) - \bar{c})^2 \quad (4.8)$$

where $x_i(t) = c^T y_i(t)$ for $i = 1, \dots, n$. The second one is a measure of the ordering of the loads:

$$V_2(t) = \sum_{i=1}^{n-1} \sum_{j=i+1}^n f(x_i(t) - x_j(t)) \quad (4.9)$$

where

$$f(x_i(t) - x_j(t)) = \max(\text{sign}(x_i(t) - x_j(t)), 0)$$

Note that here we are assuming that $e_{ae} = e_{n,1}$ and nodes are labeled as in Fig. 4.2. Therefore, $V_2(t)$ denotes the number of couples of nodes that are not ordered¹ at time t .

¹According to Algorithm 4 and the notation in Fig. 4.2 a couple of nodes $\{i, j\}$ is said to be ordered if for $i < j$, it is $x_i < x_j$.

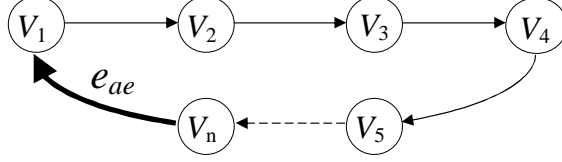


Figure 4.2: The oriented Hamiltonian cycle considered in the proof of Theorem 4.4.1 and Proposition 4.4.9.

We impose a lexicographic ordering on the performance index, i.e., $V = \bar{V}$ if $V_1 = \bar{V}_1$ and $V_2 = \bar{V}_2$; $V < \bar{V}$ if $V_1 < \bar{V}_1$ or $V_1 = \bar{V}_1$ and $V_2 < \bar{V}_2$.

The proof is based on three arguments.

1. $V_1(t)$ is a non increasing function of t . In fact, at any time t it holds $V_1(t+1) \leq V_1(t)$.

The case $V_1(t+1) = V_1(t)$ holds during a token exchange when the resulting load difference between the nodes is not reduced. In such a case the loads at the nodes may either swap or not, thus not increasing nor decreasing the value of the Lyapunov function.

The case of $V_1(t+1) < V_1(t)$ holds when a new load balancing occurs. Assume that a combination of tokens with total cost q with $0 < q < |x_i(t) - x_r(t)|$ is moved from i to r at the generic time t such that

$$|x_i(t+1) - x_r(t+1)| < |x_i(t) - x_r(t)|.$$

It is easy to verify, by simple computations, that

$$(x_i(t+1) - \bar{c})^2 + (x_r(t+1) - \bar{c})^2 < (x_i(t) - \bar{c})^2 + (x_r(t) - \bar{c})^2$$

which implies $V_1(t+1) < V_1(t)$.

We also observe that if two nodes (e.g., i and r) communicate at time t , the resulting difference among their loads at time $t+1$ is surely less or equal to the largest cost of tokens in the nodes at time t , i.e.,

$$|x_i(t+1) - x_r(t+1)| \leq \max_{j \in \mathcal{K}_i(t) \cup \mathcal{K}_r(t)} c_j \leq c_{\max}. \quad (4.10)$$

This is due to the fact that if the load difference between two nodes is greater than c_{\max} , it is always possible to move at least one token with $c \leq c_{\max}$ to the less loaded node to reduce the load difference.

2. $V_2(t)$ is a positive non increasing function of t if $V_1(t + 1) = V_1(t)$.

Function $V_2(t)$ is positive because it is the summation of positive quantities. Moreover, $V_2(t + 1) = V_2(t)$ anytime an edge connecting two nodes already ordered along the Hamiltonian cycle is chosen, or alternatively when the absorbing edge is chosen. This is due to the fact that in such a case the ordering of loads does not change.

While $V_2(t + 1) < V_2(t)$ anytime the loads of two nodes are reordered along the Hamiltonian cycle and the load difference between the loads is not reduced. This follows from the fact that if the loads of nodes i and j are not ordered at time t , i.e., for $i < j$, $x_i(t) < x_j(t)$, we have that

$$f(x_i(t) - x_j(t)) = 1.$$

If the edge connecting them is selected and they are ordered, then at time $t + 1$ it is

$$f(x_i(t + 1) - x_j(t + 1)) = 0.$$

Furthermore since the nodes are directly connected, their ordering does not affect the value of f for other couples of nodes. If a ordering happens, then $V_2(t + 1) = V_2(t) - 1$.

Finally, if at time t all the loads are ordered along the Hamiltonian cycle it is easy to verify that $V_2(t) = 0$.

3. If the Lyapunov-like function $V(t)$ has not reached its minimum at a given time t , then there exists an edge along the Hamiltonian cycle with strictly positive probability to be chosen such that $V(t + 1) < V(t)$.
- (a) If an edge is selected and the load difference between two nodes is reduced then $V_1(t + 1) < V_1(t)$.
 - (b) If there does not exist an edge such that the load difference between the two nodes is reduced, we can always select an edge such that the loads are reordered if $V_2(t) \neq 0$, then $V_2(t + 1) < V_2(t)$.
 - (c) If $V_2(t) = 0$ then the nodes connected by the absorbing edge contain the maximum and minimum load in the network. If their difference is greater than c_{max} then we can select the absorbing edge and have $V_1(t + 1) < V_1(t)$.

- (d) If $V_2(t) = 0$ and the load difference between the nodes connected by the absorbing edge is less or equal than c_{max} then $Y(t) \in \mathcal{Y}$.

Finally, at each instant of time, we proved that there exists an edge with strictly positive probability p that if selected makes $V(t+1) < V(t)$. The probability that such an edge is selected at least once in t time steps is $P(t) = 1 - (1-p)^t$. Thus since we assume p to be strictly positive, the probability that such an edge is selected goes to 1 as t goes to infinity, thus proving the statement. \square

Remark 4.4.2 *We remark that such a theorem states the convergence toward a balanced situation in which the load difference between any couple of nodes in the network is at most c_{max} . However, in principle any load balancing rule can be designed to have a greater threshold to trigger the local balancing mechanism, for instance one in which the load difference between the two nodes is $\gamma > c_{max}$. In such a case the theorem gives a design criterion for such threshold since it states that the local threshold used for the balancing mechanism will hold globally by bounding the maximum load difference between any two nodes. \blacksquare*

A characterization of the maximum distance of the final set of token distribution using Algorithm 4 from the optimal one is given by the following proposition.

Proposition 4.4.3 *Let us consider the optimal token distribution problem, and let the set \mathcal{Y} be defined as in equation (4.6). Let $V_1(Y) = \sum_{i=1}^n (c^T y_i - \bar{c})$, where $Y \equiv Y(t)$ results from the application of Algorithm 4 for a sufficiently long time t .*

The following inequalities hold for any $Y \in \mathcal{Y}$:

$$0 \leq V_1^* \leq V_1(Y) \leq \alpha \quad (4.11)$$

where

$$\alpha = \begin{cases} \frac{nc_{max}^2}{4} & \text{if } n \text{ is even,} \\ \lfloor \frac{n}{2} \rfloor \lceil \frac{n}{2} \rceil \frac{c_{max}^2}{n} & \text{if } n \text{ is odd.} \end{cases} \quad (4.12)$$

Proof. The first two inequalities are trivial. To prove the last inequality we look at the worst case, i.e., the token distribution in \mathcal{Y} that has the highest value of $V_1(Y)$.

If n is even, the worst case corresponds to a balancing where half of the nodes have a load k and the remaining half have a load $k + c_{\max}$. In this case $\bar{c} = k + 0.5c_{\max}$, and the first value of bound can be computed.

If n is odd, the worst case corresponds to a configuration where $\lfloor n/2 \rfloor$ of the nodes have a load k and the remaining $\lceil n/2 \rceil$ have a load $k + c_{\max}$. Now $c_{ave} = k + \lceil n/2 \rceil c_{\max}/n$, which gives the other value of the bound. \square

The above results enable us to characterize some cases in which Algorithm 4 provides the optimal solution to the token distribution problem.

Proposition 4.4.4 *Let c_{\min} and c_{\max} be defined as in (4.5).*

If $c_{\min} = c_{\max} = c$, then all load distributions that belong to a set of final distributions (4.6) are optimal, hence Algorithm 4 provides a token distribution for which $V_1(Y)$ is minimum and thus it is an optimal distribution.

Proof. If $c_{\min} = c_{\max}$ the set of final distributions is

$$\mathcal{Y} = \{[y_1 \cdots y_n] \mid (\forall i) c^T y_i \in \{\lfloor \frac{K \cdot c}{n} \rfloor, \lfloor \frac{K \cdot c}{n} \rfloor + c\}\}. \quad (4.13)$$

We can normalize the weight c so that it is unitary. With this formulation the problem corresponds to that of quantized consensus, and the set \mathcal{Y} coincides with the set of the *quantized-consensus distributions* defined in (29) and shown to be optimal. \square

We now prove that Algorithm 4 always reaches a blocking configuration.

Proposition 4.4.5 *Given a Hamiltonian Graph \mathcal{G} , if the network evolves according to Algorithm 4, then*

$$\forall Y(0), \exists t' : \forall t \geq t', Y(t) \equiv Y(t') \in \mathcal{Y}.$$

Proof. Due to Theorem 4.4.1 $\exists t'$ such that $\forall t \geq t', Y(t) \in \mathcal{Y}$. Let us consider the Lyapunov-like function (4.7):

$$V(t) = [V_1(t), V_2(t)].$$

It can be shown that if $V_2(t) = 0$ then the loads are ordered such that $x_i \geq x_{i+1}$ for $i = 1, \dots, n-1$. If at time t' the loads are ordered and $V_1(t')$ has reached a local minimum, then according to Algorithm 4 no token exchange is performed since no balancing is feasible and no swap is allowed. Then it follows that $Y(t' + \Delta t) \equiv Y(t') \quad \forall \Delta t \geq 0$. \square

Convergence time

In this section we discuss the expected convergence time of Algorithm 4, and provide an upper bound for arbitrary Hamiltonian graphs.

We assume that edges are selected with uniform probability, so the probability to select the generic edge $e_{i,j}$ at time t is equal to $p = 1/N$ where N is the number of edges in the network.

The *convergence time* is a random variable defined for a given initial load configuration $Y(0) = Y$ as:

$$T_{con}(Y) = \inf \{t \mid \forall t' \geq t, Y(t') \in \mathcal{Y}\}.$$

Thus, $T_{con}(Y)$ represents the number of steps required at a certain execution of Algorithm 4 to reach the convergence set \mathcal{Y} starting from a given token distribution.

Now, let us provide some further definitions that will occur in the following.

- N_{\max} is the maximum number of improvements of $V_1(Y)$ needed by any realization of Algorithm 4 to reach the set \mathcal{Y} , starting from a given configuration.
- T_{\max} is the maximum average time between two consecutive improvements of $V_1(Y)$ in any realization of Algorithm 4, starting from a given configuration.

From the previous definitions, it is possible to give an upper bound on the expected convergence time.

Proposition 4.4.6 *Let $\mathcal{E}[T_{con}(Y)]$ be the expected convergence time. It holds $\mathcal{E}[T_{con}(Y)] \leq N_{\max} \cdot T_{\max}$. ■*

Notice that the term maximum average time in the above definition is intended as in the following.

The average time between two consecutive improvements is a function of the load distribution: an unbalanced distribution has a short average time between two consecutive improvements, while a nearly balanced distribution has a long average time. In our definition we consider the longest possible average time

between two improvements and take it as an upper bound to the average time between two consecutive improvements.

In (29) an upper bound on N_{\max} is given when $c_{\max} = 1$. In our case the result still holds since it is based on the fact that the improvement of the performance index is lower bounded by $V_1(Y(t+1)) \leq V_1(Y(t)) - 2$ since the minimum token exchange allowed decreases the load difference between two nodes of at least 1. Finally, the initial value of $V_1(Y(0))$ can be upper bounded by a function of the maximum and the minimum amount of load in the generic node.

Proposition 4.4.7 (29) *For the Hamiltonian Quantized Gossip it holds:*

$$N_{\max} = \frac{(M - m)n}{4}$$

where $M = \max_i c^T y_i$ and $m = \min_i c^T y_i$.

We now focus on T_{\max} . As shown in the following proposition, it is easy to compute in the case of fully connected networks.

Proposition 4.4.8 *Let us consider a fully connected network, namely a net such that $E = \{V \times V\}$. Let n be the number of nodes.*

It holds

$$T_{\max} = \frac{n(n-1)}{2}. \quad (4.14)$$

Proof. The maximum average time between two consecutive balancing occurs when only one balancing is possible. Thus, if N is the number of edges of the net, then the probability of selecting the only edge whose incident nodes may balance their load is equal to $p = 1/N$, while the average time needed to select it is equal to N . Since the network is fully connected, if n is the number of nodes, the number of edges is $N = n(n-1)/2$ and so $T_{\max} = n(n-1)/2$. \square

Notice that the previous proposition holds for various gossip based algorithms (29, 34).

We now show that T_{\max} for Hamiltonian graphs is of the same order with respect to the number of nodes as for fully connected topologies when using the Hamiltonian Quantized Gossip Algorithm.

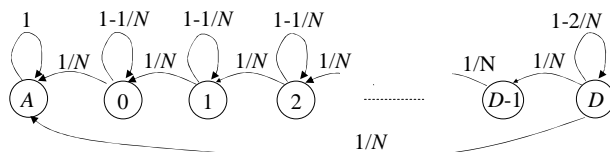


Figure 4.3: The Markov chain associated to a net containing a Hamiltonian cycle.

Proposition 4.4.9 *Let us consider a net with a Hamiltonian cycle. Let n be the number of nodes, and N be the number of arcs of the net.*

It holds

$$T_{\max} \leq N(n - 2). \quad (4.15)$$

Proof. We first observe that, due to the gossip nature of Algorithm 4 and to the rule used to select the edges, the problem of evaluating an upper bound on T_{\max} can be formulated as the problem of finding the average meeting time of two agents walking on the Hamiltonian cycle in opposite directions¹. In fact, the average meeting time of the two agents may be thought as the average time of selecting an edge whose incident nodes may balance their load. Note that in general more than two edges may balance their load, thus assuming that only two agents are walking on the graph provides us an upper bound on the value of T_{\max} .

To compute such an upper bound we determine the average meeting time of the largest and smallest load walking on the graph along the Hamiltonian cycle in the worst case. To this aim we define the discrete Markov chain in Fig. 4.3 whose states (apart from the first one, named A) characterize the distance between the two agents.

For simplicity of explanation we assume that the first agent is the one corresponding to the largest load.

The distance between the two agents is equal to the length of the path going from the first agent to the second one in the direction of nodes with increasing index. In other words, the distance between the two agents is equal to the minimum number of movements they need to perform, following the rule at Step 3 of Algorithm 4, to meet each other.

¹The problem of random walk and average meeting times has been extensively studied in different applications (88, 89).

Now, if a net has n nodes, then the Hamiltonian cycle has n edges, and the maximum distance among the two agents is equal to $D = n - 1$, while their minimum distance is equal to 1.

Note that both these conditions correspond to the case in which the two agents are in nodes incident on the same edge. However, the first case occurs when such an edge is directed from the second agent to the first one, while the second case happens when the edge is directed from the first agent to the second one. As an example, if the Hamiltonian cycle is that reported in Fig. 4.2, if the first agent is in V_1 and the second one is V_n , then their distance is null; if the first agent is in V_n and the second one in V_1 , then their distance is equal to D .

The absorbing state (node A in Fig. 4.3) corresponds to the case in which the agents are in nodes incident on the same edge and this edge is selected. Thus, the absorbing state may only be reached from nodes 1 and D , and the probability that this occurs is in both cases equal to $1/N$.

Moreover, given the rule of step 3 of Algorithm 4, the distance among two nodes with load difference greater than c_{max} may only decrease, regardless their initial position. In particular, the probability of going from node i to node $i - 1$, with $i = D, D - 1, \dots, 1$, is equal to $2/N$, because two are the edges whose selection leads to a unitary reduction of the distance among the agents.

Finally, we consider the linear system:

$$(I - P')\tau = \mathbf{1} \quad (4.16)$$

where I is the D -dimensional identity matrix; P' has been obtained by the probability matrix P of the Markov chain in Fig. 4.3 removing the row and the column relative to the absorbing state¹; τ is the D -dimensional vector of unknowns: its i -th component $\tau(i)$ is equal to the hitting time of the absorbing state starting from an initial distance equal to i , for $i = 1, \dots, D$; finally, $\mathbf{1}$ is the D -dimensional column vector of ones. Solving analytically the linear system (4.16), we found out that $\tau(i) = iN$ for $i = 1, \dots, D - 1$, and $\tau(D) = N(n - 1)/2$. Thus the maximum average hitting time of the absorbing state occurs when the distance between the two nodes is equal to $D - 1$ if $n \geq 3$. In particular, it holds $\tau(D - 1) = N(n - 2)$ that proves the statement. \square

¹It obviously holds that the hitting time of the absorbing state is null from the absorbing state itself.

Proposition 4.4.10 *An upper bound to the average convergence time of Algorithm 4 is*

$$\mathcal{E}[T_{con}(Y)] \leq \frac{(M - m)n}{4} \cdot N(n - 2) = \mathcal{O}(n^2N).$$

Proof. The statement follows from Propositions 4.4.7 and 4.4.9 and Fact 4.4.6. \square

Proposition 4.4.11 *If a net is fully connected, an upper bound to the average convergence time of Algorithm 4 is*

$$\mathcal{E}[T_{con}(Y)] \leq \frac{(M - m)n}{4} \cdot \frac{n(n - 1)}{2} = \mathcal{O}(n^3).$$

Proof. Follows from Propositions 4.4.7 and 4.4.8 and Fact 4.4.6. \square

The above propositions enable us to conclude that Algorithm 4 leads to a significant improvement respect to (29, 34) in terms of convergence time for networks with low average out-degree (e.g. path networks). Indeed for such networks the average convergence time is $\mathcal{O}(n^4)$ using the approaches in (29, 34), while it is $\mathcal{O}(n^3)$ using Algorithm 4. On the contrary, if we consider networks with a high average out-degree such as fully connected networks, in both cases the average convergence time is $\mathcal{O}(n^3)$ and the advantage of Algorithm 4 is basically that of providing a stopping criterion.

In Figure 4.4 is shown the average convergence time for a ring network of n nodes with $n = 10, \dots, 100$ and random initial loads ranging from 0 to 10. For each network size the average convergence time is taken over 100 realizations of the experiment. In such a figure is also shown a comparison with the previously computed upper bound to the average convergence time, it is evident that such a bound is not strict, i.e., the actual performance of the algorithm is considerably better than the worst case analysis prediction. Furthermore we point out that the convergence time is given in number of local updates, not time, thus disregarding the effects of parallel communications for analysis purposes.

Algorithm extension for convergence in finite time

The effectiveness of Algorithm 4 is even more evident if a periodic interval of time T_h exists such that each edge in the Hamiltonian Cycle is selected at least once. In

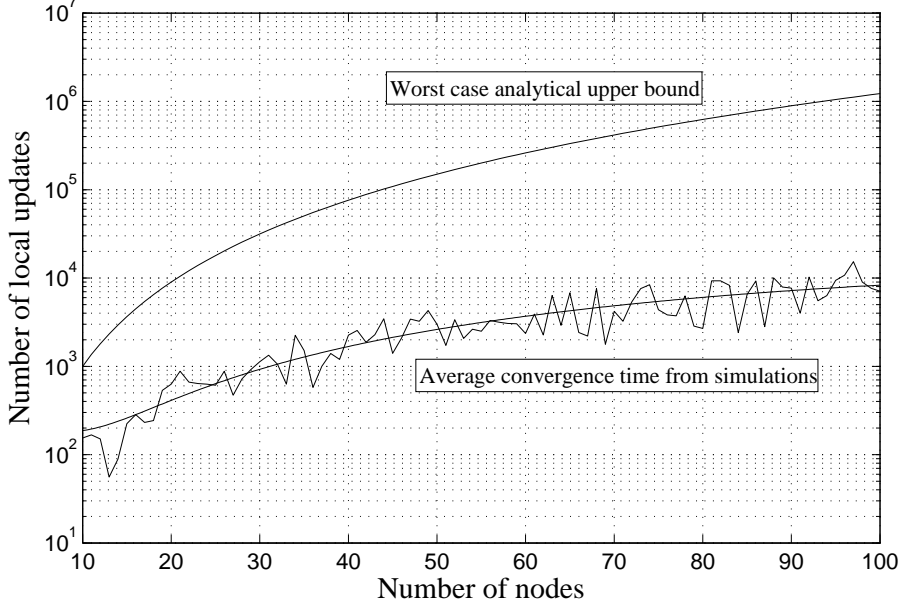


Figure 4.4: Comparison between simulation results and the worst case analytical average convergence time.

such a case Algorithm 4 converges in finite time, as will be shown in the following. Furthermore if Algorithm 4 is applied to networks whose edge selection process is deterministic, it still preserves its convergence properties while other algorithms as the one in (29) may cycle indefinitely without reaching the consensus set of final configurations. Obviously Algorithm 4 prevents the existence of such cycles due to the deterministic swap rule. In particular, the following result holds.

Proposition 4.4.12 *If there exists a period of time T_h such that each edge along the Hamiltonian cycle is selected at least once, then a deterministic upper-bound to the convergence time of Algorithm 4 is*

$$\max(T_{con}(Y)) \leq (n - 1)^2 \cdot (M - m) \cdot T_h = \mathcal{O}(n^2).$$

Proof. By Proposition 4.4.7 the maximum number of balancing between two consecutive improvements of $V(Y)$ is at most equal to $\frac{(M-m)n}{4}$. Now, if each edge of the Hamiltonian cycle is selected at least once during T_h , being the maximum distance between the two nodes with the smallest and highest load in the network

equal to $n - 1$ (see the proof of Proposition 4.4.9), then at each interval T_h their distance is surely reduced by at least 1 and they meet after at most $(n - 1)T_h$ units of time. Then, $\frac{(M-m)n}{4}(n - 1) \cdot T_h$ is the maximum number of time units required to reach the convergence set \mathcal{Y} . \square

The above proposition states a finite time bound on the convergence time of Algorithm 4.

We note that to make Proposition 4.4.12 useful in practical cases, namely if we want to use it as a criterion to know when \mathcal{Y} is reached for sure, then a slight overhead needs to be added to Algorithm 4 to evaluate the difference $M - m$ of the initial load. This can be done in a decentralized way with a consensus-like algorithm (namely consensus on $\max_i x_i(0)$).

4.5 Conclusions

In this chapter we proposed a new algorithm, the Hamiltonian Quantized Gossip Algorithm, that solves the quantized distributed average problem and the token distribution problem on Hamiltonian graphs with a grater efficiency respect to other gossip algorithms based on uniform quantization (29, 34). The main feature of the proposed algorithm is an embedded stopping criterion that will block the algorithm once quantized consensus has been achieved.

We have also shown that, if there exists a periodic interval of time where each edge along the Hamiltonian cycle is selected at least once, a finite time convergence bound can be given thus ensuring a finite and known amount of total transfers for load balancing applications (that can be used as a design criterion for the local load balancing trigger mechanism) and a decentralized criterion to stop averaging in sensor networks.

Chapter 5

Consensus problems in directed graphs

5.1 Introduction

Great effort has been directed to the study of the consensus problem — i.e. the problem of making the scalar states of a set of agents converge to the same value under local communication constraints (7, 23, 43, 50, 90, 91) — and of its many applications. One of such applications, namely wireless sensor networks and in general peer-to-peer networks, is now the focus of a huge amount of research in many disciplines of information technology. The reason why the distributed average problem has received great attention is that it allows to achieve tasks with a minimum overhead of communication since it requires only local information exchange between nodes directly connected, i.e. no routing is needed and so no congestion due to network traffic is generated. One of the networks in which this is desirable is the internet in which the availability of information on the average of local quantities generated by users behavior is of great relevance for statistical analysis, marketing, security and so on. If such objectives can be achieved without unnecessarily overloading network nodes and user bandwidth the relevance of such algorithms becomes clear.

A different kind of networks are wireless embedded sensor networks, intended to be composed of a huge number of cheap wireless sensors scattered around a target, be it a city, a forest, a war field or a polluted area. By definition if a

wireless sensor is to be cheap it has to consume very little power for achieving its task and to this end the ability to retrieve the average of the measurements with only local packet exchange is of great relevance.

Many previous works on the consensus problem and gossip algorithms (23, 38, 43, 58, 90, 91, 92, 93, 94) are based on bidirectional communications and so represent the network through an undirected graph, possibly with a switching topology. In (38) a study of convergence times of gossip algorithms based on pairwise random communications is presented for different network topologies. In (93) consensus on the average in presence of intermittent links and noise is addressed. In (94) the problem of designing the topology to maximize the rate of convergence of average consensus is addressed taking into account communication costs and constraints.

The requirement of bidirectional communications requires synchronization between transmitter and receiver and some overhead required by the communication protocols like acknowledgments. Furthermore even if a set of nodes can communicate between each other, communications are inherently sequential and pairwise if they are not done in the form of broadcasts. An attempt to use broadcasts in the distributed average problem has been made with gossip algorithms, the tradeoff of this approach is that agreement is only reached in the form of a random variable whose expectation corresponds to the average of the initial measurements and whose shape is deeply affected by the sequence in which the nodes perform broadcasts.

A different approach to this issue is the use of distributed Kalman filtering based on consensus (53, 54). A couple of years ago this problem was solved by adapting the optimal Kalman gain of such filter with respect to the outflow of each node (55) to achieve consensus on the average on arbitrary strongly connected digraphs. The proposed technique was time-variant and proposed as a decentralized iterative algorithm with synchronized updates. In this chapter, that is a journal version of (95), we propose an alternative approach based on gossip.

In (43) the study of consensus on digraphs was motivated by reduction in communication costs. Unfortunately the conclusion of the authors was that consensus on the average is achievable only for balanced digraphs, i.e. graphs in which the in-degree and out-degree of each node are the same.

Starting from this, we developed a new algorithm, with the same feature of

Laplacian-based consensus, that can achieve the same objective for the wider class of arbitrary digraphs. This generalizes the consensus problem and allows a consistent reduction of complexity since it allows the use of only broadcasting as communication mean.

Furthermore wireless sensor networks are usually required to perform tasks more complex than just computing the average of some quantity. We argue that an algorithm that allows consensus on directed graphs can actually be implemented as simple and small "overhead" on normal communication between the sensors. For instance with the ZigBee protocol for wireless networks we have packets with a maximum payload of around 104 bytes, which it is clearly much more than what is required to just send a scalar integer value of 16 bits. We argue that such consensus protocol could have a more meaningful and real application if thought as network overhead for distributed estimation purposes that does not actually "increase" the load in the network. Since no specific acknowledge or response is required, no dedicated communication is required and only the usual communication due to data transfer between nodes for other purposes is needed. With the previous assumption while the nodes use only mono-directional communications, they always know their out-degree.

Finally, the proposed algorithm poses new theoretical questions on stability of gossip algorithms since it is an instance of gossip algorithm in which local interactions are based on asymmetric, non-contractive matrices with possibly both positive and negative elements taken from a set all the products of which converge (36).

Note that, even if a formal proof of convergence of the algorithm we propose is missing, a series of simulations are presented to illustrate the effectiveness of the approach. In particular, different network topologies have been considered that are scalable in the number of nodes, and for such topologies the dependence of the convergence times upon the number of nodes is shown. Finally, we present a series of simulations to compare the proposed algorithm with other gossip algorithms known in the literature.

We point out that in Cai K. et al. 2009-2010(96, 97, 98) by exploiting a similar approach presented in Franceschelli et al. 2008-2010 (95, 99, 100, 101) based on the use of "surplus" or "storage" variables to keep memory of the initial network average in the network the problem of consensus on directed graphs is addressed

in the case of agents with uniformly quantized states.

Problem description

We model the network of agents as a directed graph $\mathcal{G}(t) = \{V, E(t)\}$, with $V = \{1, \dots, n\}$ the set of nodes (or vertices) that represent the agents, $E(t) \subseteq \{V \times V\}$ the time varying edge set that encodes the network topology, $(i, j) \in E(t)$ if and only if agent i may receive information from agent j at time t . In the following directed edges from j to i are considered to have their "tail" in j and the "head" in i .

The graph can be encoded through its $n \times n$ *adjacency matrix*

$$A(t) = \{a_{i,j}(t)\} \text{ with } a_{i,j}(t) = \begin{cases} 1 & \text{if } (i, j) \in E(t); \\ 0 & \text{otherwise.} \end{cases}$$

The in-degree of a node corresponds to the number of "heads" incident in such node while the out-degree is the number of "tails" incident on it.

We define the two $n \times n$ matrices

$$\Delta_{in}(t) = \text{diag}(\delta_{in,1}(t), \dots, \delta_{in,n}(t))$$

and

$$\Delta_{out}(t) = \text{diag}(\delta_{out,1}(t), \dots, \delta_{out,n}(t))$$

where $\delta_{in,i}$ and $\delta_{out,i}$, for $i = 1, \dots, n$, are respectively the in-degree and out-degree of node i .

The *Laplacian* of a time-varying digraph is defined as

$$\mathcal{L}(t) = \Delta_{in}(t) - A(t). \quad (5.1)$$

It is a positive semi-definite matrix and weak diagonally row dominant. Defining $\mathbf{0}$ and $\mathbf{1}$ column vectors whose n elements are all, respectively, zeros and ones, we have that $\mathcal{L}(t)\mathbf{1} = \mathbf{0}$ by construction.

To each node i for $i = 1, \dots, n$ is associated a scalar $x_i(t)$ with an arbitrary initial value $x_i(0) = x_{i0}$.

Furthermore we define the set of neighbors of node i as $\mathcal{N}_i(t) = \{j : (j, i) \in E\}$ and with $|\mathcal{N}_i(t)|$ its cardinality. We point out that since the graph is directed, node i may be a neighbor of node j while node j is not a neighbor of node i .

Note that the underlying topology of the network is *deterministic* and specifies the edges that *may* be selected by the gossip algorithm at any time. What is random is the selection of the node that performs a broadcast, involving only the edges connected to it. So at any time instant the interaction topology is a set of neighbors of the broadcaster node which is taken at random.

Our objective is to find a decentralized control law that satisfies the network topology constraints given by $\mathcal{G}(t)$ and achieves consensus on the average on the initial states.

5.2 Proposed algorithm

In our approach we associate to each node i for $i = 1, \dots, n$, in addition to $x_i(t)$ on which value a consensus on the average is sought, a companion variable $z_i(t)$ with initial value $z_i(0) = 0$. In the following we study a gossip algorithm based on mono-directional communications. Each node at each instant of time is then either transmitting information, receiving information or in an idle state.

RULE 1, Transmitter state update, node i

$$\begin{cases} x_i(t+1) = x_i(t), \\ z_i(t+1) = 0. \end{cases} \quad (5.2)$$

RULE 2, Receiver state update, node $j \in \mathcal{N}_i(t)$

$$\begin{cases} x_j(t+1) = \frac{x_j(t)+x_i(t)}{2} + 0.5z_j(t) + \frac{z_i(t)}{2\delta_{out,i}(t)}, \\ z_j(t+1) = \frac{x_j(t)-x_i(t)}{2} + 0.5z_j(t) + \frac{z_i(t)}{2\delta_{out,i}(t)}. \end{cases} \quad (5.3)$$

RULE 3, Idle nodes, $k \neq i, k \notin \mathcal{N}_i(t)$

$$\begin{cases} x_k(t+1) = x_k(t), \\ z_k(t+1) = z_k(t). \end{cases} \quad (5.4)$$

Algorithm 5 (Extended Gossip based on Broadcasts (EGB))

1. Let $t=0$, let $\mathbf{x}(0) = \mathbf{x}_0$ and $\mathbf{z}(0) = \mathbf{0}$.

2. A node i at random executes **RULE 1**
3. Each node $j \in \mathcal{N}_i(t)$ that listens to the broadcast applies **RULE 2**.
4. All the other nodes $k \neq i, k \notin \mathcal{N}_i(t)$ keep their state variable and their companion variable constant (**RULE 3**).
5. Let $t = t + 1$ and go back to step 2. ■

These interaction rules can be explained in simple words.

- *The transmitter node i* broadcasts its state value x_i to all nodes $j \in \mathcal{N}_i$. In doing so, it knows its out-degree and it also broadcasts the value $z_i(t)/\delta_{out,i}(t)$ by dividing the value of the companion variable by the number of nodes that receive the information. The transmitter node i does not change its value of $x_i(t)$ while it resets to 0 the companion variable $z_i(t)$.

- *The receiver nodes* update their $x_j(t)$ variable by computing the average between their and the received state value. Furthermore they correct their update by a fraction of their companion variable $z_j(t)$ and a fraction of the companion variable of the transmitter node $z_i(t)$. The receiver nodes update their companion variable by adding up several terms, designed to preserve the average of the network at each iteration while converging to the average of the initial measurements.

The sequence of nodes that perform the broadcast at the different time instants $t \in \mathbb{N}$ defines a signal $\mathcal{J}(t)$. As an example if node 3 is the transmitter node at time $t = 0$ then $\mathcal{J}(0) = 3$.

Assume that at time t node $i = \mathcal{J}(t)$ performs a broadcasts, the interaction topology at time t is represented by a graph $\mathcal{G}_i(t)$, obtained from $\mathcal{G}(t)$ removing all arcs whose tail is not node i . We let $A_i(t)$, $\Delta_{in,i}(t)$ and $\mathcal{L}_i(t)$ denote, respectively, the incidence matrix, the in-degree matrix and the Laplacian of this graph.

Let us define

$$P_i(t) = I - 0.5\mathcal{L}_i(t), \quad \hat{\Gamma}_i(t) = \frac{A_i(t)}{2\delta_{out,i}(t)} + 0.5\Delta_{in}(t),$$

$$\Gamma_i(t) = \frac{A_i(t)}{2\delta_{out,i}(t)} - 0.5\Delta_{in}(t) + (I - \mathbf{e}_i\mathbf{e}_i^T),$$

where I is the identity matrix and \mathbf{e}_i is the i -th canonical basis vector of dimension n .

We denote

$$C_i(t) = \begin{bmatrix} P_i(t) & \hat{\Gamma}_i(t) \\ I - P_i(t) & \Gamma_i(t) \end{bmatrix}. \quad (5.5)$$

Under the decentralized state update rule (5.2), (5.3) and (5.4), the system dynamics at time t , is:

$$\begin{bmatrix} \mathbf{x}(t+1) \\ \mathbf{z}(t+1) \end{bmatrix} = C_i(t) \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{z}(t) \end{bmatrix}, \quad i = \mathcal{J}(t). \quad (5.6)$$

Remark 5.2.1 *It is assumed that at each instant of time each node has a strictly positive probability of broadcasting its state to the neighboring nodes. This assumption is to model the inherent asynchrony of wireless communications between sensor nodes. The results on the convergence properties of algorithms developed with this assumption hold for any deterministic scheduling of the communications between the nodes because the order in which the updates are performed is not relevant to the stability of the equilibrium point of the algorithm.*

We point out that at each time instant t only the broadcaster node i has to know the number of local neighbors $|\mathcal{N}_i|$, which correspond to its out-degree at time t . The most trivial case in which such assumption holds is when the sensor network is not just executing a distributed average algorithm but is also providing other services that require the knowledge of the network topology. Our algorithm reduces the resources dedicated to the distributed average algorithm by not requiring acknowledgements to the transmitted information and fully exploits dense proximity networks by using broadcasts.

The main difference between the proposed algorithm and other consensus algorithms based on broadcasts is that in our approach convergence toward the exact initial average is ensured. In the other cases presented in the literature the consensus state is not the initial average of the measurements, but it may be any value inside the convex hull spanned by the initial conditions in the network depending on the sequence of broadcasts (39, 102).

Furthermore given that the sensor network is distributed in space, any pair of nodes sufficiently far apart can perform a broadcast while not interfering with each other. The inherent parallelism of the network is fully exploited and is expected to greatly improve the convergence time of the proposed algorithm. Nonetheless

in the following we focus our attention in studying the stability of the equilibrium point of the algorithm leaving the study of its convergence time to future work. ■

In the following, for simplicity of explanation the dependence of C_i from t will be omitted.

5.3 Convergence properties

In this section we study the convergence properties of the algorithm. We first characterize the eigenstructure of matrices C_i and then we present a conjecture on the convergence to consensus.

Proposition 5.3.1 C_i is idempotent for any $i = 1, \dots, n$.

Proof: Using the general identities $A_i^2 = A_i \Delta_{in,i} = 0$ and $\Delta_{in,i} A_i = A_i$, one can readily verify that for all $i = 1, \dots, n$ it holds $C_i^2 = C_i$. □

Since C_i is idempotent, its eigenvalues are always either 0 or 1. Unfortunately since it is not symmetric, it represents an *oblique* projection which does not result in a contractive matrix in general.

We observe, however, that the system is conservative.

Proposition 5.3.2 System (5.6) evolves on the hyperplane

$$\mathbf{1}^T \mathbf{x}(t) + \mathbf{1}^T \mathbf{z}(t) = \mathbf{1}^T \mathbf{x}(0) + \mathbf{1}^T \mathbf{z}(0).$$

Proof: For all $i = 1, \dots, n$, the row vector $[\mathbf{1}^T \ \mathbf{1}^T]$ is a left eigenvector for matrix C_i associated to eigenvalue 1, because it holds $[\mathbf{1}^T \ \mathbf{1}^T] C_i = [\mathbf{1}^T (P_i + I - P_i) \ \mathbf{1}^T (\hat{\Gamma}_i + \Gamma_i)] = [\mathbf{1}^T \ \mathbf{1}^T]$. □

Since the system is autonomous and the companion initial state can be arbitrary chosen, we select $\mathbf{z}(0) = \mathbf{0}$. With such assumption we obtain that the information about the average of the initial state is preserved despite communications are mono-directional and asynchronous. In particular, if there exists a time t in which $\mathbf{z}(t) = \mathbf{0}$, then at that time t it holds $\mathbf{1}^T \mathbf{x}(t) = \mathbf{1}^T \mathbf{x}(0)$.

In the following we provide an analysis of the equilibrium states of the proposed algorithm and corroborate the conjectured asymptotic stability of the equilibrium states by simulations.

Let us now consider the equilibrium points.

Proposition 5.3.3 *The consensus state in which $\mathbf{x}(t) = \alpha \mathbf{1}$ for some scalar α and $\mathbf{z}(t) = \mathbf{0}$ is an equilibrium state for system (5.6).*

Proof: For all $i = 1, \dots, n$, the column vector $[\mathbf{1}^T \ \mathbf{0}^T]^T$ is a right eigenvector for matrix C_i associated to eigenvalue 1, because it holds

$$C_i(t) \begin{bmatrix} \mathbf{1} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} P_i \mathbf{1} \\ (I - P_i) \mathbf{1} \end{bmatrix} = \begin{bmatrix} \mathbf{1} \\ \mathbf{0} \end{bmatrix},$$

due to the Laplacian property $\mathcal{L} \mathbf{1} = \mathbf{0}$. □

We now consider the null space of the consensus matrices.

Proposition 5.3.4 *For all $i = 1, \dots, n$, the kernel of $C_i(t)$ has dimension $\dim(\text{Ker}(C_i)) = |\mathcal{N}_i(t)| + 1$.*

Proof: It can be shown that the multiplicity of the null eigenvalue of C_i is $|\mathcal{N}_i(t)| + 1$. A set of linearly independent eigenvectors that form a basis of the null space are:

- for $j \in \mathcal{N}_i$ a vector $\mathbf{v}_j = [\mathbf{e}_j^T \ -\mathbf{e}_j^T]^T$;
- a vector $\hat{\mathbf{v}}_i = [\hat{\mathbf{x}}_i^T \ \hat{\mathbf{z}}_i^T]^T$ with

$$\hat{\mathbf{x}}_i(j) = \begin{cases} 1 & \text{if } j \in \mathcal{N}_i, \\ 0 & \text{otherwise} \end{cases}$$

and

$$\hat{\mathbf{z}}_i(j) = \begin{cases} -2\delta_{out,i} & \text{if } j = i, \\ 1 & \text{if } j \in \mathcal{N}_i, \\ 0 & \text{otherwise} \end{cases}.$$

□

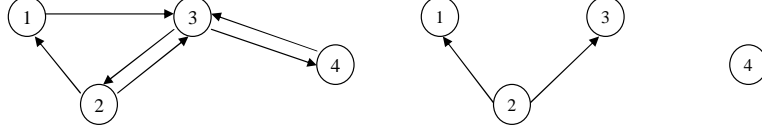


Figure 5.1: Network considered in Example 5.3.5 (left). Interaction topology when node 2 performs a broadcast in Example 5.3.5 (right).

Example 5.3.5 *Let us consider the network on the left of Fig. 5.1. When node 2 performs a broadcast, the interaction topology is represented by a directed graph, shown on the right of Fig. 5.1. The adjacency matrix for the resulting graph is*

$$A_2 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Following our previous definitions, $\delta_{out,2} = 2$, and we have:

$$P_2 = \begin{bmatrix} 1/2 & 1/2 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1/2 & 1/2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\Gamma_2 = \begin{bmatrix} 1/2 & 1/4 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1/4 & 1/2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \hat{\Gamma}_2 = \begin{bmatrix} 1/2 & 1/4 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1/4 & 1/2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix};$$

finally

$$C_2 = \left[\begin{array}{cccc|cccc} 1/2 & 1/2 & 0 & 0 & 1/2 & 1/4 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 1/2 & 0 & 0 & 1/4 & 1/2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 1/2 & -1/2 & 0 & 0 & 1/2 & 1/4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1/2 & 1/2 & 0 & 0 & 1/4 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array} \right].$$

By Proposition 5.3.4 the following is a basis of linearly independent eigenvectors for the null space:

$$\begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_3 & \hat{\mathbf{v}}_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & -4 \\ 0 & -1 & 1 \\ 0 & 0 & 0 \end{bmatrix}.$$

■

Now we consider a property that holds for strongly connected graphs.

Proposition 5.3.6 *If*

$$\hat{\mathcal{G}}[t_1, t_2] = \bigcup_{t=t_1}^{t_2} \mathcal{G}_i(t), \quad i = \mathcal{J}(t)$$

is strongly connected, then

$$\dim \left(\bigvee_{t=t_1}^{t_2} \ker(C_i(t)) \right) = 2n - 1,$$

where \vee denotes the linear combination of vector spaces.

Proof: To show this, let us take the union of all basis vectors for the null spaces of all matrices C_i , as defined in the proof of Proposition 5.3.4. Since $\hat{\mathcal{G}}[t_1, t_2]$ is strongly connected (sufficient condition), each node is at least once a transmitter and at least once a receiver. Thus combining all vectors we obtain the following matrix

$$V = [\mathbf{v}_1 \ \cdots \ \mathbf{v}_n \ \hat{\mathbf{v}}_1 \ \cdots \ \hat{\mathbf{v}}_n] = \begin{bmatrix} I & A(t) \\ -I & A(t) - 2\Delta_{out}(t) \end{bmatrix}.$$

By elementary row operations we show this matrix to be equivalent to

$$\begin{bmatrix} I & A(t) \\ 0 & 2A(t) - 2\Delta_{out}(t) \end{bmatrix} = \begin{bmatrix} I & A(t) \\ 0 & -2\mathcal{L}_{out}(t) \end{bmatrix}$$

where $\mathcal{L}_{out} = \Delta_{out} - A$ denotes the out-degree Laplacian, whose rank is $n - 1$ if the graph is strongly connected.

Thus matrix V has rank $2n - 1$ and this proves the result. \square

Thanks to the above propositions the following important result can be proved.

Proposition 5.3.7 *If $\forall t > 0$ there exists $T(t) > 0$ such that*

$$\hat{\mathcal{G}}[t, t + T(t)] = \bigcup_{\tau=t}^{t+T(t)} \mathcal{G}_i(\tau), \quad i = \mathcal{J}(\tau),$$

is strongly connected, then the subspace

$$\text{span} \left(\begin{bmatrix} \mathbf{1} \\ \mathbf{0} \end{bmatrix} \right)$$

is the largest invariant subspace for system (5.6).

Proof: The fact that $\text{span} \left(\begin{bmatrix} \mathbf{1}^T \\ \mathbf{0}^T \end{bmatrix} \right)$ is an invariant subspace was shown in Proposition 5.3.3. The fact that it is the largest invariant subspace follows from Proposition 5.3.6. In fact, the strong connectedness of $\hat{\mathcal{G}}[t, t + T(t)]$ implies the existence of a basis composed by vector $\begin{bmatrix} \mathbf{1}^T \\ \mathbf{0}^T \end{bmatrix}^T$ plus a set of vectors that span $\mathcal{K} = \bigvee_{\tau=t}^{t+T(t)} \ker(C_i(\tau))$. Hence any vector that has initially a component along a basis vector of \mathcal{K} will have that component eventually filtered. \square

We point out that the assumption of strong connectivity of the network topology *does not* require a *fixed* and constant time window T over which it is strongly connected. For Proposition 5.3.7 to hold it is sufficient that for any t there exists a $T(t) > 0$ possibly varying but finite in which $\hat{\mathcal{G}}[t, t + T(t)]$ is strongly connected. Note that such assumption is one of the most general assumptions that can be made regarding network connectivity because if for some time instant t there does not exist a $T(t) > 0$ in which $\hat{\mathcal{G}}[t, t + T(t)]$ is strongly connected, then starting from such t there exists at least a node not reachable from all the others.

Now we state the main conjecture regarding the proposed algorithm for which a formal proof is missing, further evidence of the convergence properties of the algorithm is shown by simulations.

Conjecture 5.3.8 *If $\forall t$, there exists $T(t) > 0$ such that*

$$\hat{\mathcal{G}}[t, t + T(t)] = \bigcup_{\tau=t}^{t+T(t)} \mathcal{G}_i(\tau), \quad i = \mathcal{J}(\tau)$$

is strongly connected, if the system evolves according to the state update rule described by (5.6) with $\mathbf{z}(0) = \mathbf{0}$, then:

$$\lim_{t \rightarrow \infty} \mathbf{x}(t) = \frac{\mathbf{1}\mathbf{1}^T}{n} \mathbf{x}(0).$$

■

The above conjecture is validated by several numerical experiments, some of which are reported in the following simulations section, and by the following theoretical observations.

First, by Proposition 5.3.3, it holds that $[\alpha \mathbf{1}^T \ \mathbf{0}^T]^T$ is an equilibrium state for system (5.6) for some scalar α . Secondly, by Proposition 5.3.2, being by assumption $\mathbf{z}(0) = \mathbf{0}$, it holds that $\mathbf{1}^T \mathbf{x}(t) + \mathbf{1}^T \mathbf{z}(t) = \mathbf{1}^T \mathbf{x}(0)$ for any t . Finally, by Proposition 5.3.7, $\text{span} \left([\mathbf{1}^T \ \mathbf{0}^T]^T \right)$ is the largest invariant subspace for system (5.6).

Unfortunately, the problem of deciding whether the random product of a finite set of matrices converges is still an open problem in matrix theory and all the results are either not applicable or relate to classes of matrices not suitable for our purposes.

The proposed conjecture, while intuitive and validated by simulation results poses great difficulties in its proof. First, the description of the stability of a switching linear system of the type of system (5.6) has been treated only for simple cases in which at each instant of time t the system matrix is at least para-contractive. Others have used Markov chain theory and applied it to the study of consensus problems. Some other results use the Common Lyapunov function approach to study the convergence properties of such systems. In our case the system matrix C is not symmetric, is not contractive, nor is it with non-negative elements thus invalidating almost all known general results for stability analysis of such system.

5.4 Simulations

In this section we provide simulations in order to corroborate the algorithm analysis.

Each node is initialized with values between 0 and 1 chosen at random from a uniform distribution. The simulations are scaled respect to the number of nodes in the network for topologies whose features do not change as the number of nodes is increased.

We now define an error performance index:

$$V(t) = \|\mathbf{x}(t) - \frac{\mathbf{1}\mathbf{1}^T}{n}\mathbf{x}(0)\|_2 + \|\mathbf{z}(t)\|_2.$$

Such index is a measure of how far the state of the network is from its equilibrium state, namely

$$\mathbf{x}(\infty) = \frac{\mathbf{1}\mathbf{1}^T}{n}\mathbf{x}(0), \quad \mathbf{z}(\infty) = 0.$$

The convergence time is function of the network topology and the number of nodes. Let us define the convergence time proposed in the simulations as

$$T_{con} = \inf\{\tau > 0 : \frac{V(\tau)}{V(0)} < 0.05\}, \quad (5.7)$$

i.e. the number of broadcasts needed such that the error modeled by the performance index $V(t)$ becomes less than 5% of its value at initialization time. Note that, since $V(t)$ is not a non-increasing function of t , the fact that $V(t)$ is less than 5% of its initial value at a given time t , does not imply that it remains less than this value for all $\tau > t$.

The following simulations show the average value of T_{con} over 100 realizations of the algorithm. We consider three different topologies:

- *Fully connected topology*: each broadcast is received by every node in the network.
- *Line topology*: each broadcast is received only by the 2 adjacent nodes of the broadcaster except for the one of the end nodes.
- *Square grid topology*: each broadcast is received only by those nodes in proximity of the broadcaster in a square grid.

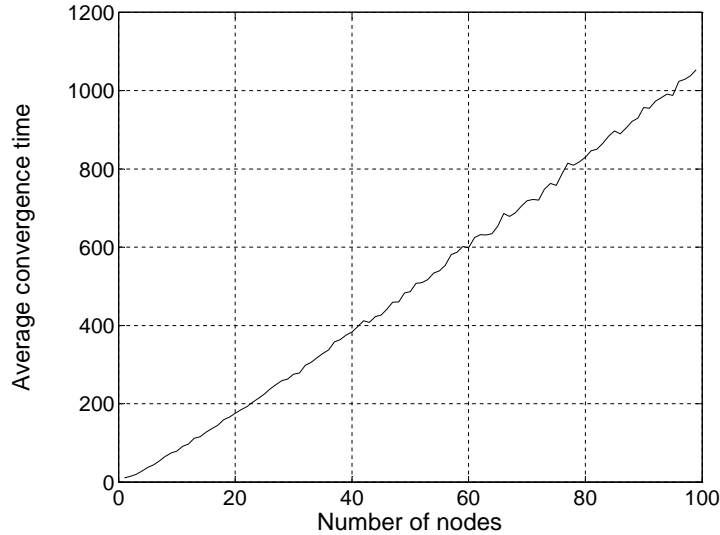


Figure 5.2: Average value of T_{con} for a fully connected network with respect to the number of nodes.

The selected topologies specify the edges that *may* be selected by the gossip algorithm at any given time instant. We simulated the algorithm on regular and well defined graphs topologies so that the presented simulations results are easily reproducible. Furthermore by selecting well defined topologies we can easily compute average convergence times when transmitter nodes are selected at random. On the contrary, in the case of random topologies the convergence time not only would vary depending on the edge selection process but also due to the given realizations of the random topologies, thus greatly increasing the variance of the convergence time and impairing the clarity of the results.

Fig. 5.2 is relative to simulations performed on fully connected networks with an increasing number of nodes. Simulations show that the average convergence time scales linearly with the number of nodes as $T_{con} \approx 10n$.

In Fig. 5.3 the simulations are performed for line networks of increasing number of nodes. Simulations show that the convergence time scales polynomially respect to the diameter of the graph which is equal to $n - 1$ for line graphs.

In Fig. 5.4 the simulations are performed for square grid networks of increasing number of nodes, such that the total number of nodes is perfect square. Simulations show that the convergence time scales polynomially respect to the diameter of the graph which is equal to $\sqrt{n} - 1$ for grid graphs.

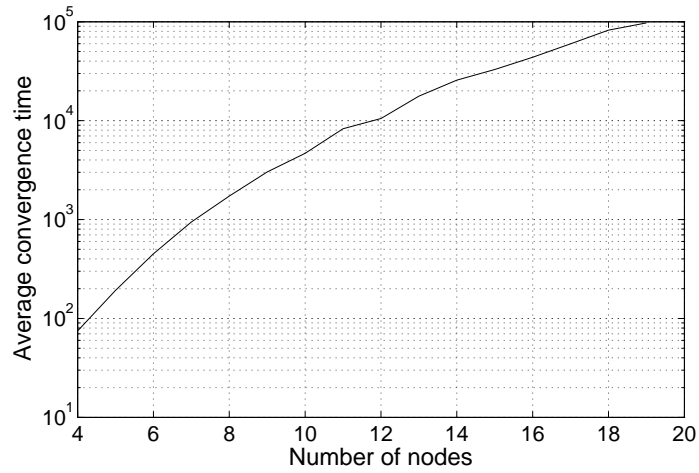


Figure 5.3: Average value of T_{con} for a line network with respect to the number of nodes.

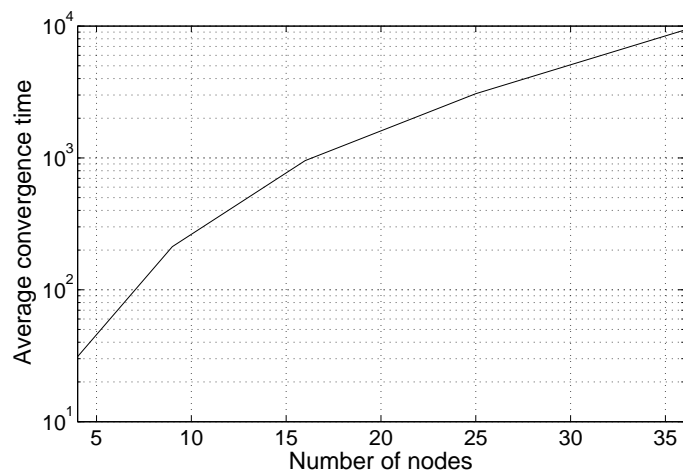


Figure 5.4: Average value of T_{con} for a square grid network with respect to the number of nodes.

The improvement of the proposed algorithm respect to other gossip algorithms is that by using broadcasts the inherent parallelism in a distributed network is fully exploited between all the nodes and not only between nodes not directly connected. This feature is especially relevant in networks with small diameter where few nodes have a very high out-degree such as small world networks.

A comparison with the Standard Gossip based on Broadcast

We now compare the proposed algorithm with a simple gossip algorithm illustrated in (39, 102) without our average preserving properties. In these works a gossip algorithm based on broadcasts is studied which can be summarized as follows.

Algorithm 6 (Standard Gossip based on Broadcasts (SGB))

- *At each instant of time a node broadcasts its value to its neighbors.*
- *If at any time a node listens to a broadcast, it computes the average between its state and the broadcaster state. It then takes this new value as its state.*
- *Repeat until all the nodes have the same value.* ■

Remark 5.4.1 *The following numerical simulations compare our algorithm with the SGB algorithm (39, 102) in terms of convergence rate. However, the main difference among the two algorithms is qualitative: the proposed algorithm converges exactly to the average of the initial states while the SGB algorithm may converge to any point inside the convex hull spanned by the initial network conditions depending on the sequence of broadcasts. It has been shown that, if the number of nodes is sufficiently high and broadcasts occur at random, then the SGB converges statistically sufficiently close to the initial average. Indeed, the main point of adding more sensors to a network to measure the same scalar quantity is to get out of cheap sensors a measurement whose precision is far greater than the precision of the single units but this can be achieved through averaging only if the network does actually converge to the initial average.* ■

Now, since the SGB algorithm does not converge to the average, we define a different performance index which becomes zero only when each node has the same value:

$$\hat{V}(t) = \|\mathbf{x}(t) - \frac{\mathbf{1}\mathbf{1}}{n}\mathbf{x}(t)\|.$$

In the simulations of Algorithm 6 we adopt the following definition of convergence time:

$$\hat{T}_{con} = \inf\{\tau > 0 : \frac{\hat{V}(\tau)}{\hat{V}(0)} < 0.05\}. \quad (5.8)$$

To corroborate the simulation results, we show the average error over the 100 realizations respect to the initial average of the network at $t = \hat{T}_{con}$, namely

$$Err(\hat{T}_{con}) = \frac{\|\mathbf{x}(\hat{T}_{con}) - \frac{\mathbf{1}\mathbf{1}}{n}\mathbf{x}(0)\|}{n}.$$

The simulations are again performed for the three topologies taken into consideration:

- Simulations on a *Fully connected topology* are shown in Fig. 5.5. The convergence time results to be constant when increasing the number of nodes. This can be explained by the fact that being the network fully connected, at each broadcast all the nodes receive information disregarding the size of the network. In Fig. 5.6 the average error at convergence time is shown for the same simulations: the error decreases as more nodes are added.

- Simulations on a *Line topology* are shown in Fig. 5.7. The convergence time appears to be polynomial in the number of nodes and an order of magnitude lower than our proposed algorithm. In Fig. 5.8 is shown the average error at convergence time for the same simulations: also in this case adding more nodes reduces the final average error.

- Simulations on a *Grid topology* are shown in Fig. 5.9. The convergence time appears to be polynomial and faster than our proposed algorithm. In Fig. 5.10 is shown the average error at convergence time for the same simulations: again it decreases when increasing the number of nodes.

Summarizing, simulations show that the SGB algorithm achieves better convergence times for growing network size trading off a finite error in the final state that decreases with network size. Such trade-off is the highest when the number of nodes is small.

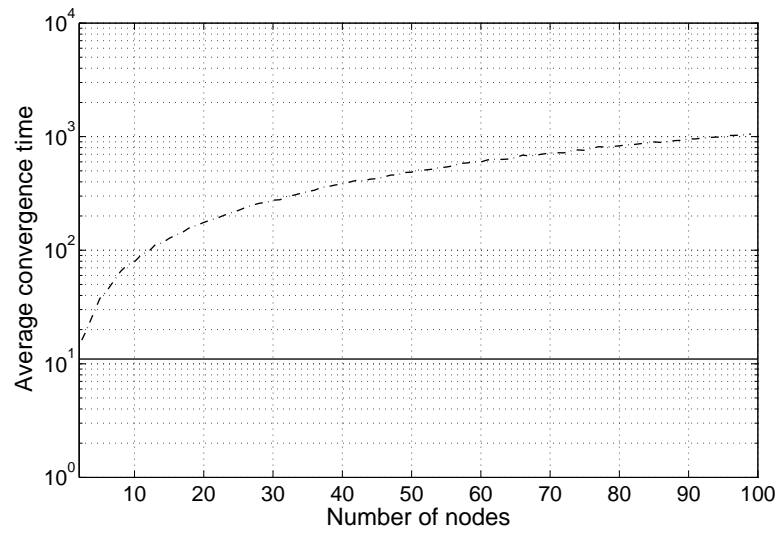


Figure 5.5: Average value of T_{con} (dashed line) and \hat{T}_{con} (continuous line) for a fully connected network with respect to the number of nodes.

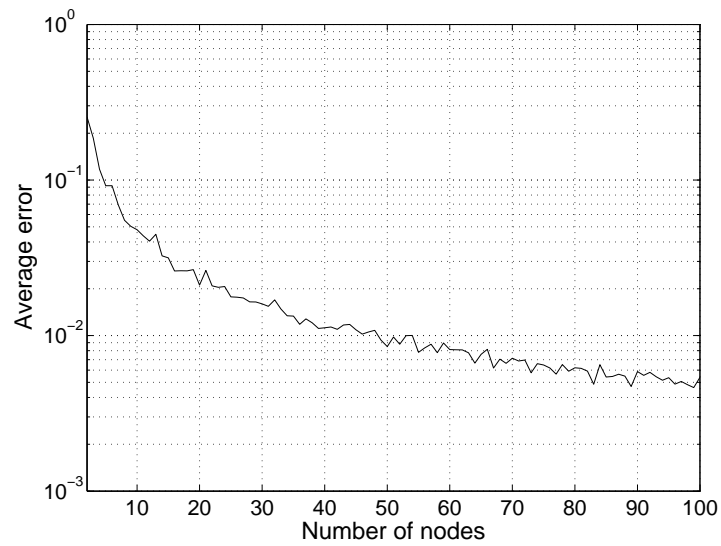


Figure 5.6: Average value of $Err(\hat{T}_{con})$ for a fully connected topology using the SGB algorithm.

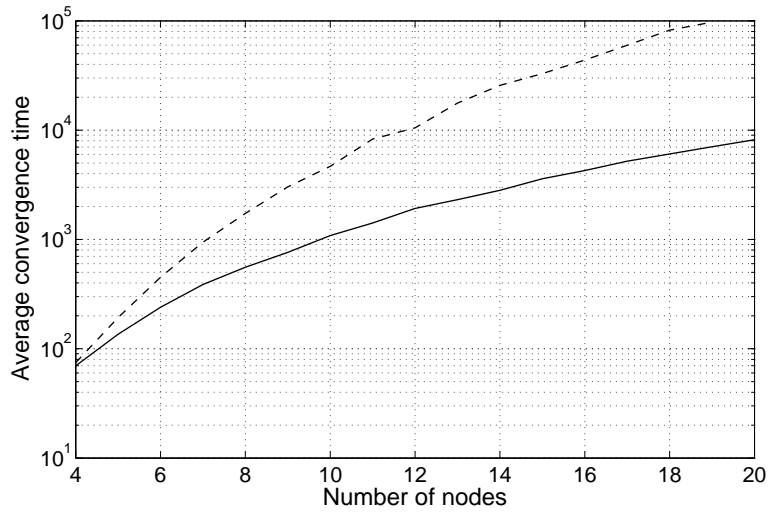


Figure 5.7: Average value of T_{con} (dashed line) and \hat{T}_{con} (continuous line) for a line topology with respect to the number of nodes.

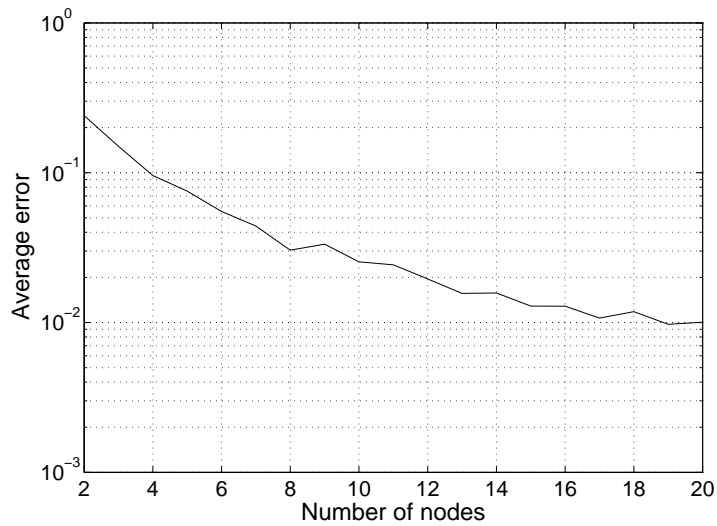


Figure 5.8: Average value of $Err(\hat{T}_{con})$ for a line topology using the SGB algorithm.

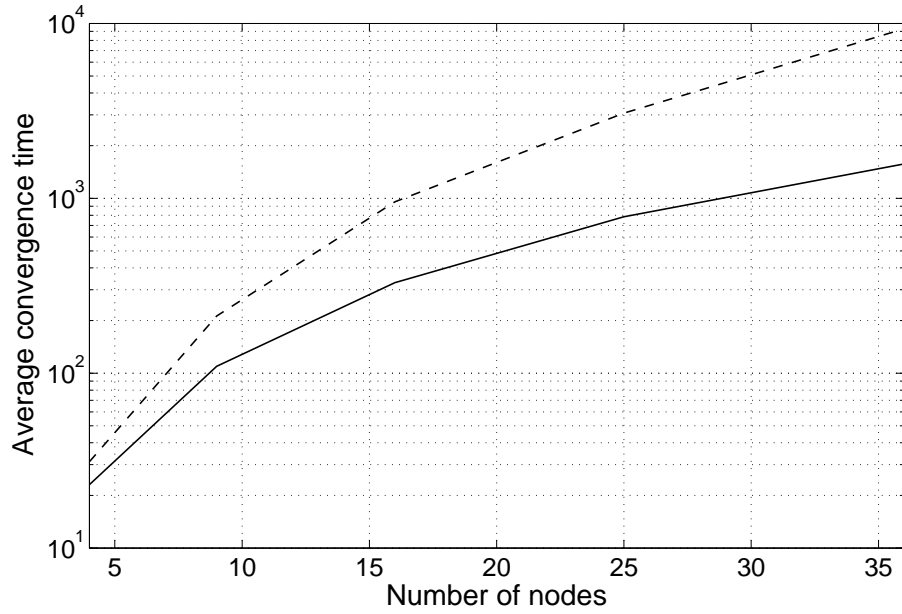


Figure 5.9: Average value of T_{con} (dashed line) and \hat{T}_{con} (continuous line) for a square grid topology with respect to the number of nodes.

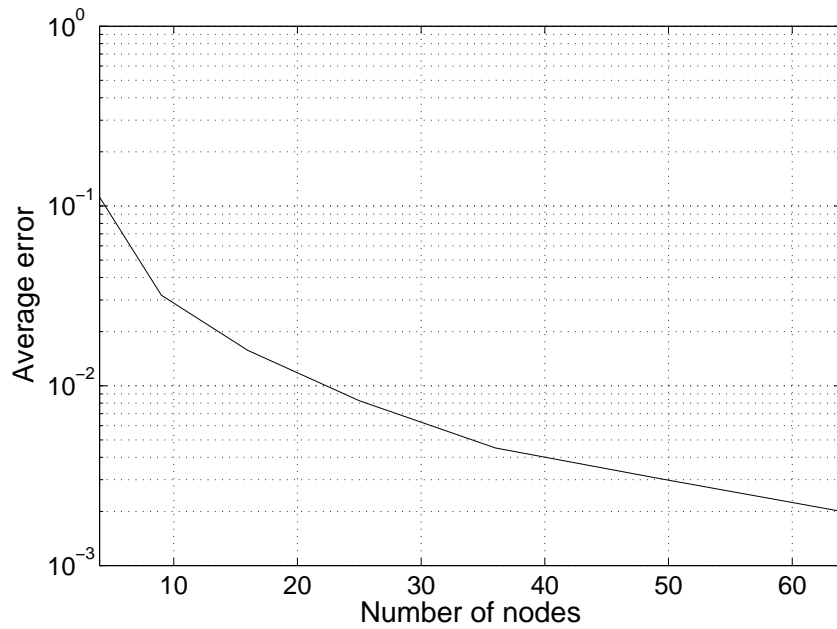


Figure 5.10: Average value of $Err(\hat{T}_{con})$ for a square grid topology using the SGB algorithm.

A comparison with standard Gossip based on pairwise averaging

We now compare our algorithm with the standard gossip algorithm based on random pairwise averaging (SGP) (38, 92, 93, 94). We believe that comparing the performances of these two algorithms is significant because they both converge exactly to the average of the initial state of the network.

A gossip algorithm based on pairwise communications can be summarized as follows.

Let $\mathcal{G} = \{V, E\}$ represent the network topology at time t . Let $x(t)$ be an n -element vector representing the state of the network where the generic element $x_i(t)$ is the state of node i . Let $\mathcal{E}(t) : \mathbb{R}^+ \rightarrow E$ be the edge selection process that at any given time instant outputs an edge $e_{ij} \in E$.

Algorithm 7 (Standard Gossip based on pairwise averaging)

- Let $t = 0$ and $x(t) = x_0$ be the initial state of the network.
- Select the edge e_{ij} given by the edge selection process $\mathcal{E}(t)$.

- Let

$$\begin{cases} x_i(t+1) = \frac{x_i(t) + x_j(t)}{2}, \\ x_j(t+1) = \frac{x_i(t) + x_j(t)}{2}. \end{cases}$$

- Let $t = t + 1$ and go back to step 2. ■

Remark 5.4.2 Before presenting the results of numerical simulations, let us discuss some qualitative difference between the two algorithms: the proposed algorithm works on arbitrary strongly connected directed graphs while the SGP algorithm requires bidirectional information exchange at each time step and thus can only be applied to connected undirected graphs. For this reason we can compare the performance of the two algorithms only for those restricted network topologies that satisfy the connectivity requirements of the SGP algorithm.

Moreover, several gossip algorithms based on pairwise averaging differ in the edge selection process, for instance in (38) the nodes are selected according to a

Broadcast radius	Transmission power	Convergence time		Total Energy	
		SGP Algorithm	EGB Algorithm	SGP Algorithm	EGB Algorithm
1.10	1.23	1406	9405	1406	9406
1.57	2.43	931	3948	2095	8883
2.10	4.41	680	1271	2723	5086
2.97	8.82	528	421	4445	3544
3.10	9.61	494	341	4446	3077
4.10	16.81	445	253	7133	4059
4.39	19.27	451	241	8352	4455
5.10	26.01	426	227	10659	5684
5.80	33.64	434	236	14100	7678
6.10	37.21	433	233	15592	8391
7.22	52.13	434	228	21894	11511
8.63	74.48	426	232	30809	16810

Table 5.1: Simulation on a 49 node grid proximity network.

Poisson process. In general, the edge selection process affects the convergence time in absolute terms, while the sequence in which edges are chosen affects the number of iterations required, which indirectly affects the convergence time. To perform a single iteration of the GSP algorithm requires two transmissions, node i needs to send its state value to node j and node j has to reply to node i with its own state. In this analysis we neglect the complexity of data link and physical layers because, even with this crude simplification of the bidirectional communication process, we can show the advantage of our algorithm respect to the SGP algorithm in terms of the total energy required to achieve a given performance.

■

In the following simulations a set of 49 wireless sensors has been placed in a 7×7 grid spaced 1 unit of length [m] between each other. The network has been simulated by varying the total transmitted power of each node to compute how the energy consumed by the algorithms scale with the transmitted power on proximity graphs. Note that we take into consideration a grid network and not the more general set of random geometric graphs, to eliminate the contribution of the randomness of the topology to the energy spent during the algorithm execution, which is already random due to the particular edge selection process.

In these simulations we take inspiration from (94) that as in our case simulates changes in network topologies due to increases in transmitted power. Such changes of the communication range are proportional to the square root of the transmitted power taking into account total communication costs in the performance index.

To model the proximity range as function of the transmitted power we consider the standard equation for radio frequency communications:

$$P_r = \frac{P_t g_t g_r}{4\pi r^2}$$

where P_t and P_r are, respectively, the transmitted and received power in watt [W], g_t and g_r are the antenna gains of the receiver and the transmitter and r is the line of sight distance between transmitter and receiver. The receiver, given the technological constraints due to the electronics and noise, needs to receive at least a given power P_r to be able to decode the message sent.

In our simulations we simplify this model by allowing the effective communication radius r to scale as $r \propto \sqrt{P_t}$. The topology of the grid network then requires at least $P_t = 1$ to be connected, i.e. to allow each node to communicate bidirectionally with at least one neighbor. Increasing the transmitted power the number of neighbors that each node can communicate with increases until the transmitted power is enough to reach any node in the network, roughly for $r = 6\sqrt{2}$ and $P_t = r^2 = 72$.

Each transmission is assumed to last τ seconds and so consume $E = P_t\tau$ [J] units of energy. Again we simplify the simulation assuming $\tau = 1$ since the improvement of the proposed algorithm respect to the SGP algorithm is the reduction of total number of transmissions and does not depend on τ .

The execution of the algorithms is stopped as soon as the performance index

$$\tilde{V}(t) = \|\mathbf{x}(t) - \frac{\mathbf{1}\mathbf{1}^T}{n}\mathbf{x}(0)\|_2$$

reaches 10% of its initial value, i.e., we define the average convergence time as

$$\tilde{T}_{con} = \inf\{\tau > 0 : \frac{\tilde{V}(\tau)}{\tilde{V}(0)} < 0.1\}, \quad (5.9)$$

In Fig. 5.11 and 5.12 a comparison between the proposed algorithm (continuous line with square markers) and the SGP algorithm (dashed line with round markers) is shown. In particular, in Fig. 5.11 the total number of transmissions is plotted against the broadcast radius which affects the topology by increasing the number of neighbors of each agent in the grid. In Fig. 5.12 the total energy consumption is shown respect to the transmitted power at the nodes.

It can be seen that, when the number of neighbors of each node is small due to little transmitted power, the SGP algorithm performs better both in number of transmissions and energy saving. This is reasonable because there is no gain in performing broadcasts if the number of nodes that can listen to it is very small. As soon as the out-degree of each node increases for increasing values of the broadcast radius, the performance of the proposed algorithm based on broadcasts becomes significantly better of almost one order of magnitude in number of transmissions and energy saving. In particular, it is clear that the proposed algorithm achieves a better performance, in the case of the grid topology, achieving a minimum-energy consumption when the broadcast radius equals 3.1, corresponding to the

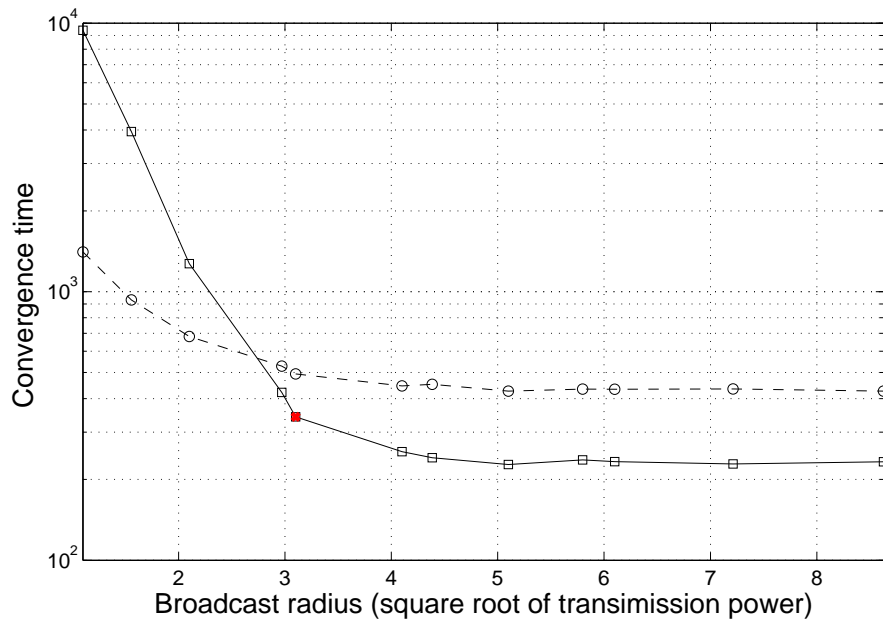


Figure 5.11: Total number of transmissions executed by the SGP algorithm (dashed line) and the proposed algorithm (continuous line) respect to the broadcast radius for a grid of sensors.

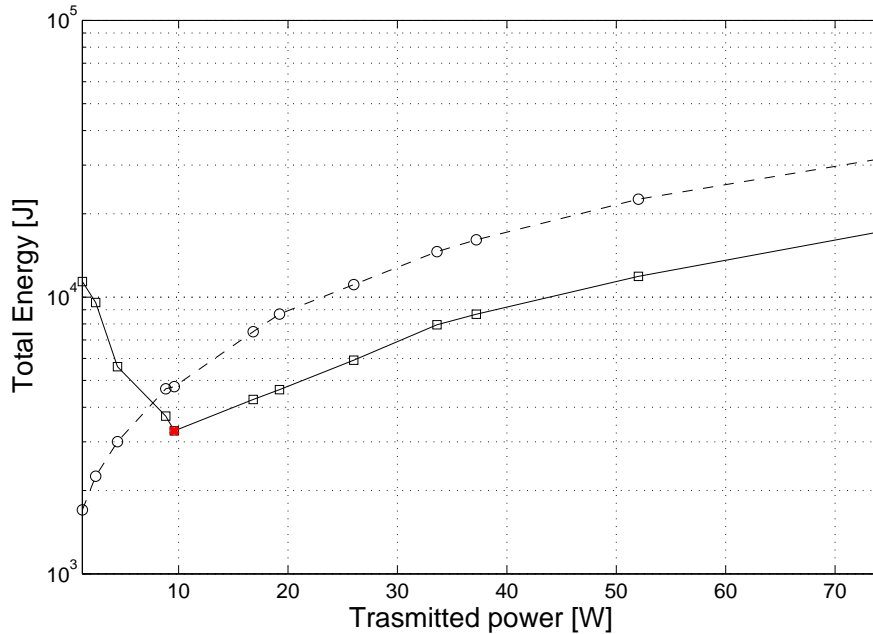


Figure 5.12: Total energy in [J] consumed by the execution of the SGP algorithm (dashed line) compared with the proposed algorithm based on broadcasts respect to normalized transmitted power [W] for a grid of sensors.

case in which each node can reach other nodes distant at most 3 rows or columns in the grid network (sample marked in red). The results of the above numerical simulations are also summarized in Table 5.1.

5.5 Conclusions

In this chapter a novel gossip algorithm based on broadcasts that achieves consensus on the average on arbitrary strongly connected digraphs has been proposed. The study of the convergence properties is preliminary and convergence is shown by simulations. The proposed algorithm is based on gossip and preserves the information about the average of the initial state during its execution. The main feature of our algorithm is that it converges *exactly* to the average of the initial state. Moreover, it works on arbitrary strongly connected digraphs.

A comparison with the standard gossip algorithm based on broadcast and with

the standard gossip based on pairwise averaging has also been made. Simulations show that the proposed algorithm achieves better convergence rates and energy saving than the standard gossip based on pairwise averaging if the number of neighbors of each node is sufficiently high.

Chapter 6

Gossip based consensus in absence of common reference frames

6.1 Introduction

In recent years multi agent systems have drawn the attention of a huge amount of researchers, for a representative example see (8, 15, 103, 104). In this framework Laplacian based controllers (17, 48, 105, 106) have been studied in many forms and applications, for instance rendezvous (18), leader following (107), attitude control (108) and many others (109, 110, 111). The majority of these algorithms, dealing with decentralized motion coordination problems, assume that the agents have access to absolute position information (GPS) and thus have a common global reference frame that makes it easy to interpret the information passed by other agents. Even when in multi agent systems the agents are not supposed to know their absolute position, many times they are assumed to have a common attitude reference to exchange information that can be achieved by using a compass, gyroscopes and occasionally gravity as common reference for their coordinate system. For space applications another technological solution is to use a frame of fixed stars to have a common reference. In all these instances several technological countermeasures have to be undertaken for the implementation of coordination algorithms increasing the total costs of the single agents. We believe that removing such hidden assumption could significantly advance the technological feasibility of mobile swarm of agents, reducing their dependence on the global

positioning system in the low level control loops. Furthermore in many space applications, where networks of mobile robots are envisioned in the so not distant future, the absence of the need for absolute position information or a common coordinate system could prove to be an essential robust feature.

On the other hand, many works on motion coordination for multi-agent systems that do not assume to have neither a common reference frame nor absolute positions suffer greatly from the very limited amount of information that can be retrieved locally, resulting in limited capabilities of the coordination algorithms.

Indeed, the availability of a common reference frame turns out to be a critical issue for several applications. For instance, it could help to significantly simplify any task involving the formation control of a team of robots, or the tracking of an object. Generally speaking, this can be thought as a service for multi-agent systems or sensor networks by which any constraint on the absence of a common reference frame could be relaxed, hence simpler or more effective algorithms could be developed.

In this chapter a novel approach to the problem of decentralized agreement toward a common point in space in a multi-agent system is proposed. The proposed approach allows to perform an agreement on the network centroid, on a common reference frame and therefore on a common heading. Using this information a global positioning system for the agents using only local measurements can be achieved. Furthermore only point-to-point asynchronous communications between neighboring agents are allowed thus achieving robustness against random communication failures. The proposed algorithms can be thought as general tools to locally retrieve global information usually not available to the agents. In this way, any assumption on the absence of a common reference frame could be relaxed and therefore, simpler algorithms could be developed.

Background on Gossip algorithms over networks

Let the network of agents be described by a time-varying graph $\mathcal{G}(t) = (V, E(t))$, where $V = \{v_i; i = 1, \dots, n\}$ is the set of nodes (agents) and $E(t) = \{e_{ij} = (v_i, v_j)\}$ is the set of edges (connectivity) representing the point-to-point communication channel availability at time t . A position $p_i \in \mathbb{R}^d$ in the d^{th} dimensional space is associated to each node $v_i \in V$, with $i = 1, \dots, n$. In particular, an edge

representing a connection between two agents exists if and only if the distance between these agents is less than or equal to their sensing radius r , namely

$$E(t) = \{e_{ij} : \|p_i(t) - p_j(t)\| \leq r, i \neq j\},$$

where $\|\cdot\|_d$ is the Euclidean norm in \mathbb{R}^d . Therefore, a generic couple of agents $\{i, j\}$ is able to sense $\|p_i - p_j\|$ reciprocally. In addition, each agent has a local reference frame defined by an orthonormal basis of vectors in \mathbb{R}^d fixed on it and, is able to determine the direction in which neighbors are sensed, strictly with respect to its own local reference frame.

In the proposed framework a gossip algorithm is defined as a triplet $\{\mathcal{S}, \mathcal{R}, \mathbf{e}\}$ where

- $\mathcal{S} = \{s_1, \dots, s_n\}$ is a set containing the local estimate s_i of each agent i in the network.
- \mathcal{R} is a local interaction rule that given edge $e_{i,j}$ and the states of agents i, j $\mathcal{R} : (s_i, s_j) \Rightarrow (\hat{s}_i, \hat{s}_j)$.
- \mathbf{e} is a edge selection process that specifies which edge $e_{ij} \in E(t)$ is selected at time t .

From an algorithmic point of view, a possible implementation of the gossip algorithm described above is given in Algorithm ??.

Definition 6.1.1 *Let us define $\mathbb{G}(t, t + \Delta t) = \{V, \mathbb{E}(t, t + \Delta t)\}$, where $\mathbb{E}(t, t + \Delta t) = \bigcup_{k=t}^{t+\Delta t} \mathbf{e}(k)$, as the graph resulting from the union of all the edges given by the edge selection process from time t to $t + \Delta t$.*

6.2 Problem description

Let us consider a network of agents with limited sensing capabilities. Each agent, which is characterized by a position in a 2-dimensional space, is able to cooperate with its neighboring agents, i.e., agents that are within its sensing radius. The following assumptions on the network of agents are made:

Algorithm 1: Gossip Algorithm

Data: $t = 0, s_i(0) = s_{i0} \quad \forall i = 1, \dots, n.$

Result: $s_i(t_{stop}) \quad \forall i = 1, \dots, n.$

while *stop_condition* **do**

- Let $t = t + 1.$;
- Select an edge $e_{ij} \in E(t)$ according to $\mathbf{e}.$;
- Update the states of the selected agents applying \mathcal{R} :

$$(s_i(t + 1), s_j(t + 1)) = \mathcal{R}(s_i(t), s_j(t)).$$

end

Assumptions 1

- *The network can be described by a connected undirected switching graph.*
- *Sensing range is limited by a maximum sensing radius $r.$*
- *Communications are asynchronous.*
- *Each node can sense the distance between itself and its neighbors.*
- *Each node can sense the direction in which it sees its neighbors with respect to its local reference frame, arbitrary fixed on it.*

■

Note that, for each agent i it is possible to express its estimate s_i with respect to a global reference frame by introducing a rotation matrix R_i as follows:

$$s_{gi} = R_i s_i + p_i. \tag{6.1}$$

Our first objective is to make the local estimate of each agent converge to a common value by applying an iterative algorithm so that:

$$\forall i, \quad \lim_{t \rightarrow \infty} s_{gi}(t) = R_i s_i(t) + p_i = \frac{1}{n} \sum_{i=1}^n (R_i s_i(0) + p_i).$$

6.3 Agreement on a common point in a 2-D space

In this section, we specify an interaction rule \mathcal{R} such that an agreement on a common point under assumptions 1 is achieved.

Given a couple of nodes $\{i, j\}$ for which an edge exists, that is $e_{ij} \in E(t)$, let us define the direction for which node i is able to sense node j with respect to its local frame as

$$\hat{c}_{ij} = R_i^T \frac{(p_j - p_i)}{\|p_j - p_i\|},$$

where $p_i, p_j \in \mathbb{R}^2$. Clearly, the following property holds $R_i c_{ij} = -R_j c_{ji}$. Furthermore we define the orthogonal versor \hat{c}_{ij}^\perp so that a right handed frame is built as follows:

$$\begin{bmatrix} R_i \hat{c}_{ij}^\perp \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \wedge \begin{bmatrix} R_i \hat{c}_{ij} \\ 0 \end{bmatrix}$$

In addition, let the relative distance between two nodes i and j be:

$$d_{ij} = d_{ji} = \|p_i - p_j\|_2.$$

Finally, let the network of agents be deployed in a 2-dimensional space. The proposed algorithm consists of an edge selection process \mathfrak{e} that specifies which edge $e_{ij} \in E(t)$ is active at time t and a local interaction rule \mathcal{R} that specifies how to update the estimates of agents v_i and v_j .

Now follows the definition of \mathcal{S} and \mathcal{R} :

Definition 6.3.1 (\mathcal{S})

Let $\mathcal{S} = \{s_1, s_2, \dots, s_n\}$, with $s_i \in \mathbb{R}^2, \forall i = 1, \dots, n$ be the set of current agents local estimates, each one in their own reference frame. ■

Definition 6.3.2 (\mathcal{R})

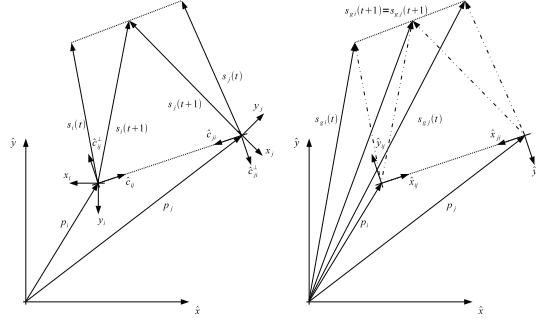


Figure 6.1: Example of algorithm iteration involving two nodes. a) With respect to the agents local reference frames. b) With respect to a global reference frame.

- Let

$$\begin{aligned} \Delta(t) &= \frac{d_{ij} - s_j(t)^T \hat{c}_{ji} + s_i(t)^T \hat{c}_{ij}}{2}, \\ \Delta^\perp(t) &= \frac{s_i(t)^T \hat{c}_{ij}^\perp - s_j(t)^T \hat{c}_{ji}^\perp}{2}, \end{aligned} \quad (6.2)$$

- \mathcal{R} :

$$s_i(t+1) = \Delta(t) \cdot \hat{c}_{ij} + \Delta^\perp(t) \cdot \hat{c}_{ij}^\perp, \quad (6.3)$$

■

As a support for the algorithm description, Fig. 6.2 depicts a possible scenario involving two nodes, namely i and j .

A couple of remarks are now in order:

- This update rule leads itself to an easy decentralized implementation of the algorithm,
- All the parameters are local to the agents and independent to any specific reference frame as they rely on a common direction given by the line of sight between the two agents.
- From an implementation point of view, each quantity such as $s_j(t)^T \hat{c}_{ji}$ can be locally computed by each agent in its own coordinates frame.
- The proposed gossip algorithm allows to converge to a common point in a 2-dimensional space. As it will be shown, the convergence to the centroid of the network is simply a consequence of the particular choice of the initial

conditions.

Lemma 6.3.3 *The gossip algorithm $\{\mathcal{S}, \mathcal{R}, \mathbf{e}\}$, with \mathcal{S}, \mathcal{R} defined respectively as in (6.3.1), (6.3.2) can be equivalently stated with respect to a global common reference frame as follows:*

$$\begin{aligned} x(t+1) &= W(\mathbf{e}(t))x(t), \\ y(t+1) &= W(\mathbf{e}(t))y(t). \end{aligned}$$

where $W(\mathbf{e}(t))$ is a matrix representation of the update rule \mathcal{R} , $x(t) = [x_1(t), \dots, x_n(t)]^T \in \mathbb{R}^n$, $y(t) = [y_1(t), \dots, y_n(t)]^T \in \mathbb{R}^n$, and $s_{gi}(t) = [x_i(t) \ y_i(t)]^T$.

Proof: Let us consider a generic update for a couple of agents $\{i, j\}$. Given the estimates $(s_i(t), s_j(t))$ at time t , according to the rule \mathcal{R} given in (6.3.2) the estimates at time $t+1$ would be:

$$s_i(t+1) = \Delta(t) \cdot \hat{c}_{ij} + \Delta^\perp(t) \cdot \hat{c}_{ij}^\perp,$$

Now by substituting the Δ according to the definition given in (6.2) we have:

$$\begin{aligned} s_i(t+1) &= \frac{1}{2} (d_{ij} - s_j(t)^T \hat{c}_{ji} + s_i(t)^T \hat{c}_{ij}) \cdot \hat{c}_{ij} + \\ &\quad + \frac{1}{2} (s_i(t)^T \hat{c}_{ij}^\perp - s_j(t)^T \hat{c}_{ji}^\perp) \cdot \hat{c}_{ij}^\perp, \end{aligned}$$

At this point, let us consider the update of the agent i with respect to a global frame as given in (6.1):

$$\begin{aligned}
s_{gi}(t+1) &= R_i \frac{1}{2} (d_{ij} - s_j(t)^T \hat{c}_{ji} + s_i(t)^T \hat{c}_{ij}) \cdot \hat{c}_{ij} \\
&\quad + \frac{1}{2} R_i (s_i(t)^T \hat{c}_{ij}^\perp - s_j(t)^T \hat{c}_{ji}^\perp) \cdot \hat{c}_{ij}^\perp + p_i \\
&= \frac{1}{2} (d_{ij} - s_j(t)^T \hat{c}_{ji} + s_i(t)^T \hat{c}_{ij}) \cdot \hat{x}_{ij} + \\
&\quad + \frac{1}{2} (s_i(t)^T \hat{c}_{ij}^\perp - s_j(t)^T \hat{c}_{ji}^\perp) \cdot \hat{y}_{ij} + p_i
\end{aligned}$$

where $[\hat{x}_{ij}, \hat{y}_{ij}]^T$ are the equivalent of $[\hat{c}_{ij}, \hat{c}_{ij}^\perp]^T$ in a common global reference frame. At this point, with respect to the local reference frame of agent i we have that:

$$\begin{aligned}
x_{s_i} &= s_i(t)^T \hat{c} & y_{s_i} &= s_i(t)^T \hat{c}^\perp \\
x_{s_j} &= d_{ij} - s_j(t)^T \hat{c}_{ji} & y_{s_j} &= -s_j(t)^T \hat{c}_{ji}^\perp
\end{aligned}$$

which allows to re-write the previous equation as follows:

$$\begin{aligned}
s_{gi}(t+1) &= \left(\frac{x_{s_i} + x_{s_j}}{2} \right) \cdot \hat{x}_{ij} + \left(\frac{y_{s_i} + y_{s_j}}{2} \right) \cdot \hat{y}_{ij} + p_i \\
&= \frac{x_{s_i} \hat{x}_{ij} + y_{s_i} \hat{y}_{ij} + p_i}{2} + \frac{x_{s_j} \hat{x}_{ij} + y_{s_j} \hat{y}_{ij} + p_i}{2} \\
&= \frac{s_{gi}(t) + s_{gj}(t)}{2}
\end{aligned}$$

Therefore according to the updating rule \mathcal{R} given in (6.3.2) with respect to a global reference frame we have:

$$\begin{aligned}
s_{gi}(t+1) &= \frac{s_{gi}(t) + s_{gj}(t)}{2} \\
s_{gj}(t+1) &= \frac{s_{gi}(t) + s_{gj}(t)}{2}
\end{aligned}$$

Hence, we can decouple the coordinate system and study the evolution of the states in the two different axes separately:

$$\begin{aligned}
x(t+1) &= W(\mathbf{e}(t))x(t), \\
y(t+1) &= W(\mathbf{e}(t))y(t).
\end{aligned}$$

where, if at time t edge $e_{ij} = (i, j)$ is selected, we have:

$$W(e_{ij}) = I - \frac{(\hat{e}_i - \hat{e}_j)(\hat{e}_i - \hat{e}_j)^T}{2}$$

where $\hat{e}_i = [0 \dots 0 \underbrace{1}_i 0 \dots 0]^T$ is a $n \times 1$ vector with all the components equal to 0 but the i -th component equal to 1. \square

Now, let us define the set of fixed points $C(e_{ij}) = \text{Fix } W(e_{ij}) = \{x \in \mathbb{R}^n : W(e_{ij})x = x\}$ and let $\hat{C}_{(t,t+\Delta t)} = \bigcap_{e_{ij} \in \mathbb{E}(t,t+\Delta t)} C(e_{ij})$ be the intersection of this set of fixed points over time $[t, t + \Delta t]$. Now, let us define the quasi projection of x_0 onto C as $\mathcal{Q}_c x_0 = \{x \in C : \|x - c\| \leq \|x_0 - c\|, \forall c \in C\}$ (60).

The following Lemma states that if the graph representing the union of the selected edges is connected over a window of time, then the space representing the intersection of the images of the matrices corresponding to those edges is $\text{span}\{\mathbf{1}_n\}$.

Lemma 6.3.4 *If ϵ is such that $\forall t, \exists \Delta t : \mathbb{G}(t, t + \Delta t)$ is connected, then:*

$$\hat{C}_{(t,t+\Delta t)} = \bigcap_{e_{ij} \in \mathbb{E}(t,t+\Delta t)} C(e_{ij}) = \text{span}\{\mathbf{1}_n\} \quad (6.4)$$

where $\mathbf{1}_n = [1, \dots, 1]^T$ is a $n \times 1$ unit vector with all the components equal to 1.

Proof: For any given edge e_{ij} the related matrix $W(e_{ij})$ is an orthogonal projection matrix, i.e., W is idempotent and symmetric. Moreover:

$$C(e_{ij}) = \text{Ran}(W(e_{ij})) = \text{Ker}(I - W(e_{ij}))$$

Let us recall that for any orthogonal projection matrix $W(e_{ij})$ the following property holds ():

$$\left[\bigcap \text{Ran}(W(e_{ij})) \right]^\perp = \bigcup \text{Ker}(W(e_{ij}))$$

Now, let us define the structure of the kernel for a generic matrix $W(e_{ij})$ as follows:

$$\text{Ker}(W(e_{ij})) = \text{span}\left\{ [0, \dots, \underbrace{1}_i, \dots, 0, \dots, \underbrace{-1}_j, \dots, 0]^T \right\}$$

Note that the vector $b_{ij} = [0, \dots, \underbrace{1}_i, \dots, 0, \dots, \underbrace{-1}_j, \dots, 0]^T$ can be thought as the element of the incidence matrix \mathcal{J} (42) for the graph $\mathbb{G}(t, t + \Delta t)$ which describes the link between agent i and agent j . At this point, by exploiting this analogy along with results coming from the graph theory we know that:

$$\text{rank}(\mathcal{J}) = n - c$$

where n is the number of vertices and c the number of connected components. Hence, if the graph $\mathbb{G}(t, t + \Delta t)$ is connected we will have that:

$$\text{rank}(\mathcal{J}) = n - 1.$$

Therefore:

$$\text{rank}(\hat{C}_{t,t+\Delta t}) = \text{rank}(\mathcal{J})^\perp = 1.$$

Moreover, by noticing that \mathcal{J} is by construction a row-sum matrix, the following holds:

$$\hat{C}_{(t,t+\Delta t)} = \text{Ran}(\mathcal{J})^\perp = \text{span}\{\mathbf{1}_n\}.$$

□

To link the connectivity of the graph representing the union of the selected edges to the contractive property respect to $\text{span}\{\mathbf{1}_n\}$ of the product of the para-contracting matrices $W(e_{ij})$, the following lemma is needed:

Lemma 6.3.5 *If ϵ is such that $\forall t, \exists \Delta t : \mathbb{G}(t, t + \Delta t)$ is connected, then there exists a norm such that:*

$$\begin{aligned} \|W(e_{ij})x - c\| &\leq \|x - c\|, \\ \forall c \in \hat{C}_{(t,t+\Delta t)}, \quad \forall e_{ij} \in \mathbb{E}_{(t,t+\Delta t)}, \quad \forall x \in \mathbb{R}^n \end{aligned} \tag{6.5}$$

$$\begin{aligned} \|\Phi_{(t,t+\Delta t)}x - c\| &< \|x - c\|, \\ \forall c \in \hat{C}_{(t,t+\Delta t)}, \quad \forall x \in \mathbb{R}^n \setminus \hat{C}_{(t,t+\Delta t)} \end{aligned} \tag{6.6}$$

where $\Phi_{(t,t+\Delta t)} = \prod_{e_{ij} \in \mathbb{E}(t,t+\Delta t)} W(e_{ij})$.

Proof: For any given edge e_{ij} the related matrix $W(e_{ij})$ is an orthogonal projection matrix, i.e., W is idempotent and symmetric. Moreover, the following holds:

$$\|W(e_{ij})x\| \leq \|x\|, \quad \forall x \in \mathbb{R}^n.$$

In our case, both the euclidian norm $\|\cdot\|$ and the infinity norm $\|\cdot\|_\infty$ are suitable. Using the euclidian norm and the fact that $c = \gamma \mathbf{1}$ with $\gamma \in \mathbb{R}$, we notice that:

$$\|x - c\|^2 = \|x\|^2 + \|c\|^2 - 2c^T x$$

Therefore, the inequality (6.5) can be rewritten as follows:

$$\begin{aligned} \|W(e_{ij})x - c\| &\leq \|x - c\|, \\ \|W(e_{ij})x\|^2 + \|c\|^2 - 2c^T W(e_{ij})x &\leq \|x\|^2 + \|c\|^2 - 2c^T x \end{aligned}$$

At this point by noticing that any matrix $W(e_{ij})$ is a row-sum matrix the following holds:

$$\mathbf{1}_n^T W(e_{ij}) = \mathbf{1}_n^T \quad W(e_{ij})\mathbf{1}_n = \mathbf{1}_n$$

Now, by recalling that for any $c \in \hat{C}_{t,t+\Delta t}$ we have $c = \gamma \cdot \mathbf{1}_n$, the following holds:

$$c^T W(e_{ij}) = \gamma \cdot \mathbf{1}_n^T W(e_{ij}) = \gamma \cdot \mathbf{1}_n^T = c^T$$

An therefore the previous inequality can be rewritten as follows:

$$\begin{aligned} \|W(e_{ij})x\|^2 - 2c^T x &\leq \|x\|^2 - 2c^T x \\ \|W(e_{ij})x\|^2 &\leq \|x\|^2 \end{aligned}$$

which by construction is always verified $\forall c \in \hat{C}_{(t,t+\Delta t)}$, and $\forall e_{ij} \in \mathbb{E}_{(t,t+\Delta t)}$.

A similar argument holds for inequality (6.9). In particular, in this case it should be noticed that for any matrix $W(e_{ij})$ the following holds:

$$\|W(e_{ij})x\| < \|x\|, \quad \forall x \notin C(e_{ij}).$$

Therefore, if we consider the product $\Phi_{(t,t+\Delta t)} = \prod_{e_{ij} \in \mathbb{E}(t,t+\Delta t)} W(e_{ij})$, we have

$$\|\Phi_{(t,t+\Delta t)}x - c\| < \|x - c\|, \quad \forall c \in \hat{C}_{(t,t+\Delta t)}, \quad \forall x \notin \hat{C}_{(t,t+\Delta t)}$$

as since $\mathbb{G}(t, t + \Delta t)$ is connected, $\forall x \notin \hat{C}_{(t,t+\Delta t)}$ there will always be at least an edge e_{ij} so that $x \notin C(e_{ij})$ and therefore

$$\|W(e_{ij})x\| < \|x\|.$$

□

In the above Lemma 6.3.5 it is shown that the agents estimates eventually contract toward $\text{span}\{\mathbf{1}_n\}$. In the following Lemma it is shown that the trajectories of the system actually converge to some point in $\text{span}\{\mathbf{1}_n\}$.

Lemma 6.3.6 *If ϵ is such that $\forall t, \exists \Delta t : \mathbb{G}(t, t + \Delta t)$ is connected, then for any sequence of intervals $\{l_i\}$ with $l_i = l_{i-1} + \Delta t_i$ and $l_j > l_i \forall j > i$, it holds:*

$$d(x(l_i), \text{span}\{\mathbf{1}_n\}) \rightarrow 0. \tag{6.7}$$

Proof: The proof is a simple consequence of the results given in Lemma (6.3.4) and Lemma (6.3.5). In particular, by exploiting Lemma (6.3.4) we have that for any sequence of intervals $\{l_i\}$:

$$\hat{C}_{(l_0, l_1)} = \hat{C}_{(l_1, l_2)} = \dots = \hat{C}_{(l_{i-1}, l_i)} = \text{span}\{\mathbf{1}_n\}.$$

Moreover, by exploiting Lemma (6.3.5) we have that for any sequence of intervals $\{l_i\}$:

$$\begin{aligned} \|\Phi_{(l_{i-1}, l_i)}x(l_{i-1}) - c\| &< \|x(l_{i-1}) - c\| \\ &\vdots \\ \|\Phi_{(l_0, l_1)}x(l_0) - c\| &< \|x(l_0) - c\| \end{aligned}$$

Therefore:

$$d(x(l_i), \text{span}\{\mathbf{1}_n\}) \rightarrow 0 \quad \forall c \in \text{span}\{\mathbf{1}_n\}$$

□

Theorem 6.3.7 *Let us consider a gossip algorithm $\{\mathcal{S}, \mathcal{R}, \mathbf{e}\}$, with \mathcal{S}, \mathcal{R} defined respectively as in Definition (6.3.1), and Definition (6.3.2). If \mathbf{e} is such that $\forall t, \exists \Delta t : \mathbb{G}(t, t + \Delta t)$ is connected, then:*

$$\lim_{t \rightarrow \infty} s_{gi}(t) = R_i s_i(t) + p_i = \frac{1}{n} \sum_{i=1}^n (R_i s_i(0) + p_i), \quad (6.8)$$

$$\forall i = 1, \dots, n.$$

Proof: By following Lemma (6.3.3), the proposed gossip algorithm can be rewritten in a common global reference frame. Moreover, the state evolution can be investigated independently for each axis as follows:

$$\begin{aligned} x(t+1) &= W(\mathbf{e}(t))x(t), \\ y(t+1) &= W(\mathbf{e}(t))y(t). \end{aligned}$$

Let us focus only on the $x(t)$ axis as the same holds for the $y(t)$ axis. Now, due to Lemma (6.3.4) we know that for any given interval $[t, t + \Delta t]$:

$$\hat{C}_{(t, t + \Delta t)} = \bigcap_{e_{ij} \in \mathbb{E}(t, t + \Delta t)} C(e_{ij}) = \text{span}\{\mathbf{1}_n\}$$

In addition, due to Lemma (6.3.5) we know that for any given interval $[t, t + \Delta t]$ such that $\mathcal{G}(t + \Delta t)$ is connected the following holds:

$$\begin{aligned} \|\Phi_{(t, t + \Delta t)} x - c\| &< \|x - c\|, \\ \forall c \in \hat{C}_{(t, t + \Delta t)}, \quad \forall x \in \mathbb{R}^n \setminus \hat{C}_{(t, t + \Delta t)} \end{aligned}$$

Finally, due to Lemma (6.3.6) we know that exists a sequence of intervals l_i so that:

$$d(x(l_i), \text{span}\{\mathbf{1}_n\}) \rightarrow 0.$$

Therefore, the sequence $\{x(l_i)\}$ converges in norm to some points in $\text{span}\{\mathbf{1}_n\}$, that is

$$\|x(l_i) - c\| \rightarrow 0 \quad \text{then} \quad \{x(l_i)\} \rightarrow c, \quad c \in \text{span}\{\mathbf{1}_n\}.$$

In addition, each single matrix $W(e_{ij})$ is a symmetric row-sum matrix:

$$\mathbf{1}_n^T W(e_{ij}) = \mathbf{1}_n^T \quad W(e_{ij}) \mathbf{1}_n = \mathbf{1}_n.$$

Therefore, the sum of the vector components must be preserved over time at each iteration. This implies that for a given $c = \gamma \mathbf{1}_n$:

$$\begin{aligned} \sum_{i=1}^n c_i &= \sum_{i=1}^n x_i(l_0) \\ n\gamma &= \sum_{i=1}^n x_i(l_0) \\ \gamma &= \frac{\sum_{i=1}^n x_i(l_0)}{n}. \end{aligned}$$

From this it follows that:

$$x(l_i) \rightarrow \frac{\sum_{i=1}^n x_i(l_0)}{n} \mathbf{1}_n.$$

In the same way, for the $y_g(t)$ we have:

$$y(l_i) \rightarrow \frac{\sum_{i=1}^n y_i(l_0)}{n} \mathbf{1}_n.$$

Therefore, for each agent i we have:

$$s_{gi}(t) \rightarrow \left[\begin{array}{c} \frac{\sum_{i=1}^n x_i(l_0)}{n} \\ \frac{\sum_{i=1}^n y_i(l_0)}{n} \end{array} \right] = \frac{1}{n} \sum_{i=1}^n (R_i s_i(0) + p_i)$$

which proves the statement. □

Corollary 6.3.8 *Let us consider the gossip algorithm defined by $\{\mathcal{S}, \mathcal{R}, \mathbf{e}\}$ as in Theorem 6.3.7. If each agent initializes its state $s_i(0) = 0$ to zero, then all the agents estimates converge to the network centroid:*

$$\lim_{t \rightarrow \infty} s_{gi}(t) = R_i s_i(t) + p_i = \frac{1}{n} \sum_{i=1}^n p_i, \quad \forall i = 1, \dots, n. \quad (6.9)$$

Proof: The proof follows from Theorem 6.3.7 if each agent i has $s_i(0) = 0$. \square

Algorithm 2: Reference Frame Agreement Algorithm

Data: $\mathcal{F}_i = \{f_{1,i}, f_{2,i}\}$

Result: A_i^r

- Compute the versors $r_{x,i}$ and $r_{y,i}$:

$$r_{x,i} = \frac{(f_{2,i} - f_{1,i})}{\|f_{2,i} - f_{1,i}\|} \quad r_{y,i} = r_{x,i}^\perp$$

- Compute the translation vector t_i :

$$t_i = \|f_{1,i} - p_i\|,$$

- Compute the homogeneous transformation matrix A_i^r :

$$A_i^r = \begin{bmatrix} R_i^r & t_i \\ 0 & 1 \end{bmatrix}$$

6.4 Agreement on a common reference frame in a 2-D space

In Section 6.3 an algorithm for the agreement on a common point in a 2-D space has been described. In this section, this result is used to build an algorithm to

reach an agreement on a common reference frame in a 2-D space. To this end, by exploiting Algorithm ?? according to Theorem 6.3.7, the network of agents first achieve an agreement on a set of two common points whose representation is obviously local to the agent reference frame, i.e., $\mathcal{F}_i = \{f_{1,i}, f_{2,i}\}$. Then by using Algorithm ??, each agent builds a rotation matrix A_i^r with respect to a common reference frame defined according to \mathcal{F}_i . In order to do that, the two points $\mathcal{F}_i = \{f_{1,i}, f_{2,i}\}$ can be enumerated according to the temporal order in which they have been computed. Moreover, we may have the first point $f_{1,i}$ identify the origin of the frame $O_r = f_{1,i}$ and use the second point $f_{2,i}$ to compute a common x versor, while the common y versor can be chosen to achieve a right-handed orthogonal frame.

6.5 Agreement on a common point in d-D space

Let us now consider a d-D dimensional case with $d > 2$. Each agent, which is characterized by a position in a d -dimensional space, is able to collaborate with the neighboring agents, i.e., agents that are within its range of sensing. Information exchanged between agents is only the distance between them and the direction of the line of sight with respect to their local reference frame.

In this section, we specify an interaction rule \mathcal{R} such that an agreement on a common point under assumptions 1 can be achieved in a d -dimensional space. We make use of notation of the 2-d case, except that vectors are now d -dimensional.

The new definition for the local interaction rule is:

Definition 6.5.1 (\mathcal{R})

- *Let*

$$\Delta(t) = \left(\frac{d_{ij} + s_j(t)^T c_{ji} - s_i(t)^T c_{ij}}{2} + s_i(t)^T c_{ij} \right), \quad (6.10)$$

- \mathcal{R} :

$$s_i(t+1) = s_i(t) + \Delta(t) \cdot c_{ij}, \quad (6.11)$$

■

Some remarks are now in order:

- All the parameters are local to the agents and independent to any specific reference frame as they rely on a common direction given by the line of sight between the two agents, such computation is not affected by a d -dimensional space.
- From an implementation point of view, each quantity such as $s_j(t)^T \hat{c}_{ji}$ can be locally computed by each agent in its own coordinates frame.
- The proposed local interaction rule differs from the 2-d case in that only the projections of the estimates along the line of sight of the agents are averaged, not the whole estimate.

It will now be proved that, if the graph representing the network is fully connected, and each edge has a strictly positive probability to be chosen at each instant of time, then the probability that all the agents agree on where is the network centroid goes to one as time goes to infinity.

Theorem 6.5.2 *If the network of agents is fully connected then*

$$\forall i \in V, Pr \left(\lim_{t \rightarrow \infty} s_{gi}(t) = \frac{\sum_{i=1}^n p_i}{n} \right) = 1$$

Proof:

Given a suitable Lyapunov-like function of the state of the network, the proof relies on a probabilistic argument to show that such a function converges to zero as time goes to infinity almost surely, depending on the edge selection process.

By considering two generic agents i and j , the generic configuration at time t given in Figure 6.2 is used as support for the proof.

Let O_g be the coordinates of the network centroid in the common global reference frame. Then if the estimates of the agents are updated using Algorithm ?? and we choose

$$V(t) = \sum_{i=1}^n \|s_{gi}(t) - O_g\|_2^2$$

as a candidate Lyapunov function. $V(t)$ is a quadratic function and so is positive for any $t \geq 0$. The following manipulations show that $V(t+1) \leq V(t)$.

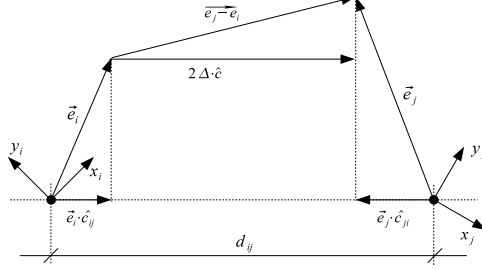


Figure 6.2: Example of algorithm iteration involving two nodes.

At time t only nodes i, j change their estimation, thus possibly changing the value of $V(t+1)$. Thus we have

$$\begin{aligned} V(t+1) - V(t) &= \|s_{gi}(t+1) - O_g\|^2 + \|s_{gj}(t+1) - O_g\|^2 \\ &\quad - \|s_{gi}(t) - O_g\|^2 - \|s_{gj}(t) - O_g\|^2 \end{aligned}$$

Since the global reference frame is arbitrary we chose $O_g = \mathbf{0}$ for sake of clarity. Thus

$$\begin{aligned} V(t+1) - V(t) &= \\ &= \|s_{gi}(t+1)\|^2 + \|s_{gj}(t+1)\|^2 - \|s_{gi}(t)\|^2 - \|s_{gj}(t)\|^2 \end{aligned}$$

Now from the description given in Algorithm ??, we know that

$$s_i(t+1) = s_i(t) + \Delta(t) \cdot c_{ij},$$

$$s_j(t+1) = s_j(t) + \Delta(t) \cdot c_{ji}.$$

where:

$$\Delta(t) = \left(\frac{d_{ij} + s_j(t)^T c_{ji} - s_i(t)^T c_{ij}}{2} + s_i(t)^T c_{ij} \right),$$

by definition $c_{ij} = -c_{ji} = \hat{c}$ with $\hat{c}^T \hat{c} = 1$, the previous equation can be re-written as

$$\begin{aligned} V(t+1) - V(t) &= \|s_{gi} + \Delta \hat{c}\|^2 + \|s_{gj} - \Delta \hat{c}\|^2 \\ &\quad - \|s_{gi}\|^2 - \|s_{gj}\|^2 \\ &= s_{gi}^T s_{gi} + 2\Delta \hat{c}^T s_{gi} + \Delta^2 + s_{gj}^T s_{gj} \\ &\quad - 2\Delta \hat{c}^T s_{gj} + \Delta^2 - s_{gi}^T s_{gi} - s_{gj}^T s_{gj} \\ &= 2\Delta^2 + 2\Delta \hat{c}^T (s_{gi} - s_{gj}) \end{aligned}$$

where the temporal index has been omitted for sake of clarity.

Now, by observing the Fig. 6.2 can be noticed that Δ is nothing more than the projection over \hat{c} at time t of the vector $s_{gj} - s_{gi}$ scaled by a factor of two,

$$\Delta = \|s_{gj} - s_{gi}\| \frac{\cos(\alpha)}{2}$$

where α is the angle between s_{gj} and s_{gi} . Now, by substituting for Δ :

$$\begin{aligned} V(t+1) - V(t) &= \\ &= 2\Delta^2 + 2\Delta \hat{c}^T (s_{gi} - s_{gj}) \\ &= \frac{\|s_{gj} - s_{gi}\|^2 \cos(\alpha)^2}{2} - \|s_{gj} - s_{gi}\|^2 \cos(\alpha)^2 \leq 0 \end{aligned}$$

where

$$\begin{aligned} \hat{c}^T (s_{gi} - s_{gj}) &= -\hat{c}^T (s_{gj} - s_{gi}) \\ &= -\|s_{gi} - s_{gj}\| \cos(\alpha). \end{aligned}$$

Thus proving that $V(t)$ is a non-increasing function of time. $V(t+1) = V(t)$ each time an edge connecting two nodes who have exactly the same estimate of the network centroid are chosen.

If $V(t) \neq 0$ necessarily there exist at least two nodes in the network such that $s_{gi} \neq s_{gj} \neq O_g$. Since we assume that the network is fully connected, i.e. each node has the possibility to communicate with any other node, and given that each communication link is assumed to have a strictly positive probability to be activated at each instant of time, then we have that the probability

$$Pr \left(\lim_{t \rightarrow \infty} V(t'+t) - V(t') < 0 \right) = 1$$

since we are sampling a finite set of elements an infinite amounts of times.

Since for $V(0)$ we have that

$$\forall i, s_{gi} = \frac{\sum_{i=1}^n p_i}{n},$$

then

$$\forall i \in V, Pr \left(\lim_{t \rightarrow \infty} s_{gi}(t) = \frac{\sum_{i=1}^n p_i}{n} \right) = 1,$$

thus proving the statement. ■

6.6 Simulations

In order to corroborate the mathematical analysis, several simulations have been performed by exploiting a framework developed by the authors using Matlab. Different network topologies such as fully connected, arbitrary or tetrahedrons-based have been investigated. The following edge selection process has been exploited for the simulations: at each time step, an agent is selected randomly using a uniform probability distribution, it then communicates with all its neighbors in random order. Such a selection process has been chosen to mimic the expected behavior of the algorithm in a real application where communications are sequential and asynchronous.

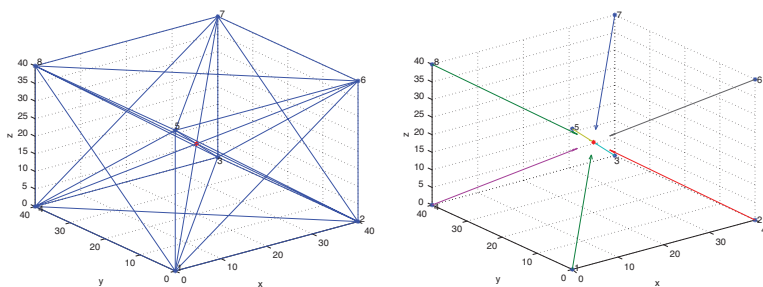


Figure 6.3: Fully connected graph with 8 nodes.

Fully connected Graph

In the case of a fully connected graph the algorithms always converges, as proven in Theorem 6.5.2. In Fig. 6.3, a configuration where a network is deployed according to the vertexes of a cube (8 agents) is shown. In particular, each node converges to the right estimate of the network centroid after only few iterations. Note that, this particular embedding was chosen only for sake of visualization and it does not affect the convergence of the algorithm.

As far as the convergence time is concerned, several simulations considering a varying number of vertexes ranging from 10 nodes to 100 nodes have been performed. Results are given in Table 6.1. In detail, 10 different configurations were considered, each one was run 50 times and at each single iteration a deployment was randomly generated. Table 6.1 shows the average number of iteration re-

quired to the algorithm in order to converge for each configuration. Although the number of iterations increases with the number of vertexes, it is important to recall the inherent parallelism of the algorithm previously discussed which is not revealed by this table. Indeed, at each iteration in a real context there might be several couples of nodes performing the algorithm at the same time, while the code running in Matlab is sequential.

Table 6.1: Convergence Rate - Fully Connected Graph

Number of Vertexes	Number of Iterations
10	57
20	65
30	77
40	82
50	90
60	98
70	106
80	116
90	127
100	136

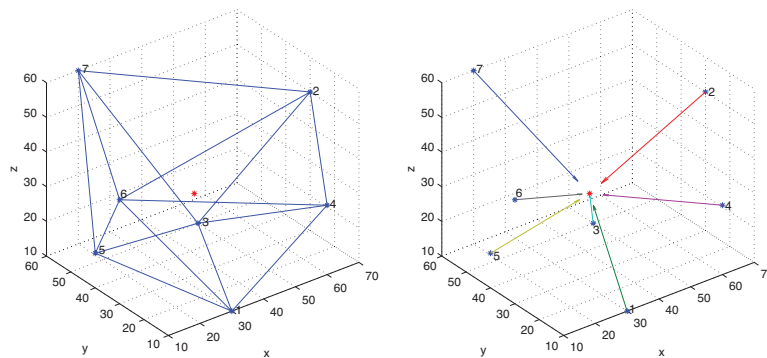


Figure 6.4: Arbitrary graph with 7 nodes.

Arbitrary Connected Graphs

A general necessary condition for the convergence of Algorithm ?? for arbitrary connected graphs is now given.

Theorem 6.6.1 *If a network of agents with $p_i \in \mathbb{R}^d$ executes Algorithm ??, a necessary condition for the agents to have the same common estimate of the network centroid is that each agent has at least d neighbors.*

Proof: The agents perform estimation updates along the line of sight between them, at each iteration they can adjust their estimate along only one direction. Thus to be able to adjust their estimate in a \mathbb{R}^d space they need at least d independent directions over which perform their update. This condition is not sufficient, a counter-example in Fig.6.6 is provided. ■

A reason why this happens is that the algorithm involves a projection of the agents' estimate along the line of sight with their neighbors, this may fail to propagate enough information about the agent's estimate if the graph is not sufficiently connected. On the other hand Theorem 6.5.2 proves that $V(t) = \sum_{i=1}^n \|s_{gi}(t) - O_g\|_2^2$ is a non-increasing function for any arbitrary graph thus the execution of the proposed local interaction rule may only either improve or leave intact the current error on the estimation of the common reference point.

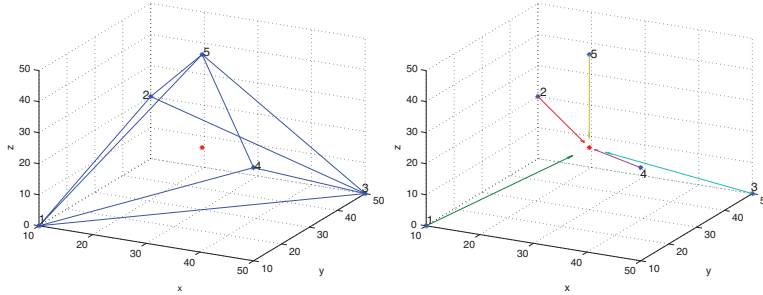


Figure 6.5: Arbitrary graph with 5 nodes.

6.7 Conclusions

In this chapter a novel approach to the problem of decentralized agreement toward a common point in space in a multi-agent system has been addressed. In

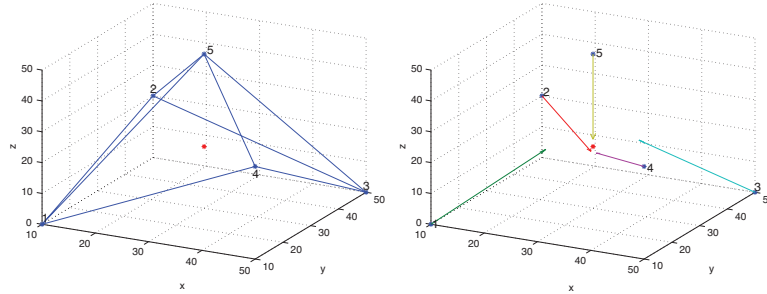


Figure 6.6: Arbitrary graph with 5 nodes.

this scenario, an agent is assumed to be able to sense the distance between itself and its neighbors and the direction in which it sees its neighbors with respect to its local reference frame. The proposed approach allows to perform an agreement on the network centroid, on a common reference frame and therefore on a common heading. Using this information a global positioning system for the agents using only local measurements can be achieved. Furthermore only point-to-point asynchronous communications between neighboring agents are allowed thus achieving robustness against random communication failures. The proposed algorithms can be thought as general tools to locally retrieve global information usually not available to the agents. In this way, any assumption on the absence of a common reference frame could be relaxed and therefore, simpler algorithms could be developed.

Collaboration among agents, which involves the computation of the relative distance and the direction of their line of sight with respect to the local reference frame of each agent, was limited to the exchange of the projection of the actual estimates along the direction of the line of sight. In the 2-case the proposed algorithm converges on arbitrary connected undirected graphs while in the d -dimensional case network connectivity is only necessary.

Chapter 7

Fault detection and recovery in consensus networks

7.1 Introduction

One recently discovered aspect of a Laplacian-based consensus for networked systems is its connection to the heat equation through the introduction of so-called partial difference equations as discrete analogs of partial differential equations (113, 114). This analogy enables the connection to traditional boundary-value problems. In particular, it has been showed in (115?) that the introduction of single anchor nodes, i.e., a single, immobile agent, results in a rendezvous at the location of that agent, provided that the underlying graph remains connected. Similarly, with multiple anchor nodes, the remaining agents converge to the convex hull spanned by the anchor nodes (115).

As a consequence of this, the immobile agents will in effect change the system performance most significantly in that the agents will no longer converge to the centroid of the initial configuration, as would otherwise be the case. Taking this observation one step further, one way in which mobile networks, executing a consensus-based control strategy, can be "hi-jacked" is by either adding a hostile (immobile) agent or by rendering one agent immobile. Moreover, if the hostile agent is moving, it would in essence be able to move all the original agents away from the target area. This fact can be thought of as a rather extreme form of non-robustness with respect to outliers in that outliers are given more or less

complete control over the system performance.

In this chapter we take these observations and discuss how to add robustness to the system in the sense that hostile/faulty agents may be identified and their influence nullified. In particular, what we propose in this chapter is a set of tools for achieving this in a decentralized manner under the banner of so-called *motion probes*. A motion probe is a maneuver executed either by a single agent or by a team of agents, intended to allow the agents to infer certain properties about the network. Moreover, these movements should be such that they preserve desirable properties, such as keeping the centroid static. We point out that such tool can be used as a mean of identification of faulty behavior (e.g. if an agent is stuck or is not responding is probably faulty) but we do not investigate how to achieve the task in this chapter: it will be object of future research. We show however that within this approach it is also possible to recover the original centroid of the good agents once the faulty agents have been identified.

The outline of this chapter is as follows: In Section 7.2, we briefly discuss the problem under consideration in the setting of linear consensus protocols. Section 7.3 presents a first result of this chapter, namely the motion probes for preserving the rendezvous point (typically the initial centroid of the network) and for nullifying future impacts of hostile/faulty agents. Following this, in Section 4, we present a tool for network fault recovery. It consists of a method to nullify any contribution to the final rendezvous point that the faulty agents may have caused prior to their detection. We conclude the chapter with some examples and, in Section 5, with the concluding remarks.

7.2 Problem description

We will be considering the discrete time version of the consensus problem, even though it should be noted that the results can be extended to the continuous time setting in a straightforward manner. A general formulation of the linear, nearest neighbor rule for solving the consensus problem is to let the state of the system evolve as

$$x(k+1) = Px(k), \quad (7.1)$$

where P is a stochastic, indecomposable, aperiodic matrix, as discussed in (?). Moreover, $x \in \mathbb{R}^n$ is an aggregated state vector, with each component x_i

representing a scalar state associated with agent $i = 1, \dots, n$.

There is a tight connection between the above formulation of the consensus problem and graph theory. In fact, we may model the network as an undirected graph $G = V \times E$, with V being a set of vertices $V = \{1, \dots, n\}$ that represent the agents, and where the edge set $E \subseteq V \times V$ encodes the network topology in that $(i, j) \in E$ if and only if agents i and j can share information. Based on a matrix representation of this graph, using algebraic graph theory (42), the graph can be encoded through its adjacency matrix A . The adjacency matrix is an $n \times n$ matrix such that $a_{i,j} = 1$ if and only if $(i, j) \in E$ and is 0 elsewhere.

Let $N_i \subset V$ be the set of vertices adjacent to vertex i , and let $|N_i|$ denote its cardinality. We can then define the degree matrix Δ as the diagonal matrix whose diagonal entries are $\Delta_{i,i} = |N_i|$. Using these matrices, a standard, discrete time model of consensus networks is the one defined by

$$x(k+1) = (I - \epsilon L)x(k), \quad (7.2)$$

where I is the identity matrix, $L = \Delta - A$ is the graph Laplacian of the graph G , and $\epsilon > 0$. Under this dynamics, the matrix P in Equation (7.1) becomes

$$P = I - \epsilon L. \quad (7.3)$$

Following the notation in (43), we will refer to P as a Perron matrix.

In the model studied in this chapter we assume that the topology of the network is represented by a time varying, undirected graph $G(t) = (V, E(t))$, where the edge set is time dependent, which corresponds to edges being created and disappearing in the network. This could for instance be caused by communication failures, or by the movements of the individual agents as they enter and leave each others' sensory ranges. Again we let the neighbors of agent i at time t be $N_i(t) = \{j \in V \mid (i, j) \in E(t)\}$.

Now, as the adjacency and degree matrices are time dependent, the Laplacian will depend on time as well and rather than explicitly computing $L(t)$, we assume that we have an enumeration of all possible graphs over n agents. We define T as the index set of all connected graphs

$$T = \{i \mid G_i = (V, E_i) \forall E_i \subseteq V \times V \text{ is connected}\}.$$

In fact, we will assume that the graph that is currently encoding the network topology is connected, i.e., its index belongs to T , and the linear consensus dynamics that we will employ can thus be given for $k \geq 0$ by

$$x(k+1) = P_{i(k)}x(k) \quad x(0) = x_0, \quad i(k) \in T. \quad (7.4)$$

Where x_0 is the initial state of the system. We slightly generalize the dynamics given by (7.3) assuming that the system matrix $P_{i(k)}$ (corresponding to the connected graph that describes the network topology at time k , i.e., $G_{i(k)}$) is given by

$$P_{i(k)} = I - \epsilon DL_{i(k)}, \quad (7.5)$$

where D is a positive definite, diagonal matrix representing the speed (or gain) of each agent. Obviously for $D = I$ (i.e., all agents have the same speed) equation (7.5) gives a Perron matrix.

It is straightforward to verify that P_i in Equation (7.5) is a stochastic, indecomposable, aperiodic matrix for any sufficiently small, positive $\epsilon \geq 0$. This follows since $P_i = I - \epsilon DL_i = I - \epsilon D(\Delta_i - A_i)$. Hence, for the diagonal entries of P_i to be positive, we need $P_i = I - \epsilon D\Delta_i$ to be positive. This means that the diagonal entries of P_i are $1 - \epsilon d_j \Delta_{i,jj}$. But, we know for sure that the maximum degree that a node can have is equal to $n - 1$, where n is the total number of agents, so an upper bound on ϵ such that the assumption on P_i is satisfied is that $\epsilon < \min_j 1/(d_j(n - 1))$, where d_j is the j^{th} diagonal entry of D .

7.3 Motion Probes

In this section we provide the basic tool - the so-called motion probe - for detection of misbehavior in a multi-agent system with single integrator dynamics that is running a linear consensus algorithm.

The formulation of the consensus dynamics in Equation (7.5) is only valid as long as all agents are executing the prescribed control laws. However, if a subset of the agents do not, the dynamics change and one would typically like to be able to detect this change in dynamics and mitigate its effects. For this, we propose to let individual agents perform controlled movements that is somewhat different from the pure consensus dynamics in order to excite the system.

In fact, the main contribution in this chapter is the characterization of what movements the agents should perform to excite the network so that tasks such as intruder and fault detection can be carried out so that the network initial weighted average stays invariant after the process.

To study the evolution of such systems we need to point out the connection between this formulation of the consensus dynamics and that of discrete time Markov chains. In fact, one can think of P_i as being the transition matrix in a Markov chain, with a corresponding, unique stationary distribution π_i such that $\lim_{k \rightarrow \infty} P_i^k = \mathbf{1}\pi_i^T$, where $\mathbf{1}$ is the vector with ones in each entry.

The following result can then be directly obtained:

Lemma 7.3.1 *The stationary distribution of*

$$\mathbf{P}(t) = \prod_{k=0}^{t-1} P_{i(k)}$$

for any $t \geq 1$, where $P_{i(k)} = I - \epsilon D L_{i(k)}$, $i(k) \in T$, is

$$\pi = D^{-1} \mathbf{1} \alpha$$

with α being a normalizing scalar such that $\sum_{j=1}^n \pi_j = 1$.

Although the proof follows directly from the basic properties of stochastic, indecomposable, aperiodic matrices, we state it here for the sake of completeness.

Proof: The proof is given in two steps:

1) The stationary distribution of $P_i = I - \epsilon D L_i$, $i \in T$, always exists because (as discussed above) P_i is a stochastic, indecomposable, aperiodic matrix as long as ϵ is sufficiently small. And, this stationary distribution satisfies $\pi_i^T P_i = \pi_i^T$, i.e., $\pi_i^T (I - \epsilon D L_i) = \pi_i^T$, which in turn implies that

$$\pi_i^T D L_i = \mathbf{0}.$$

Since L_i is the Laplacian of the connected undirected graph G_i , it is symmetric with $\mathbf{1}$ as the unique eigenvector such that $\mathbf{1}^T L_i = \mathbf{0}$. This implies that $\pi_i^T D = \beta \mathbf{1}^T$ for some $\beta \in \mathbb{R}$. Hence, the stationary distribution of P_i , normalized such that $\sum_{j=1}^n \pi_{i_j} = 1$ is

$$\pi_i = D^{-1} \mathbf{1} \alpha,$$

with α normalizing the distribution such that $\sum_{j=1}^n \pi_{i_j} = 1$.

In fact, since we produce the normalized stationary distribution, α only depends on D . Hence, π_i is only a function of D , i.e., it is not a function of i , which means that the stationary distribution of any P_i , $i \in T$, is

$$\pi^T = D^{-1} \mathbf{1} \alpha,$$

with a normalizing α .

2) Since we have just established that P_i for any $i \in T$ has the same stationary distribution defined by D , it follows that

$$\begin{aligned} \pi^T \mathbf{P}(t) &= \pi^T \prod_{k=0}^{t-1} P_{i(k)} = \pi^T P_{i(t-1)} \prod_{j=0}^{t-2} P_{i(j)} \\ &= \pi^T \prod_{j=0}^{t-2} P_{i(j)} = \pi^T, \end{aligned}$$

and hence

$$\pi^T = D^{-1} \mathbf{1} \alpha,$$

with α normalizing the distribution such that $\sum_{j=1}^n \pi_j = 1$, which concludes the proof. \blacksquare

Now, what we want is to move a subset of the agents in such a manner that we can infer certain properties of the network. However, at the same time we want to make sure that certain other properties of the network stay the same at the end of the moment. In particular, we will use Lemma 7.3.1 to establish constraints on the movements (or motion probes) that preserve the weighted average of the initial states.

In order to allow for the agents to exert a control action different from the consensus-based maneuver, we assume that the network behavior can be described by:

$$x(k+1) = P_{i(k)} x(k) + Bu(k) \quad x(0) = x_0, \quad i(k) \in T. \quad (7.6)$$

Here B is an $n \times n$ matrix whose is typically the identity matrix, u is a vector of inputs whose its i^{th} component u_i being the scalar input exerted by agent i .

Theorem 7.3.2 *Given the network dynamics in Equation (7.6). If*

$$\sum_{k=0}^{t-1} u(k) = 0,$$

then

$$\pi^T x(t) = \pi^T x(0),$$

where π is the stationary distribution in Lemma 7.3.1.¹

Proof: We have that

$$x(t) = \prod_{k=0}^{t-1} P_{i(k)} x(0) + \sum_{k=0}^{t-1} \prod_{j=0}^k P_{i(j)} B u(k).$$

Multiplying by π^T on both sides of the above equation and observing that π is the stationary distribution of $P_i \forall i \in T$, we get

$$\pi^T x(t) = \pi^T x(0) + \sum_{k=0}^{t-1} \pi^T B u(k).$$

Since $\pi^T B$ does not depend on k it can be taken out of the summation as

$$\pi^T x(t) = \pi^T x(0) + \pi^T B \sum_{k=0}^{t-1} u(k).$$

By hypothesis, $\sum_{k=0}^{t-1} u(k) = 0$, and hence

$$\pi^T x(t) = \pi^T x(0),$$

which concludes the proof. ■

The previous theorem can be understood in the context of the partial difference equation analogy with the heat equation. Any agent applying an input can be seen as an agent that is "warming up" or "cooling down" the network, depending on the sign of the input. Since the system is conservative (no heat can flow away), in order to recover the initial thermal equilibrium point the only information needed is how much heat has flown in or out from the network. This quantity corresponds to the integral of the applied input. Hence, if the integral is zero, the total heat present in the network has been preserved, and the initial,

¹ If the n agents are moving in a m -dimensional space, the extension to $\mathbb{R}^{n \times m}$ requires the presence of a relative inertial reference for each agent. This means that the assumption that $\sum_{i=0}^{k-1} u(i) = 0$ needs to hold for $u \in \mathbb{R}^{n \times m}$.

thermal equilibrium point will be reached under the regular evolution of the heat equation.

It should moreover be pointed out that such a motion probe can be performed by any agent in a completely decentralized fashion since no information is required to flow through the network for its computation. In other words, the motion probe can be used to achieve a variety of tasks like *obstacle avoidance*, *failure detection* (i.e., if an agent does not react to the motion probe it is clearly not running the consensus algorithm), and *connectivity preservation* (an agent may just slow down to avoid disconnection with a far agent and then speed up later to preserve the centroid). And, this is done while preserving the weighted centroid of the network, as per Theorem 7.3.2.

7.4 Fault recovery

If we assume that we have been able to locate a faulty agent (perhaps using the motion probe discussed in the previous section), what we would like to do is to isolate that agent from the network. Moreover, we would like to not only cancel out that agent's effect on the system after recovery, but also to nullify the agents total effect, from time $t = 0$ and onwards. The reason why this might be useful can for instance be seen in a networked robot system where we do not want to let the faulty agent drag the team away from the desired rendezvous point. Similarly, in a sensor network, we typically need to eliminate the contribution from a faulty sensor.

We will let the network topology be static in the following paragraphs, even though it should be noted that all the computations will still hold under slightly more general assumptions (as discussed later). We assume that the agents are ordered in such a way that the first $n - m$ agents are non-faulty, and the remaining m agents are faulty. In fact, we let the system dynamics be given by

$$x(k + 1) = \hat{P}x(k) + \hat{B}u(k) \quad x(0) = x_0. \quad (7.7)$$

Here \hat{B} is

$$\hat{B} = \left[\begin{array}{c|c} B_g & \mathbf{0}_{n-m \times m} \\ \hline \mathbf{0}_{m \times n-m} & B_f \end{array} \right] \quad (7.8)$$

where B_g and B_f are the input matrices of respectively the good and the faulty agents and B_g corresponds to the identity matrix with the appropriate dimensions. B_g is assumed to be the identity matrix since this system represent a collection of agents that though collaborating to achieve a common goal need to perform some tasks on their own (i.e., motion probes or else). However, \hat{P} is no longer a Perron matrix.

\hat{P} can be expressed as follows: As in the previous section, we let D denote the positive definite, diagonal weight (or gain) matrix associated with each agent. Moreover, let W_f be the $n \times n$ matrix whose entries are zeros except the bottom right $n \times m$ block which is the identity matrix.

$$W_f = \left[\begin{array}{c|c} \mathbf{0}_{n-m \times n-m} & \mathbf{0}_{n-m \times m} \\ \hline \mathbf{0}_{m \times n-m} & I_{m \times m} \end{array} \right]$$

Using this notation \hat{P} becomes

$$\hat{P} = I - \epsilon D(L - W_f L), \quad (7.9)$$

where L is the graph Laplacian.

It is straightforward to show that \hat{P} in Equation (7.9) can be written as the following, partitioned block matrix:

$$\hat{P} = \left[\begin{array}{c|c} P_g & D_f \\ \hline \mathbf{0}_{m \times n-m} & I_{m \times m} \end{array} \right] \quad (7.10)$$

Here D_f is a $(n - m) \times m$ matrix whose elements in the i^{th} row are non-zero only corresponding to faulty agents neighbors of agent i .

In the following we denote u_g the inputs to the good agents and u_f the inputs of the faulty ones. Since \hat{P} is block diagonal, we can now write the dynamics of the non-faulty agents (denoted by x_g) and view the position of the faulty agents (x_f) as inputs. We moreover recall that u_f will not affect x_g directly, and we get the following partitioned system:

$$\begin{aligned} x_g(k+1) &= P_g x_g(k) + D_f x_f(k) + B_g u_g(k) \\ x_f(k+1) &= x_f(k) + B_f u_f(k). \end{aligned} \quad (7.11)$$

Viewed at the level of the individual non-faulty agents, the dynamics of agent i is in fact given by

$$x_i(k+1) = x_i(k) - \epsilon d_i \sum_{j \in N_i} (x_i - x_j) + u_i(k),$$

where, as before, N_i is the set of vertices adjacent to vertex i . Letting $F = \{i \mid \text{agent } i \text{ is faulty}\}$ allows us to separate the contributions to the non-faulty agent's evolution as

$$x_{g,i}(k+1) = x_{g,i}(k) - \epsilon d_i \left(\sum_{j \in N_i/F} (x_{g,i} - x_{g,j}) \sum_{j \in N_i \cap F} (x_{g,i} - x_{f,j}) \right) + u_i(k).$$

We may call P_s the matrix that describes the dynamic of the good agents and see as input the distance between the faulty agents and their neighbors and corresponds to $P_s = I - \epsilon D L_i$ where L_i is the subgraph induced by the good agents. With simple algebraic manipulations one may note that $P_s = P_g + H$ where H is a $n - m \times n - m$ diagonal matrix whose i^{th} element corresponds to the number of faulty agents in the neighborhood of agent i multiplied by ϵd_i :

$$H = \epsilon D \text{diag} \left(|N_1 \cap F| \quad \cdots \quad |N_{n-m} \cap F| \right)$$

The following theorem provides a tool for network recovery after some agents have been disconnected from the network for some reason and it is of interest to nullify their contribution to the final state of the network. The theorem is stated for a fixed connected topology and then the result is extended to the case of a switching topology.

Theorem 7.4.1 *Let the network of agents be described by Equation (7.11). If the following conditions hold:*

1. *At time t_0 all m faulty agents have been detected.*
2. *The neighbors of faulty agents apply a control input such that at time t*

$$\sum_{k=t_0}^t u_{rec}(k) = - \sum_{k=0}^{t_0-1} (D_f x_f(k) - H x_g(k)).$$

Then, at time t a complete recovery of the weighted average of the initial states of the non-faulty agents has been performed, i.e.,

$$\pi^T x_g(0) = \pi^T x_g(t).$$

If, furthermore, the induced subgraph of G containing all the non-faulty agents is connected after time t

$$\lim_{k \rightarrow \infty} x_g(t+k) = \mathbf{1}\pi^T x_g(0).$$

Proof: The states of the non-faulty agents follow the equation

$$x_g(k+1) = P_g x_g(k) + D_f x_f(k) + B_g u_g(k).$$

What each non-faulty agent can actually measure, in the proposed discrete time consensus algorithm, is the difference between its own state and all the neighbors'. The generic neighbor i of a faulty agent j sees as input

$$N_{ij}(k) = e_i^T (D_f e_j e_j^T x_f(k) - H x_g(k))$$

and can easily keep the information about the total contribution from each neighbor by remembering $\sum_{k=0}^{t_0-1} N_{ij}(k)$. The evolution of the system can now be described by the following equations:

$$\begin{aligned} x_g(k+1) &= P_s x_g(k) + D_f x_f(k) - H x_g(k) + B_g u_g(k) \\ x_g(0) &= x_{g0} \\ x_f(k+1) &= x_f(k) + B_f u_f(k) \quad x_f(0) = x_{f0}. \end{aligned}$$

We recall that $P_s = P_g + H$ is a stochastic, indecomposable, aperiodic matrix by construction since $P_s = I - \epsilon D L_s(G)$, where $L_s(G)$ is the Laplacian of the induced subgraph of the non-faulty agents (that is assumed to be connected). The states of the good agents at time t_0 as function of the faulty agents and their initial state can be written as:

$$x_g(t_0) = P_s^{t_0} x_g(0) + \sum_{k=0}^{t_0-1} P_s^k N(k) + \sum_{k=0}^{t_0-1} P_s^k B_g u_g(k).$$

At time t_0 all the faulty agents are detected and each neighbor of a faulty node is disconnected. Then, after an arbitrary number of time steps, each agent that was a neighbor of a faulty one applies the proposed control input based on the information preserved locally about the contribution of the faulty neighbor to

its dynamic. For this, let $\widehat{u}_g = u_g + u_{rec}$ be the non-faulty agents' inputs between time t_0 and t , where u_g is any input that the non-faulty (good) agents may want to perform (i.e., a motion probe or else) and u_{rec} is the recovery input. When, at time t , each agent stops applying the recovery input, the state of the network is

$$\begin{aligned} x_g(t) &= P_s^{t-t_0}(P_s^{t_0}x_g(0) + \sum_{k=0}^{t_0-1} P_s^k N(k) \\ &\quad + \sum_{k=0}^{t_0-1} P_s^k u_g(k)) + \sum_{k=t_0}^{t-1} P_s^k B_g \widehat{u}_g(k). \end{aligned}$$

Multiplying both sides of the above equation by the stationary distribution of P_s , π^T :

$$\begin{aligned} \pi^T x_g(t) &= \pi^T P_s^{t-t_0}(P_s^{t_0}x_g(0) + \sum_{k=0}^{t_0-1} P_s^k N(k) \\ &\quad + \sum_{k=0}^{t_0-1} P_s^k B_g u_g(k)) + \pi^T \sum_{k=t_0}^{t-1} P_s^k B_g \widehat{u}_g(k), \end{aligned}$$

since $\pi^T P_s^k = \pi^T$, $\forall k \geq 0$, we get

$$\begin{aligned} \pi^T x_g(t) &= \pi^T x_g(0) + \pi^T \sum_{k=0}^{t_0-1} N(k) + \pi^T B_g \sum_{k=0}^{t_0-1} u_g(k) \\ &\quad + \pi^T B_g \sum_{k=t_0}^{t-1} \widehat{u}_g(k). \end{aligned}$$

Recalling that B_g is the identity matrix and $\widehat{u}_g = u_g + u_{rec}$ gives

$$\begin{aligned} \pi^T x_g(t) &= \pi^T x_g(0) + \pi^T \sum_{k=0}^{t_0-1} N(k) + \pi^T \sum_{k=0}^{t_0-1} u_g(k) \\ &\quad + \pi^T \sum_{k=t_0}^{t-1} u_g(k) + \pi^T \sum_{k=t_0}^{t-1} u_{rec}(k). \end{aligned}$$

Since

$$\sum_{k=t_0}^t u_{rec}(k) = - \sum_{k=0}^{t_0-1} N(k)$$

we get

$$\pi^T x_g(t) = \pi^T x_g(0) + \pi^T \sum_{k=0}^{t-1} u_g(k).$$

Then, if u_g is identically null or is the resulting of tasks accomplished with motion probes we know that

$$\sum_{k=0}^{\tau-1} u(k) = 0, \quad \forall \tau \geq t_0,$$

and thus

$$\pi^T x_g(t) = \pi^T x_g(0),$$

thus proving the first part of the theorem.

If furthermore G stays connected after all the faulty agents have been removed from the network, the hypothesis for the consensus algorithm to converge are satisfied and:

$$\lim_{k \rightarrow \infty} x_g(k) = \mathbf{1} \pi^T x_g(0),$$

and the second part of the theorem follows. ■

We point out the following main features of the proposed recovery procedure:

1. Only the agents that are neighbors to a faulty agent need to apply the recovery input.
2. The information needed by the generic agent to recover after detection is

$$N_{ij}(k) = e_i^T (D_f e_j e_j^T x_f(k) - H x_g(k))$$

that essentially consists of the summation of the difference between its state and the faulty neighbor's (i.e., their distance) starting from the initial instant of time, i.e., a single variable for each neighboring agent.

3. Any agent that has detected a faulty agent in its neighborhood may apply the recovery at any time. There is no need that all the faulty agents are detected at the same time. The recovery procedure may then be applied in an asynchronous way.

The extension of the previous theorem to the switched topology case needs a brief introduction. If the agents have an identifying ID such that through communication any agent is able to identify any other, then the extension to the switched topology case is trivial if we assume that the ID of the faulty nodes can be broadcasted through the network. In such a case, what an agent needs to perform is simply the proposed recovery procedure. The only difference lies in the fact that the set of neighbors may change and so more memory is needed to remember the total contribution from all of the past neighbors. Such a quantity of memory in the worst case corresponds to $(n - 1)N$ variables, where n is the number of agents and N is the dimension of the space in which they are moving.¹

Next we give an example of fault recovery for a simplified case of a network of unmanned aerial vehicles. These UAVs are assumed to be tasked with flying at the same height while avoiding be dragged to the ground when one of them is shot down. These agents are moreover assumed to be aware only of the relative distances of the neighbors.

Example 7.4.2 *Let the evolution of the network of agents in Figure 7.2 represent a set of unmanned aerial vehicles that need to rendezvous at a certain altitude, described by*

$$x(k + 1) = Px(k),$$

where

$$P = I - \epsilon \begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ -1 & -1 & -1 & 3 \end{bmatrix}$$

and $x(0) = [66.12 \ 62.24 \ 68.86 \ 69.33]^T$. The rendezvous point is then at 66.64 and we note that the average of the states of the non-faulty agents 1, 2 and 3 is 65.74 at the initial time. The evolution of such network is shown in Figure 7.1.

¹If this hypothesis does not hold but the topology is slowly switching, in the sense that detection is "fast" respect to changes in the network, then a sufficient condition for fault recovery with no communication and limited memory is that from the instant of time that agent i becomes faulty until the time in which the fault detection occurs the neighbors of agent i do not decrease in number (in the sense that new neighbors are welcomed but no neighbor may leave \mathcal{N}_i). In such a case any agent needs to remember the contribution of only its actual neighbors.

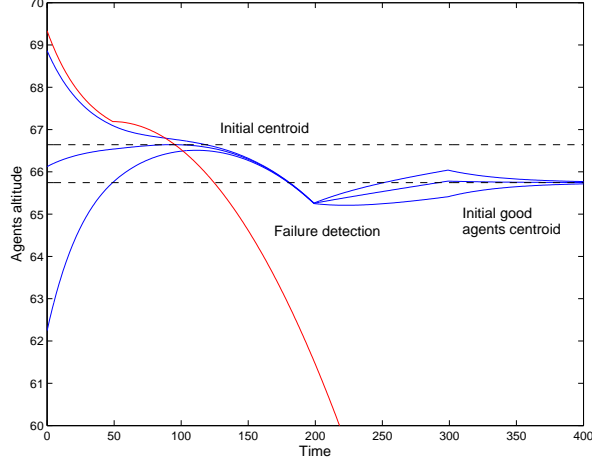


Figure 7.1: Evolution of the network of agents in Example 7.4.2.

The agents perform the consensus algorithm on their altitude when at $k = 50$ agent 4 is shot down and starts falling to the ground as shown in Figure 7.1. The other agents, still unaware of what happened try to follow his movements, being dragged to the ground themselves. The dynamic of the network during this period is described by:

$$x_g(k+1) = \left(I - \epsilon \begin{bmatrix} 3 & -1 & -1 \\ -1 & 3 & -1 \\ -1 & -1 & 3 \end{bmatrix} \right) x_g(k) - \epsilon \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} x_f(k)$$

$$x_f(k+1) = x_f(k) + u(k).$$

Then at time $k = 200$ by some means the neighbors of the broken agent realize that agent 4 was broken (i.e., identify the suspicious behavior and execute a motion probe or through other means achieve the same result). Agent 4 is then disconnected from the neighbors, the dynamic of the network then becomes:

$$x_g(k+1) = \left(I - \epsilon \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix} \right) x_g(k) + \epsilon \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} u(k)$$

Where $u(k)$ is the recovery input. In this example all the agents detect the misbehavior, disconnect the faulty agent and apply the recovery input at the same

instant of time for clarity sake. All these operations may take place at different instants of time in a real application.

Agent 1 knows the summation of inputs due to agent 4

$$\sum_{i=0}^{k-1} x_1(i) - x_4(i) = 72.8$$

Agent 2 knows

$$\sum_{i=0}^{k-1} x_2(i) - x_4(i) = -48.2$$

Agent 3 knows

$$\sum_{i=0}^{k-1} x_3(i) - x_4(i) = 158.42$$

So all of them need to apply an input whose summation over a finite number of steps is opposite to the one applied by the broken agent. For simplicity the chosen input is constant with a length of 100 time steps (i.e., the agents are completely free to do whatever they like as long as the total contribution of agent 4 is nullified). Such inputs for agent 1, 2 and 3 are:

$$u_{rec,1}(k) = -0.72, \quad k = 201, \dots, 300$$

$$u_{rec,2}(k) = 0.48, \quad k = 201, \dots, 300$$

$$u_{rec,3}(k) = 1.58, \quad k = 201, \dots, 300$$

and zero elsewhere.

When they all finish applying the recovery at time 300, we note in Figure 7.1 that the average of their altitudes has become exactly their average at the initial instant of time, namely 65.74. From this point on the network evolves as a standard consensus network, reaching a practical rendezvous around time $k = 400$. ■

7.5 Fault Diagnosis

In this section we focus on how to add robustness to the consensus algorithm by providing a method of active fault diagnosis and identification to be paired with the fault recovery algorithm. The main contribution of this work is:

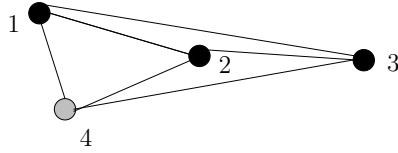


Figure 7.2: Network of agents in Example 7.4.2.

1. A classification of some faults in a sensor network that disrupt the consensus dynamics.
2. A heuristic to classify suspected behavior of the nodes when a fault is likely.
3. The application of a method developed in (99) to actively probe the nodes suspected to be faulty.
4. An heuristic to identify faulty nodes once they have been probed.
5. The application of a method developed in (99) to recover the network convergence property after a fault has been detected.

We consider two different kinds of faults.

1. *Stuck at* : This fault is usually due to faults in the electronics of the sensor: it consists in a sensor that is "stuck" at an arbitrary value. This makes the nodes that correctly follow the prescribed communication protocol drift away toward the faulty node value.
2. *Divergence fault* : This fault is due to software or hardware failure: it consists in a sensor constantly increasing (or decreasing) its state value indefinitely (or until a saturation occurs). This makes the nodes that correctly follow the communication protocol drift away in the direction of the faulty node value.

Our approach for diagnosis and recovery can be summarized into three steps that the single nodes can apply independently and asynchronously:

1. *Detection of suspicious behavior* : This step is performed by each node toward each of its neighbors. The detection of suspicious behavior is carried out by observing the neighbors' states and checking through a set of rules the presence of anomalous behavior.

2. *Fault identification* : This step is performed by the nodes who identified a suspicious behavior in one of their neighbors. It consists in applying an appropriate input to the network and check the reaction of the suspected node to identify an eventual fault.
3. *Fault recovery* : This step is performed by each neighbor of a faulty node once its faulty behavior has been detected. It allows the sensor network to disconnect the faulty node and recover the average of the initial states as if the faulty node had never influenced the network dynamics.

Detection of suspicious behavior

In our approach each node constantly observes the neighbors states checking for suspicious behavior. In the following we identify some behaviors associated with their respective fault and we will refer to them as "suspicious" throughout the rest of the chapter.

1. *Stuck at* : This fault is characterized by a node that doesn't update anymore its state but remains visible to its neighbors. To identify such behavior is sufficient to check whether the state of a particular node is not equal to the neighbors' state and it is not changing. In the general case the state of a healthy node may not change only if it already corresponds to the average of the initial states and the rest of the network is in a very unlikely trajectory that keeps the input of such node always equal to zero. A rule of the thumb to detect such behavior is simply to check whether the neighbor state is not changing for a sufficiently long time despite the difference between the node state and the suspected node state is different from zero and greater than an appropriate ε small enough such that this kind of behavior can be detected before all the nodes converge to the same value.
2. *Multiple stuck at faults* : In this case as discussed in (115), the states of the nodes converge toward a value in the convex hull spanned by the nodes stuck at a certain value. This kind of fault is easily detected by observing that the nodes do not reach the same state after a sufficiently long time (where for sufficiently long we mean 4 – 5 times the slowest time constant

given by the second largest eigenvalue that corresponds to the algebraic connectivity of the graph representing the network).

3. *Divergence fault* : This fault is characterized by an indefinite constant increment (or decrement) of the node's state. This kind of fault can be due for instance to software or hardware bugs. This fault prevents the network to converge toward a common value. As such it can be identified by detecting the sustained increments (or decrements) of the node through model based identification techniques.

Algorithm description

We now present the actual algorithm run by each node to perform identification and fault recovery. The generic node i runs the following algorithm with parameters.

- Δ : a sliding window horizon. This value should be comprised between the slowest and fastest time constant in the system.
- ε : a threshold to determine if a node value has remained constant; it will be used to identify a "stuck at" node.
- γ : a threshold to determine if agent j is suspected to be affected by a "divergence fault".
- T_{max} : roughly an estimation of the consensus convergence time; a good choice is 4 – 5 times the slowest time constant given by the graph algebraic connectivity.

Algorithm 8 (Diagnostic Algorithm)

1. Let $t = 0$;
2. While $t \leq T_{max}$, do
3. For each neighbor $j \in N_i$:
If $|x_j(t+k) - x_j(t)| < \varepsilon, \forall k = 0, \dots, \Delta$, or

if $|x_j(t+1+k) - x_j(t+k)| > \gamma > 0, \forall k = 0, \dots, \Delta,$

label that node as suspected.

4. If at least one neighbor j is labeled as suspected, execute a motion probe and observe its reaction.
5. If during the execution of the motion probe, node j does not change behavior, namely if it remains stuck at its value within a threshold ε or it continues to diverge within a certain threshold γ , label it as faulty.
6. For each neighbor j labeled as faulty, disconnect it and execute the recovery for node j .
7. end while.
8. Stop.

7.6 Simulations

In this section simulations are provided to show the effectiveness of the heuristics. We take the sensor network in Figure 7.3 as case study for the following examples. The network consists in 9 sensors labeled $S_i : i = 1, \dots, 9$, that at time $t_0 = 0$ perform a measurement and then fuse such information by performing iterative averaging based on the Laplacian of the network graph. In the following the convergence time will be shown as number of iterations. The initial state of the network has been taken randomly with a uniform distribution between 0 and 10:

$$x(t_0) = [0.7818; 4.4268; 1.0665; 9.6190; 0.0463 \\ 0.0463; 7.7491; 8.1730; 8.6869; 0.8444]$$

'Stuck at' fault

In Figure 7.4 is shown the evolution of the state of the sensor network. At time $T_1 = 50$ the sensor node S_1 experience a fault and remains stuck at a value around 2.4. As a consequence if the fault recovery algorithm proposed in this chapter is not applied we may see that all the nodes wrongly converge toward the state of the faulty node disrupting the information retrieved from the measurements.

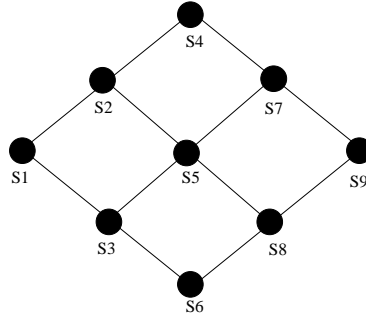


Figure 7.3: The sensor network topology used for the simulations.

In Figure 7.5 is shown the evolution of the same network when the proposed algorithm is implemented. At time $T_1 = 50$ the sensor node S_1 breaks again. The neighbors of sensor S_1 , namely sensor S_2 and S_3 , after some time (the amount of such time is a design parameter that has to be estimated as function of the worst case convergence time) detect the suspicious behavior of node S_1 and perform a Motion Probe. The Motion Probe performed has been chosen to be a single period lasting 20 iterations of a square wave with amplitude equal to 0.1. Both sensors after performing the Motion Probe detect the faulty behavior of node S_1 that is not reacting. Consequently they disconnect it from the network and apply the recovery algorithm nullifying its entire contribution to the network. From this point on the network evolves as a standard consensus network and converges to the average of the initial state of the healthy nodes.

'Multiple Stuck at' faults

We now study the case of multiple stuck at faults and show how the recovery algorithm can be applied asynchronously with respect to each of its phases.

In Figure 7.6 is shown the evolution of the sensor network in Figure 7.3 for the same initial conditions as for previous simulations. At time $T_1 = 50$ sensor S_1 stops updating its states. At time $T_2 = 150$ also sensor S_7 stops updating. At this point the sensor network does not converge anymore to a common value but each sensor state converges toward a point in the convex hull spanned by the faulty nodes values. In this condition any attempt to retrieve information about the original average of the measurements would fail since each node presents a

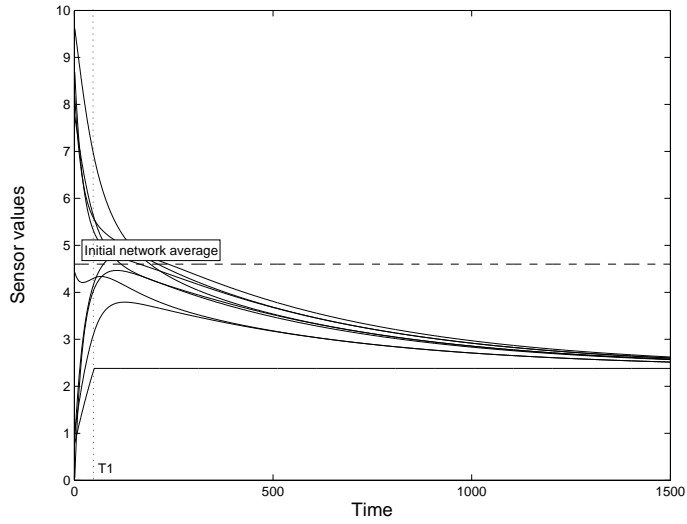


Figure 7.4: Evolution of a sensor network with a single "stuck at" fault.

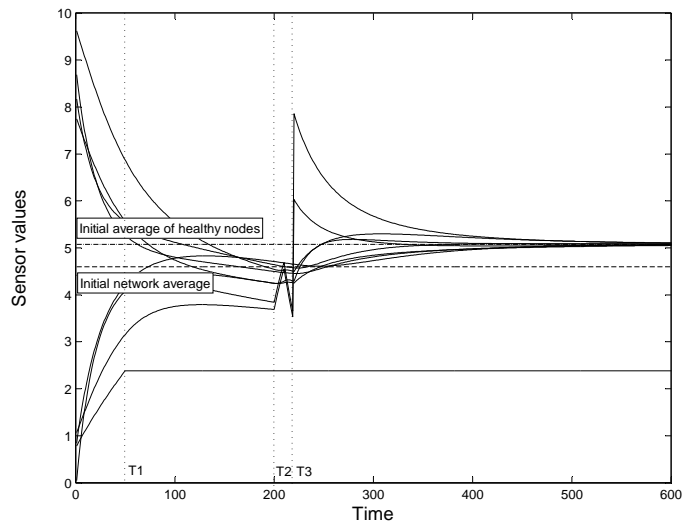


Figure 7.5: Evolution of a sensor network with a single "stuck at" fault when the fault recovery procedure is implemented.

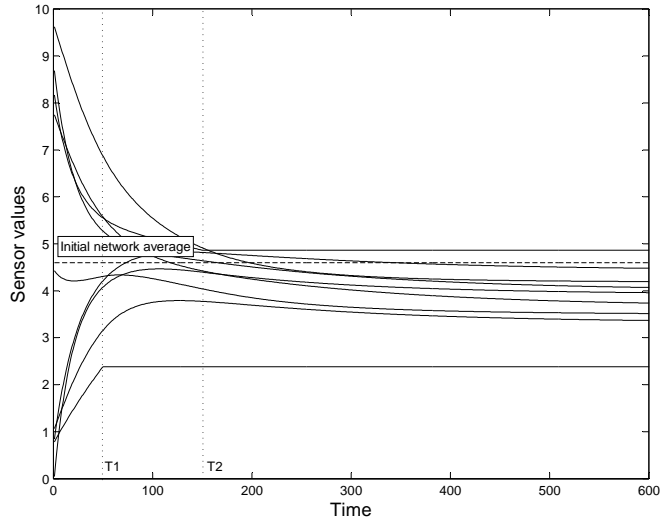


Figure 7.6: Evolution of a sensor network with multiple "stuck at" faults.

different estimate.

In Figure 7.7 is shown the evolution of the same network with two faulty nodes implementing the proposed algorithm for recovery. At time $T_1 = 50$ sensor S_1 is stuck; at time $T_2 = 150$ sensor S_7 is stuck as well. At time T_3 sensors S_2 and S_3 suspect of the behavior of node S_1 and perform a Motion probe. At time T_4 sensors S_2 and S_3 detect the faulty behavior of node S_1 , disconnect it from the network and apply the recovery input to nullify its contribution. At time T_5 sensors S_4 and S_5 (neighbors of S_7) detect the suspicious behavior of the second faulty nodes, S_7 , while sensor S_9 for some reason does not recognize such behavior. At time T_6 sensors S_4 and S_5 complete the motion probe, detect no reaction and disconnect from sensor S_7 nullifying its contribution. After some time also node S_9 detects the suspicious behavior of node S_7 and at time T_7 it disconnects it and applies the recovery. At this point the network of the remaining nodes, that despite the faulty nodes remained connected, converges toward the initial average of the healthy nodes' states. This simulation gives an example of how the phases of fault identification and recovery can be applied asynchronously by the neighbors of the faulty nodes.

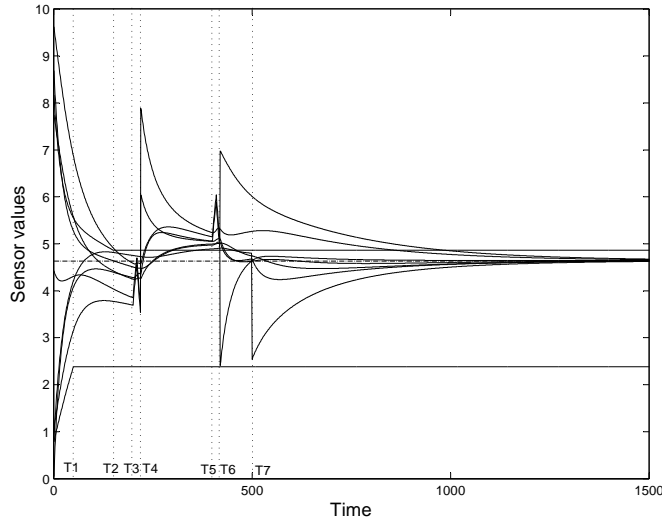


Figure 7.7: Evolution of a sensor network with multiple "stuck at" faults when the fault recovery procedure is implemented.

'Divergence' faults

In this last set of simulations we show how the proposed method for fault identification and recovery can be applied also to different or more general kinds of fault.

In Figure 7.8 is shown the sensor network in Figure 7.3 performing consensus. Here at time $t = T_1$ node S_1 starts to disobey to the consensus protocol and starts to increase (in such a case, linearly) indefinitely its value due to a software or hardware bug. The remaining nodes in the network unaware of the fault start following its path and the original measurements are lost.

In Figure 7.9 is shown the same network with the same initial conditions as the previous examples with a fault at sensor S_1 at $t = T_1$ when the fault recovery and identification procedure is implemented. This time the neighbors of sensor S_1 detect its suspicious behavior thanks to a fault model embedded in their memory. As soon as they detect a neighbor increasing or decreasing at a certain rate they execute a motion probe and wait for the node's reaction. Since node S_1 does not change its behavior as function of the inputs given by nodes 2 and 3, it is correctly identified as faulty and disconnected from the network.

Remark 7.6.1 *Since the algorithm is decentralized, the overhead required is pro-*

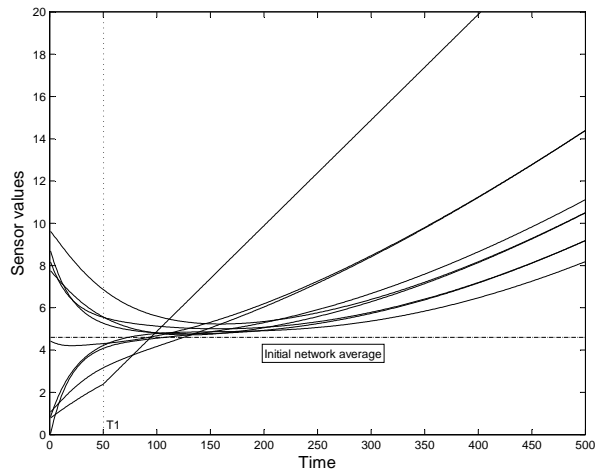


Figure 7.8: Evolution of a sensor network with single divergence fault.

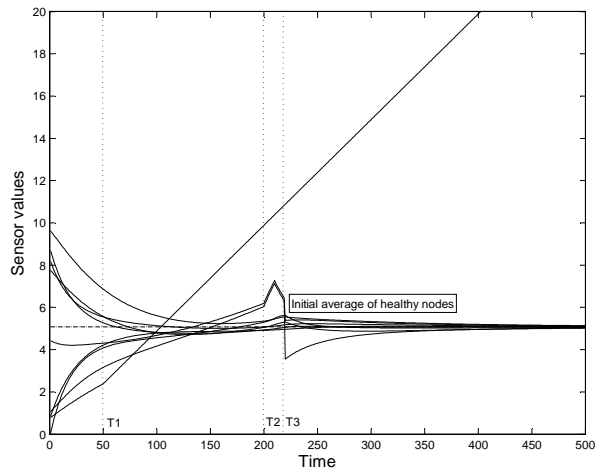


Figure 7.9: Evolution of a sensor network with single divergence fault when the fault recovery procedure is implemented.

portional to the average degree of the network. If the average degree of the network is constant as the number of nodes increases, the trade-off between overhead/performance improves. Furthermore, the presence of only one faulty node disrupts the network convergence properties with any number of nodes. It becomes clear that as the number of nodes increases, the probability of node failure increases and the fault diagnosis and recovery algorithm becomes essential.

7.7 Conclusions

In this chapter we consider the problem of how to move agents in a networked system in such a way that they do not change the desired rendezvous point during the maneuver. Such movements are referred to as motion probes. And, in particular, we show how such motion probes can be used to identify faulty agents that do not exhibit the prescribed dynamical behavior. Moreover, once these faulty agents have been identified, we show how to nullify their impact on the behavior of the non-faulty agents. We also outline some particularly promising directions for future research in this new area of motion probes for exciting networked control systems.

Chapter 8

Consensus based decentralized Laplacian Spectrum estimation

8.1 Introduction

Nowadays a great effort is made by the research community to provide coordination and estimation algorithms for networked multi-agent systems (6, 7, 24, 50). Such systems represent an ideal abstraction of actual networks of mobile robots that are envisioned to perform the most various kind of tasks in the future.

A key feature of any implementation of a networked multi-agent system is the network topology. Algebraic graph theory (42) is widely known to provide powerful tools and abstractions to describe networks of agents, e.g., proximity graphs (16). In particular, the representation of the network topology of a multi-agent system by graphs (116), where nodes represent agents and edges represent the existence of an interaction between them, has proven to be a powerful modeling tool.

In algebraic graph theory a great effort has been made to characterize the topological properties of a graph. The spectral analysis has proven to be an effective tool to achieve that. The key idea is to encode a graph topology through a matrix and then investigate its properties from an algebraic standpoint. Unfortunately this approach cannot be used in a networked system due to its centralized nature. In order to overcome such a limitation, we have switched the point of view: instead of investigating the properties of the matrix encoding the graph

from an algebraic perspective, we have encoded the same properties within a dynamical system which is distributed by nature and from which these properties can be retrieved by applying tools coming from the signal processing theory.

In this chapter a novel algorithm based on local interactions between agents to retrieve information concerning the eigenvalues of the whole network is proposed. The availability of this information in a decentralized fashion paves the way for the design of new control and estimation algorithms. The basic idea of the algorithm is to make the state of the agents oscillate only at frequencies corresponding to the eigenvalues of the network topology. In this way, the problem of decentralized eigenvalues estimation is mapped into a problem of signal processing that each agent can efficiently and independently solve by applying the *Fast Fourier Transform (FFT)* algorithm. We present our theory in continuous time. The content of this chapter has been published in (117).

We point out that while the proposed method is adapted to estimate the eigenvalues of the Laplacian matrix, it can be trivially extended to any other symmetric positive semi-definite matrix.

8.2 Related literature

The second smallest eigenvalue of the laplacian matrix, i.e., the algebraic connectivity of the communication graph, plays a crucial rule in the convergence time of various control and estimation algorithms (23). Several works can be found in the literature dealing with the estimation (and control) of the algebraic connectivity.

In (118) a geometric analysis of wireless connectivity in vehicle networks is proposed. In this work, the authors introduce a localized notion of connectivity and propose a function to measure its robustness which is proven to provide a sufficient condition for global connectedness of the network under certain conditions.

In (119) a decentralized algorithm to increase the connectivity of a multi-agent system is proposed. The authors introduce a decentralized supergradient algorithm which allows to maximize the algebraic connectivity. In particular, the authors prove that a supergradient direction for the algebraic connectivity is a function of the corresponding eigenvector whose computation is carried out by the Decentralized Orthogonal Iteration Algorithm, while the control action

aiming to drive the swarm toward a configuration corresponding to the optimal Laplacian is based on a potential flow approach.

In (120) a potential field for maintaining connectivity of mobile networks is proposed. The authors consider the problem of controlling a network of agents so that the resulting motion always preserves the connectivity property of the network itself. To this end, the connectivity condition is translated to differentiable constraints on the individual agent motion by considering the dynamics of the Laplacian matrix and its spectral properties. Artificial potential fields are then used to drive the agents to configurations away from the undesired space of disconnected networks while avoiding collisions with each other.

In (121) the problem of controlling a group of agents so that the resulting motion always preserves the connectivity property of the underlying network is faced. The authors propose a distributed feedback and provably correct control framework that imposes no restrictions on the network topology other than the desired connectivity specification. The approach is based on a key control decomposition, where connectivity control of the network structure is performed in the discrete space of graphs and relies on local estimates of the network topology, algebraic graph theory, and market-based control, while motion control of the agents is performed in the continuous configuration space by means of local potential fields used to maintain nearest neighbor links.

In (122) a decentralized estimation procedure that allows each agent to track the algebraic connectivity of a time-varying graph is proposed. The authors propose a decentralized power iteration algorithm that estimates the eigenvector corresponding to the second smallest eigenvalue of a weighted Laplacian matrix. This eigenvector estimation procedure is then exploited to estimate the algebraic connectivity of a graph. This information is used by a decentralized controller which aims to maintain the global connectivity of the graph over time.

In (123) the connectivity maintenance problem for ad-hoc networks of robotic agents with double integrator dynamics is investigated. In particular, the problems whether control inputs can be defined for each agent in order to maintain network connectivity and whether it is possible to compute the closest connectivity-maintaining controls in a distributed fashion by assuming certain desired controls are given to the agents are addressed.

In Sahai T. et al. 2010 (?) the problem of computing with a decentralized

algorithm the eigenvectors and eigenvalues of the Laplacian matrix is addressed with a similar approach respect to Franceschelli et al. 2009 (117). The authors propose the use of the discrete wave equation as local interaction rule in a network of agents to induce an oscillating behavior so that the computation of the fast Fourier transform by the generic agent yields the local component of every eigenvector of the Laplacian matrix. Their work differs with respect to (117) in that they use a different local interaction rule to produce the oscillating behavior and propose the application of the method to clustering problems by showing its improved convergence time respect to the state of the art algorithms based on random walks.

The proposed approach presents several advantages compared to the results which can be found in the literature. First of all, it is decentralized by nature and it does not require any additional effort for the decentralization of centralized techniques such as in the case of the Power Iteration Algorithm or the Orthogonal Iteration Algorithm . Secondary, it does provide an estimate of the algebraic connectivity with a predefined accuracy in a finite time, while all the other approaches provide estimates with an asymptotic convergence with no information concerning the achievable accuracy. Finally, our approach provides not only an estimate of the algebraic connectivity but for the whole spectrum of the graph describing the network topology, thus providing additional information which could be used for several control purposes.

8.3 Problem description

Let us consider a network of agents whose interactions can be described by a time varying graph $\mathcal{G}(t) = \{V, \mathcal{E}(t)\}$. $V = \{1, \dots, n\}$ is the set of nodes, where n is the number of agents. $\mathcal{E}(t) \subseteq \{V \times V\}$ is the set of edges: an edge $e_{i,j}(t)$, with $i \neq j$, exists between nodes i and j at time t if agent i interacts with agent j at time t .

We define $A(t)$ the time varying adjacency matrix: it is a $n \times n$ matrix whose generic element $a_{i,j}(t) = 1$ if $i \neq j$ and $e_{i,j}(t) \in \mathcal{E}(t)$, $a_{i,j}(t) = 0$ otherwise. The number of the incoming edges Δ_i of node i is called the *indegree*. We define $\Delta(t)$ as the degree matrix: it is a $n \times n$ time varying diagonal matrix whose elements are $\Delta_i(t)$, i.e., the indegree of node i at time t .

In the following we will refer to $\mathcal{N}_i(t)$ as the neighborhood of agent i , namely the set of indices of the agents directly connected through an edge with agent i at time t . Clearly, it is $|\mathcal{N}_i(t)| = \Delta_i(t)$.

Finally we define the time varying Laplacian of graph \mathcal{G} as $\mathcal{L}_{\mathcal{G}}(t) = \Delta(t) - A(t)$. To simplify the notation we will refer to it as $\mathcal{L}(t)$ implying its dependence on \mathcal{G} .

Let us now review some properties of the Laplacian of an undirected graph. First of all, \mathcal{L} is a weakly diagonal dominant symmetric matrix by construction. Furthermore we have that the row sum and the column sum are each equal to zero. In particular any graph Laplacian has always at least one null structural eigenvalue whose corresponding eigenvector is the vector of ones $\mathbf{1}$ of appropriate dimensions; in other words $\forall \mathcal{G}, \mathcal{L} \mathbf{1} = \mathbf{0}$ and $\mathbf{1}^T \mathcal{L} = \mathbf{0}^T$. The number of null eigenvalues corresponds to the number of connected components of \mathcal{G} and $\text{Rank}(\mathcal{L}) = n - c$, where n is the number of nodes and c is the number of connected components of \mathcal{G} (24).

Furthermore, being the Laplacian a real symmetric matrix all its eigenvalues are real. In addition, according to the Gershgorin disc theorem they are all positive and can be located in $[0, 2 \Delta_{max}]$, where Δ_{max} is the maximum degree between the nodes in the graph.

We point out that while we are focused on determining the eigenvalues of \mathcal{L} we can carry out the very same analysis for different matrices encoding a graph \mathcal{G} as long as they are symmetric and possibly positive semi-definite. One of such matrices is the normalized Laplacian for which it is known that the maximum eigenvalue is always unitary thus avoiding any issues regarding the optimal sampling frequency to be used. We presented the method focusing on the standard Laplacian since it has more evident applications in multi-agent systems. In particular, if a multi-agent system employs a feedback based on the Laplacian, the knowledge of the spectrum of the network not only provides critical information on the topology itself but also on the dynamical system that encodes the network topology.

8.4 Proposed algorithm

The proposed algorithm consists of each agent performing the following state updating rule:

$$\begin{cases} \dot{x}_i(t) = \sum_{j \in \mathcal{N}_i(t)} (z_j(t) - z_i(t)), \\ \dot{z}_i(t) = -\sum_{j \in \mathcal{N}_i(t)} (x_i(t) - x_j(t)). \end{cases} \quad (8.1)$$

The behavior of the network state when each agent performs the above updating rule can be described by the following time varying autonomous linear system:

$$\begin{bmatrix} \dot{x}(t) \\ \dot{z}(t) \end{bmatrix} = \mathcal{A}(t) \cdot \begin{bmatrix} x(t) \\ z(t) \end{bmatrix} \quad (8.2)$$

where

$$\mathcal{A}(t) = \begin{bmatrix} \mathbf{0} & \mathcal{L}(t) \\ -\mathcal{L}(t) & \mathbf{0} \end{bmatrix} \quad (8.3)$$

and $\mathbf{0}$ is the null $n \times n$ matrix.

This is a linear switching system, where the linear autonomous dynamics change abruptly when an edge is added or removed from the network.

In the following we will refer to \mathcal{A} as the matrix governing the dynamics of the linear system during any interval of time in which no topology change occurs.

Note that for any network topology the matrix \mathcal{A} is skew symmetric (i.e., $\mathcal{A}^T = -\mathcal{A}$). Moreover, any skew symmetric matrix has eigenvalues only on the imaginary axis of the Gauss plane. In particular, as proved in the following theorem, the eigenvalues of \mathcal{A} can be analytically derived from the eigenvalues of the Laplacian matrix \mathcal{L} .

Theorem 8.4.1 *Let \mathcal{G} be an undirected graph with Laplacian \mathcal{L} . Let matrix \mathcal{A} be defined as in eq. (8.3). To any eigenvalue $\lambda_{\mathcal{L}}$ of \mathcal{L} it corresponds a couple of complex and conjugates eigenvalues $\lambda_{\mathcal{A}}, \bar{\lambda}_{\mathcal{A}}$ of \mathcal{A} , namely $\lambda_{\mathcal{A}}, \bar{\lambda}_{\mathcal{A}} = \pm j\lambda_{\mathcal{L}}$, whose corresponding eigenvectors are as follows:*

$$v_{\lambda_{\mathcal{A}}} = [v_{\lambda_{\mathcal{L}}} \quad jv_{\lambda_{\mathcal{L}}}]^T, \quad \bar{v}_{\bar{\lambda}_{\mathcal{A}}} = [v_{\lambda_{\mathcal{L}}} \quad -jv_{\lambda_{\mathcal{L}}}]^T$$

Proof:

By definition, the eigenvalues of \mathcal{A} are the solutions of

$$\det(\mathcal{A} - \lambda I) = \det \left(\begin{bmatrix} -\lambda I & \mathcal{L}(t) \\ -\mathcal{L}(t) & -\lambda I \end{bmatrix} \right) = 0.$$

Since \mathcal{A} is a block matrix whose blocks commute (124), then

$$\det(\mathcal{A} - \lambda I) = \det(\lambda^2 I + \mathcal{L}^2) = 0.$$

Hence,

$$\det(\mathcal{L}^2 + \lambda^2 I) = \det(\mathcal{L} + j\lambda I) \det(\mathcal{L} - j\lambda I) = 0.$$

Let us now recall that $\lambda_{\mathcal{L}}$ are the eigenvalues of the Laplacian matrix L , that is: $\det(\mathcal{L} - \lambda_{\mathcal{L}} I) = 0$ which are all real and positive. Therefore, by substituting $\lambda_{\mathcal{L}} = -j\lambda$ within the term $\det(\mathcal{L} + j\lambda I)$ and $\lambda_{\mathcal{L}} = j\lambda$ within the term $\det(\mathcal{L} - j\lambda I)$, we obtain: $\lambda_{\mathcal{A}}, \bar{\lambda}_{\mathcal{A}} = \pm j\lambda_{\mathcal{L}}$, thus proving the first statement. Now, by definition, the eigenvectors of \mathcal{A} are the solutions of:

$$\begin{bmatrix} \mathbf{0} & \mathcal{L} \\ -\mathcal{L} & \mathbf{0} \end{bmatrix} \cdot \begin{bmatrix} v' \\ v'' \end{bmatrix} = \lambda_{\mathcal{A}} \begin{bmatrix} v' \\ v'' \end{bmatrix}$$

from which by substituting $\lambda_{\mathcal{A}} = j\lambda_{\mathcal{L}}$, we obtain the following linear system of equations:

$$\begin{cases} \mathcal{L} v'' = \lambda_{\mathcal{A}} v' \\ -\mathcal{L} v' = \lambda_{\mathcal{A}} v'' \end{cases}.$$

Now, by substituting $\lambda_{\mathcal{A}} = j\lambda_{\mathcal{L}}$, we obtain:

$$\begin{cases} \mathcal{L} v'' = j\lambda_{\mathcal{L}} v' \\ -\mathcal{L} v' = j\lambda_{\mathcal{L}} v'' \end{cases}$$

for which a possible solution is $[v_{\lambda_{\mathcal{L}}}, jv_{\lambda_{\mathcal{L}}}]^T$. The same argument holds for the conjugate eigenvalue $\bar{\lambda}_{\mathcal{A}} = -j\lambda_{\mathcal{L}}$ for which a possible solutions is $[v_{\lambda_{\mathcal{L}}}, -jv_{\lambda_{\mathcal{L}}}]^T$. \square

Remark 8.4.2 It is relevant to point out that a different choice of eigenvectors of \mathcal{A} corresponding to the null eigenvalue is the following:

$$v_{\lambda_{\mathcal{A}}} = [\mathbf{1} \quad \mathbf{0}]^T, \quad \bar{v}_{\lambda_{\mathcal{A}}} = [\mathbf{0} \quad \mathbf{1}]^T.$$

Indeed, this representation was used in (117) to prove part of the result given in Theorem 8.4.3, namely that the DC component of the trajectory of $x_i(t)$ corresponds to the average of the initial states. ■

By Theorem 8.4.1 it follows that *each* state of *each* agent follows an oscillating trajectory which is a linear combination of sinusoids oscillating at and only at frequencies corresponding to the eigenvalues of the network at time t , as detailed by Theorem 8.4.3. Note that, when a topology switch occurs a phase and module shift are experienced in frequency, while the state trajectory remains continuous for each agent.

In the following we assume that the m *distinct* eigenvalues of the Laplacian are labeled as follows: $0 = \lambda_1 < \lambda_2 < \dots < \lambda_m$.

Theorem 8.4.3 *Let us consider a system described by eq. (8.2) relative to a network whose graph \mathcal{G} is connected. Let $x(0) = x_0$ and $z(0) = z_0$ be the state initial conditions. Let $\delta(\cdot)$ be the Dirac's delta function. Let λ_j be an eigenvalue of the Laplacian matrix \mathcal{L} of graph \mathcal{G} and m the number of distinct eigenvalues. Let v_1 be the unitary norm eigenvector corresponding to $\lambda_1 = 0$, and $v_j^{(k)}$, $k = 1, \dots, \nu_j$ be the ν_j unitary norm eigenvectors associated to $\lambda_j > 0$.*

The module of the Fourier transform of the i -th state components $x_i(t)$ and $z_i(t)$, $i = \{1, \dots, n\}$, can be written as:

$$\begin{aligned} |\mathcal{F}[x_i(t)]| &= |X_i(f)| = a_{1,i} \delta(0) + \sum_{j=2}^m \frac{a_{j,i}}{2} \delta(f \pm \lambda_j/2\pi), \\ |\mathcal{F}[z_i(t)]| &= |Z_i(f)| = b_{1,i} \delta(0) + \sum_{j=2}^m \frac{b_{j,i}}{2} \delta(f \pm \lambda_j/2\pi), \end{aligned}$$

In addition, the coefficients $a_{j,i}$ and $b_{j,i}$ are given by:

– For $\lambda_1 = 0$ ($\nu_1 = 1$ since the graph is connected):

$$a_{1,i} = v_1(i) v_1^T x(0) = \frac{\mathbf{1}^T x(0)}{n}, \quad b_{1,i} = v_1(i) v_1^T z(0) = \frac{\mathbf{1}^T z(0)}{n}. \quad (8.4)$$

– For $\lambda_j > 0$ and $\nu_j \geq 1$:

$$a_{j,i} = b_{j,i} = \sqrt{\left[\sum_{k=1}^{\nu_j} \left(v_j^{(k)}(i) v_j^{(k)T} x(0) \right) \right]^2 + \left[\sum_{k=1}^{\nu_j} \left(v_j^{(k)}(i) v_j^{(k)T} z(0) \right) \right]^2}, \quad (8.5)$$

Proof:

The state trajectory of $x_i(t)$ is a linear combination of the system modes. Since \mathcal{A} is skew symmetric, and any skew symmetric matrix is a normal matrix, i.e. $\mathcal{A}^* \cdot \mathcal{A} = \mathcal{A} \cdot \mathcal{A}^*$, thanks to the Spectral Theorem it is always diagonalizable through a unitary matrix¹. Thus all the eigenvalues have geometric multiplicity equal to their algebraic multiplicity (or equivalently, unitary index). By Theorem 8.4.1 to each Laplacian eigenvalue λ_j it corresponds a couple of pure imaginary eigenvalues of \mathcal{A} equal to $\lambda_{\mathcal{A}}, \bar{\lambda}_{\mathcal{A}} = \pm j\lambda_j$. Therefore, for an agent i the state trajectory $x_i(t)$ is equal to:

$$x_i(t) = a_{1,i} + \sum_{j=2}^m a_{j,i} \sin(\lambda_j t + \phi_j),$$

that is, a dc component plus a linear combination of sinusoids whose amplitudes and phase shifts are function of the initial conditions and of the graph topology.

Now, we compute the coefficients of the module of the Fourier transform of $x_i(t)$. When referring to the eigenvalues and eigenvectors of \mathcal{L} , $\lambda_{\mathcal{L}_i}$ and $v_{\lambda_{\mathcal{L}_i}}$ for $i = 1, \dots, n$, we drop the subscripts \mathcal{L} and $\lambda_{\mathcal{L}}$, respectively, and refer to them as λ_i and v_i for $i = 1, \dots, n$ instead.

Since \mathcal{A} is a skew-symmetric matrix, it can be diagonalized by means of a unitary matrix V such that $\mathcal{A} = VDV^*$, where D is a diagonal matrix whose elements are arranged as $D = \text{diag}\{j\lambda_1, j\lambda_2, \dots, j\lambda_n, -j\lambda_1, -j\lambda_2, \dots, -j\lambda_n\}$, and V is a complex matrix whose columns are the eigenvectors of \mathcal{A} . Furthermore, applying Theorem 8.4.1, matrix V is rearranged to match the disposition of the eigenvalues of D as follows:

$$V = \begin{bmatrix} v_1 & v_2 & \dots & v_n & v_1 & v_2 & \dots & v_n \\ jv_1 & jv_2 & \dots & jv_n & -jv_1 & -jv_2 & \dots & -jv_n \end{bmatrix}.$$

In the following it is assumed that v_i , $i = 1, \dots, n$, are normalized eigenvectors such that $\|v_{\lambda_i}\| = 1$. To keep this notation we need to normalize the eigenvectors of \mathcal{A} such that $VV^* = I$, thus $\left\| \alpha \begin{bmatrix} v_i & jv_i \end{bmatrix}^T \right\| = 1$.

By simple manipulations we find $\alpha = \frac{1}{\sqrt{2}}$. The state trajectories of the system

¹A unitary matrix U is a complex matrix such that $U^*U = UU^* = I$, where U^* is the complex conjugate of U .

are captured by the matrix exponential which takes the following form in our case:

$$e^{At} = V e^{Dt} V^* = \frac{1}{2} \begin{bmatrix} v_1 & \dots & v_n & v_1 & \dots & v_n \\ jv_1 & \dots & jv_n & -jv_1 & \dots & -jv_n \end{bmatrix} \begin{bmatrix} e^{j\lambda_1 t} & & & & & \\ & \dots & & & & \\ & & e^{j\lambda_n t} & & & \\ & & & e^{-j\lambda_1 t} & & \\ & & & & \dots & \\ & & & & & e^{-j\lambda_n t} \end{bmatrix} \begin{bmatrix} v_1^T & -jv_1^T \\ \dots & \dots \\ v_n^T & -jv_n^T \\ v_1^T & jv_1^T \\ \dots & \dots \\ v_n^T & jv_n^T \end{bmatrix}.$$

It follows that the state trajectory x_i of agent i has the following form:

$$x_i(t) = \frac{1}{2} \begin{bmatrix} v_1(i) & v_2(i) & \dots & v_n(i) & v_1(i) & v_2(i) & \dots & v_n(i) \end{bmatrix} e^{Dt} V^* \begin{bmatrix} x(0) \\ z(0) \end{bmatrix},$$

and by highlighting with respect to a given eigenvalue λ_j we have:

$$x_i(t) = \frac{1}{2} \begin{bmatrix} \dots & v_j(i) & \dots & v_j(i) & \dots \end{bmatrix} \begin{bmatrix} \dots & & & & \\ & e^{j\lambda_j t} & & & \\ & & \dots & & \\ & & & e^{-j\lambda_j t} & \\ & & & & \dots \end{bmatrix} \begin{bmatrix} \dots \\ v_j^T x(0) - jv_j^T z(0) \\ \dots \\ v_j^T x(0) + jv_j^T z(0) \\ \dots \end{bmatrix}.$$

By some manipulations we find:

$$\begin{aligned} x_i(t) &= \sum_{j=1}^n \left[\frac{1}{2} v_j(i) e^{j\lambda_j t} (v_j^T x(0) - jv_j^T z(0)) + \frac{1}{2} v_j(i) e^{-j\lambda_j t} (v_j^T x(0) + jv_j^T z(0)) \right] \\ &= \sum_{j=1}^n \left[v_j(i) (\cos(\lambda_j t) v_j^T x(0) + \sin(\lambda_j t) v_j^T z(0)) \right] \end{aligned}$$

thus, according to the notation of Theorem 8.4.3, the coefficient $a_{j,i}$ associated to the eigenvalue λ_j for the i -th agent can be written as:

– For $\lambda_1 = 0$ and $v_1 = \mathbf{1}$:

$$a_{1,i} = v_1(i) v_1^T x(0) = \frac{1}{\sqrt{n}} \frac{\mathbf{1}^T x(0)}{\sqrt{n}} = \frac{\mathbf{1}^T x(0)}{n}$$

where $v_1 = \frac{1}{\sqrt{n}} \mathbf{1}$ is the unitary norm eigenvector associated to λ_1 (see Lemma 8.4.1 and the following Remark 8.4.2).

– For $\lambda_j > 0$ and $\nu_j \geq 1$:

$$a_{j,i} = \sqrt{\left[\sum_{k=1}^{\nu_j} \left(v_j^{(k)}(i) v_j^{(k)T} x(0) \right) \right]^2 + \left[\sum_{k=1}^{\nu_j} \left(v_j^{(k)}(i) v_j^{(k)T} z(0) \right) \right]^2}.$$

A similar reasoning can be repeated for the computation of the coefficients $b_{j,i}$'s.

In particular, the trajectory of $z_i(t)$ is:

$$z_i(t) = \frac{1}{2} \begin{bmatrix} \dots & jv_j(i) & \dots & -jv_j(i) & \dots \end{bmatrix} \begin{bmatrix} \dots & e^{j\lambda_j t} & & & \\ & & \dots & & \\ & & & e^{-j\lambda_j t} & \\ & & & & \dots \end{bmatrix} \begin{bmatrix} \dots & v_j^T x(0) - jv_j^T z(0) & \dots \\ \dots & \dots & \dots \\ \dots & v_j^T x(0) + jv_j^T z(0) & \dots \\ \dots & \dots & \dots \end{bmatrix}$$

and by some manipulations it follows:

$$\begin{aligned} z_i(t) &= \sum_{j=1}^n \left[\frac{1}{2} j v_j(i) e^{j\lambda_j t} (v_j^T x(0) - j v_j^T z(0)) - \frac{1}{2} j v_j(i) e^{-j\lambda_j t} (v_j^T x(0) + j v_j^T z(0)) \right] \\ &= \sum_{j=1}^n \left[\frac{1}{2} j v_j(i) \left(e^{j\lambda_j t} v_j^T x(0) - e^{-j\lambda_j t} v_j^T x(0) \right) + \frac{1}{2} v_j(i) \left(e^{j\lambda_j t} v_j^T z(0) + e^{-j\lambda_j t} v_j^T z(0) \right) \right] \\ &= \sum_{j=1}^n \left[-\frac{1}{2} v_j(i) 2 \sin(\lambda_j t) v_j^T x(0) + \frac{1}{2} v_j(i) 2 \cos(\lambda_j t) v_j^T z(0) \right] \\ &= \sum_{j=1}^n \left[v_j(i) \left(-\sin(\lambda_j t) v_j^T x(0) + \cos(\lambda_j t) v_j^T z(0) \right) \right]. \end{aligned}$$

Thus, according to the notation of Theorem 8.4.3, the coefficient $b_{j,i}$ associated to the eigenvalue λ_j for the i -th agent can be written as:

– For $\lambda_1 = 0$ and $\nu_1 = 1$:

$$b_{1,i} = v_1(i) v_1^T z(0) = \frac{1}{\sqrt{n}} \frac{\mathbf{1}^T z(0)}{\sqrt{n}} = \frac{\mathbf{1}^T z(0)}{n}.$$

– For $\lambda_j > 0$ and $\nu_j \geq 1$:

$$b_{j,i} = \sqrt{\left[\sum_{k=1}^{\nu_j} \left(v_j^{(k)}(i) v_j^{(k)T} x(0) \right) \right]^2 + \left[\sum_{k=1}^{\nu_j} \left(v_j^{(k)}(i) v_j^{(k)T} z(0) \right) \right]^2}.$$

□

The above theorem states the key result. In fact, it implies that each agent can efficiently and independently solve the eigenvalues estimation problem by simply using the *Fast Fourier Transform* (FFT) algorithm.

Note that, the correct estimation of the eigenvalues is guaranteed by having a dwell-time for the switching topology greater than the time window used to acquire enough samples for the FFT computation, as it will be detailed in Section 8.5.

This approach allows to detect in a decentralized way *changes* in the network topology, such as network disconnections, changing in the formation of the agents, loss of crucial links that decrease the network algebraic connectivity and so on.

Remark 8.4.4 *Two important remarks are now in order:*

- *The value of $x_i(t)$ can be seen as the output of the i -th agent. Now, if the system is not observable from the output $x_i(t)$ then some coefficients a_j can be null and thus the corresponding mode cannot be detected by agent i . This implies that the corresponding Laplacian eigenvalue λ_j cannot be estimated by agent i . The observability properties of system (8.2) are addressed in Theorem 8.4.6.*
- *Since we are looking at the modulus of the spectrum of $x_i(t)$, if an eigenvalue has algebraic multiplicity greater than one it will result in a single line of higher amplitude. The problem of how to estimate the multiplicity of the eigenvalues will be object of future research. ■*

Remark 8.4.5 *Theorem 8.4.3 highlights the usefulness of the proposed local interaction rule: not only the eigenvalues of the network can be easily identified but also the components of the corresponding eigenvectors relative to the agent are directly linked to the magnitude of the frequency peaks. A first direct consequence is that the proposed approach is an alternative solution to the consensus problem in that the agents cooperate in such a way that their DC component corresponds to the average of their initial conditions. ■*

Observability of the system modes

The observability and controllability of the graph Laplacian has been studied in (125, 126). Regarding observability, the authors of (125) have developed a

framework to study the observability of \mathcal{L} from a graph theoretical perspective. In particular, they have found out that equitable partitions are a relevant topological feature that affects observability. Furthermore, their result is relevant to the case in which we want to *build* a network that is observable.

If a network implements the local updating rule in eq. (8.1) the states of all the agents oscillate at the frequencies given by the eigenvalues of \mathcal{L} . If system (8.2) is not observable with respect to a given output matrix then its outputs oscillate only at the frequencies given by the observable modes of \mathcal{L} .

The following theorem enables us to conclude that the observability properties of system (8.2) are strictly related to the observability properties of the graph Laplacian, and so to the results concerning observability provided in (125).

Theorem 8.4.6 *Let \mathcal{A} be the matrix describing the group dynamics as in (8.3). Let C be a $n \times n$ matrix. Let*

$$\mathcal{A} = \begin{bmatrix} \mathbf{0} & \mathcal{L} \\ -\mathcal{L} & \mathbf{0} \end{bmatrix} \quad \text{and} \quad \hat{C} = \begin{bmatrix} C & \mathbf{0} \\ \mathbf{0} & C \end{bmatrix}.$$

(\mathcal{A}, \hat{C}) is observable iff (\mathcal{L}, C) is observable.

Proof: The observability matrix of the augmented systems is

$$\hat{\mathcal{O}} = \left[\hat{C}^T \quad (\hat{C}\mathcal{A})^T \quad (\hat{C}\mathcal{A}^2)^T \quad \vdots \quad (\hat{C}\mathcal{A}^{2n-1})^T \right]^T$$

and one can readily verify that for $k \in \mathbb{N}$ it holds

$$\hat{C}\mathcal{A}^{2k} = (-1)^k \begin{bmatrix} C\mathcal{L}^{2k} & \mathbf{0} \\ \mathbf{0} & C\mathcal{L}^{2k} \end{bmatrix} \quad \text{and} \quad \hat{C}\mathcal{A}^{2k+1} = (-1)^k \begin{bmatrix} \mathbf{0} & C\mathcal{L}^{2k+1} \\ -C\mathcal{L}^{2k+1} & \mathbf{0} \end{bmatrix}.$$

Hence by simple row permutation (and multiplication by -1 if required) it holds

$$\text{rank } \hat{\mathcal{O}} = \text{rank} \begin{bmatrix} \mathcal{O} & \mathbf{0} \\ \mathcal{O}\mathcal{L}^n & \mathbf{0} \\ \mathbf{0} & \mathcal{O} \\ \mathbf{0} & \mathcal{O}\mathcal{L}^n \end{bmatrix} = 2 \text{ rank } \mathcal{O}$$

where $\mathcal{O}(\mathcal{L}, C)$ is the observability matrix of (\mathcal{L}, C) .

Thus, it follows that (\mathcal{A}, \hat{C}) is observable iff (\mathcal{L}, C) is observable. \square

The above theorem points out that, even if the system is not observable from a single agent perspective, it will always be observable if matrix C is the identity matrix, i.e., if we consider all the information that agents locally retrieve.

Moreover, even if each agent cannot observe all the eigenvalues in a decentralized way, a specific consensus algorithm can be implemented on certain particularly significant eigenvalues, e.g., the largest one, or the second smallest one. This opens up to the scenario of consensus on the eigenvalues.

8.5 Implementation issues of the approach

The proposed interaction rule (8.2) is described as a marginally stable linear system since all the eigenvalues lie on the imaginary axis. The stability of the model of a physical system with eigenvalues exactly on the imaginary axis is not considered to be robust because even the slightest parameter uncertainties may render the system unstable. In our case there is no parameter uncertainty because system (8.2) is based on the Laplacian matrix whose elements depend only on the number of links between the agents and for any network topology system (8.2) is only marginally stable. Furthermore we point out that since no sensing/measurements are involved, no noise is generated from the application of the local interaction rule. The only possible noise effects come from quantization and communication link.

Estimation Accuracy

In this section, the estimation accuracy by means of FFT analysis is investigated. As a term of comparison the idea proposed in (122) is considered. In their work, the authors propose a decentralized estimation procedure that allows each agent to track the algebraic connectivity of a time-varying graph. In order to achieve that, the authors introduce a decentralized power iteration algorithm that enables each agent i to compute an estimate of the i -th component of the eigenvector corresponding to the second smallest eigenvalue of a weighted Laplacian matrix. The most important difference among the approach described in (122) and the solution proposed in this work is the convergence time of the algorithm. Indeed, the solution presented in (122) allows to achieve an asymptotic

tracking of the algebraic connectivity while our approach allows to compute an estimate in finite time. Furthermore, the estimation accuracy can be considered as a design parameter in our approach, while it is not characterized in (122). In particular, let us assume a certain accuracy Δf is required for the estimation of the algebraic connectivity, i.e., λ_2 . Furthermore, the highest possible frequency of the sinusoids is $f_c = \frac{\omega_c}{2\pi} < \frac{n}{\pi}$, therefore the sampling frequency can be chosen accordingly as $f_s = \frac{n}{\pi}$. At this point, by recalling the relationship which links the frequency resolution to the sampling frequency we obtain that the number samples required to achieve a given accuracy is $m = \frac{f_s}{\Delta f} = \frac{n}{\Delta f \pi}$. This implies that if the acquisition time to take one sample of the evolution of system (8.2) is T_s then to achieve the desired frequency resolution we need to sample (8.2) for $T = m T_s$ units of time. If the which by substituting m becomes $T = \frac{f_s}{\Delta f} T_s$.

Dwell Time

In this section, the dwell-time T_d , i.e., the minimum time between topology switching, required for the FFT analysis is investigated. Generally speaking, the FFT acquisition time T must be at least equal to the maximum period of the signal to be analyzed. In this context, since a linear combination of sinusoids is considered, the acquisition time $T = m T_s$, where m is the number of samples and T_s is the sampling time, must be at least equal to the longest period of one of the sinusoids, $T_c = \frac{2\pi}{\omega_c}$. Note that, by assuming the number of agents n to be available, an upper bound for the total number of samples required can be easily obtained by exploiting the Gershgorin disc theorem and by recalling that $\omega_c \leq 2 \Delta_{max}$ with $\Delta_{max} \leq n - 1$. This implies that $m \geq \frac{n}{\pi}$. Therefore, for any network topology on n agents, the dwell-time for topology switchings must be at least $T_d > m T_s$ or equivalently $T_d > \frac{n}{\Delta f \pi} T_s$, where T_s is the time required to obtain one sample of the network evolution from an agent. In order to be able to record a sufficient number of samples for the FFT analysis, $T_d > \frac{n}{\Delta f \pi} T_s$ is a sufficient condition, if a topology switching occurs before such dwell time then the spectrum of the two network topologies will be superimposed in the output of the FFT, this behavior is not completely undesirable in that it allows to actually detect that a topology change has taken place even if we do not know the actual

topology.

Simulations

In order to corroborate the mathematical results, simulations have been performed by exploiting the 4th Order Runge-Kutta Method (RK4) for the approximation of solutions of ordinary differential equations to simulate system (8.2).

Fig. 8.1 shows a possible embedding of a network topology involving 6 agents.

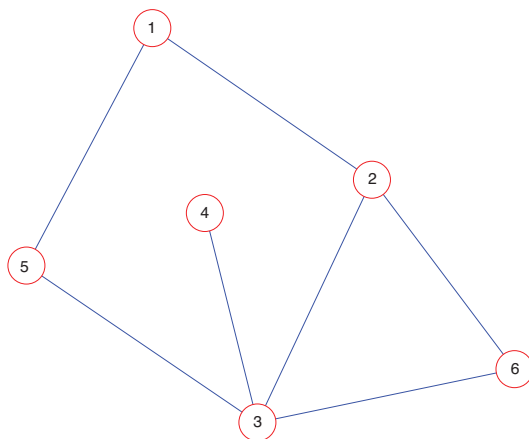


Figure 8.1: Network topology with 6 agents.

Fig. 8.2 shows the related spectrum observed by the node with $ID = 1$ with respect to the output associated to the state variable $x_1(t)$.

It can be noticed that the magnitude of the FFT shown in Fig. 8.2 has a dc component equal to zero. This is obtained according to Theorem 8.4.3 by having each agent i initializing its $x_i = 0$. Moreover, the nested box in Fig. 8.2, gives a more detailed view of the peaks. Indeed, these are exactly all the eigenvalues of the Laplacian matrix associated to this network topology (neglecting the null structural eigenvalue $\lambda_1 = 0$): $\sigma(L) = [0, 0.88, 1.45, 2.53, 3.86, 5.36]$. In addition, a leakage effect in the computed spectrum is experienced due to the non periodic nature of the block of the data recorded.

Fig. 8.3 shows a random network topology involving 20 agents. Fig. 8.4 shows the related spectrum observed by the node with $ID = 3$ with respect to the output associated to the state variable $x_3(t)$.

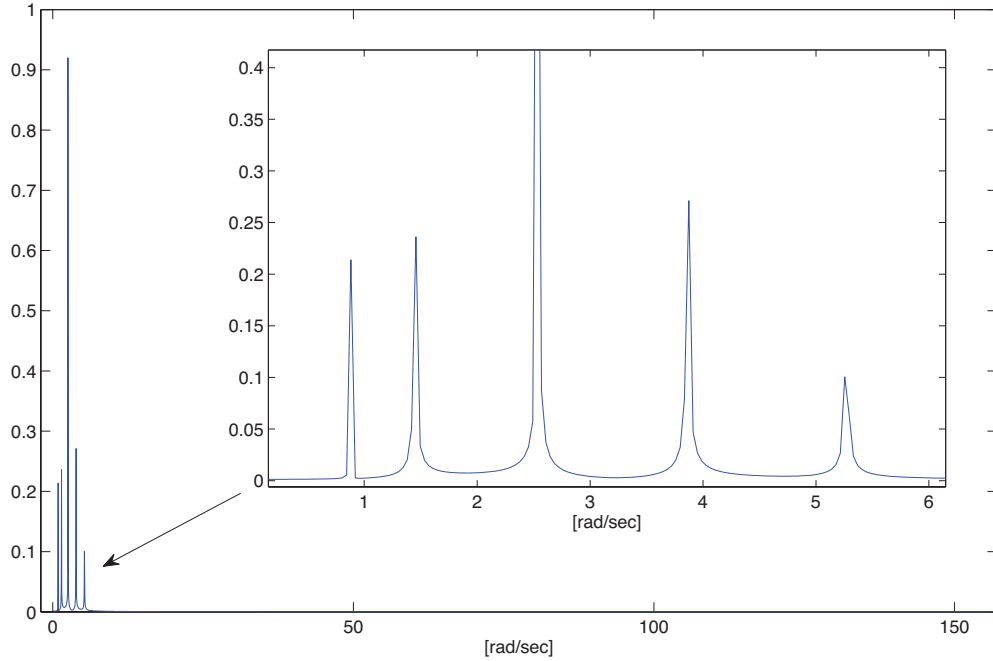


Figure 8.2: Spectrum of the dynamic matrix describing a network composed of 6 agents computed by the agent with $ID = 1$ with respect to the output associated to the state variable $x_1(t)$. Eigenvalues of the Laplacian matrix are: $\sigma(L) = [0, 0.88, 1.45, 2.53, 3.86, 5.36]$.

The nested box in Fig. 8.4, gives a more detailed view of the peaks highlighting the equivalence with the eigenvalues of the Laplacian matrix associated to this network topology (neglecting the structural eigenvalue $\lambda_1 = 0$):

$$\sigma(\mathcal{L}) = [0, 1.63, 2.11, 2.75, 3.62, 4.09, 4.26, 4.75, 5.10, 5.75, 6.52, 6.74, 7.37, 7.63, 8.30, 8.39, 9.32, 9.69, 10.75, 11.18].$$

So far, two possible network topologies have been considered and related spectrums have been analyzed. In the following, a switching topology for a network composed of 6 agents is investigated. Fig. 8.5 depicts the topology variation over time. In detail, starting from the topology depicted in Fig. 8.5-a, a few connections among agents are dropped and a few new connections are created over time in such a way to reach the ring topology given in Fig. 8.5-d.

Fig. 8.8 shows how the spectrum varies over time according to the topology variation, with respect to the output $x_1(t)$ observed by the agent with $ID = 1$. In

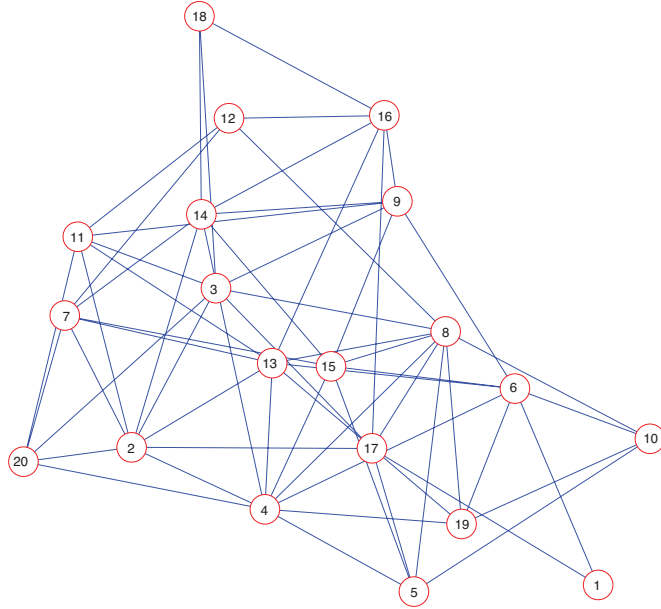


Figure 8.3: Network with 20 agents.

particular, besides the dc component which is null according to Theorem 8.4.3, 5 peaks are observed in the starting topology (Fig. 8.6-a). After the first and second variations happen, it can be observed how the number of peaks remains equal to 5 while their location changes in frequency according to the variation of the eigenvalues associated to the related Laplacian graphs (Fig. 8.6-b and Fig. 8.6-c). A different scenario is obtained once the ring topology is reached as only 3 peaks are recognized (Fig. 8.8-d). Indeed, the related Laplacian graph has two eigenvalues with algebraic multiplicity equal to 2, namely $\lambda_1 = 1$ and $\lambda_2 = 3$.

Finally, a transition step should be noticed in the spectrogram for each topology variation where peaks are not clearly defined. Indeed, this can be explained by the fact that during a topology transition the block of data used for the FFT algorithm is composed of data related to two different topologies. This information, as previously stated, could be effectively used as a reliable indicator to recognize when a variation happens.

Figure 9.2 describes a network of vehicles whose formation changes over time. In particular, on the left side the topology variation with respect to time is given, while on the right side the evolution of the average value of the agents and the estimated average value from the DC component of the FFT.

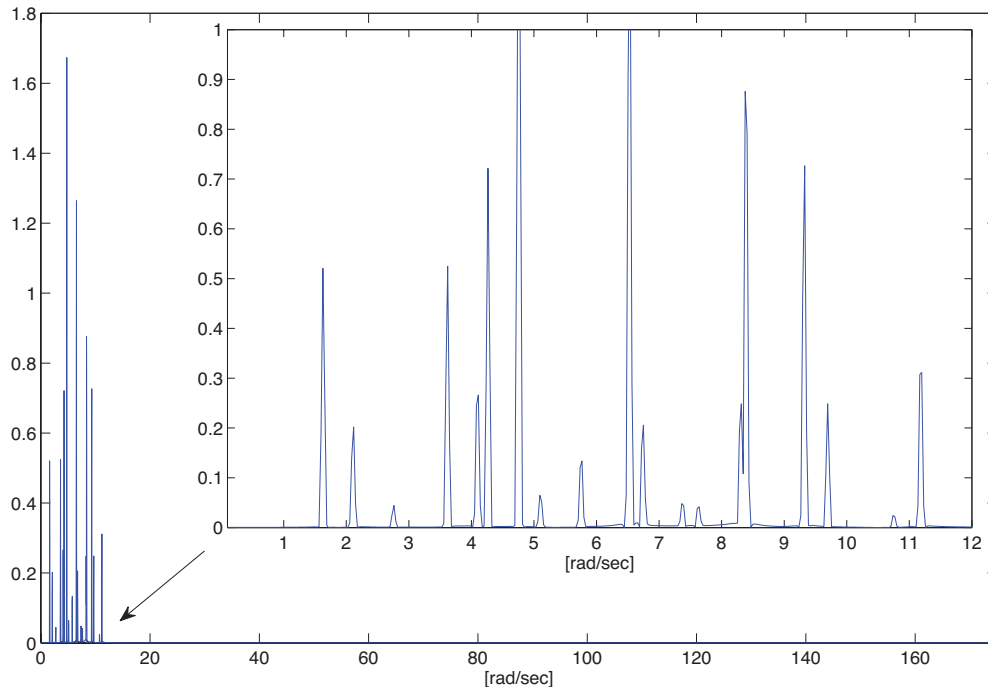


Figure 8.4: Spectrum of the dynamic matrix describing the first network topology composed of 20 agents computed by the agent with $ID = 3$ with respect to the output associated to the state variable $x_3(t)$.

Figure 8.8 shows the spectrogram of the time varying topology shown in Figure 9.2 computed by the agent i with respect to its state variable $x_i(t)$. Note that, an additive zero mean gaussian noise was added to the state updating rule given by eq. (8.1).

8.6 Conclusions

In this chapter a novel decentralized algorithm to estimate the Laplacian spectrum of a network has been proposed. Each agent interacts with its neighbors so that its state oscillates at the frequencies corresponding to the eigenvalues of the network topology. In this way the estimation problem is reduced to a problem of signal processing solvable by using the FFT algorithm.

A theoretical analysis of the proposed techniques along with numerical simulations has been provided. Note that this technique could be extended to the

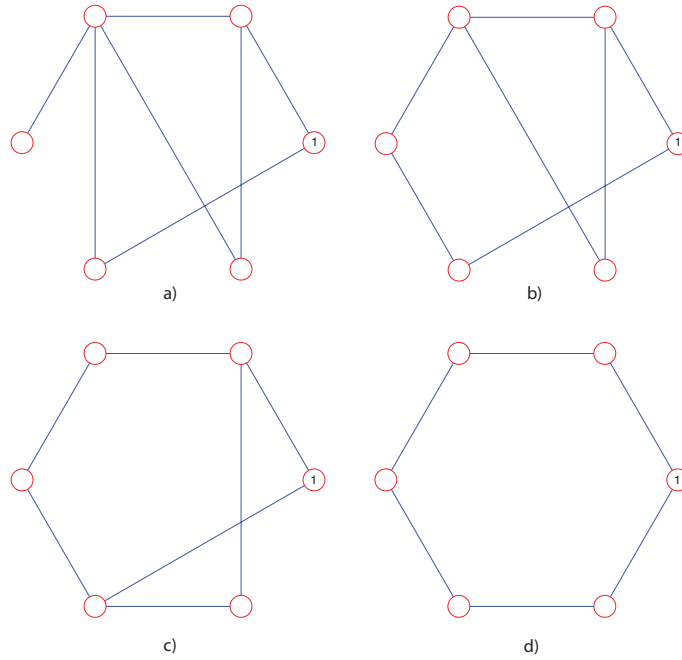


Figure 8.5: Topology variation with respect to time for a network composed of 6 agents.

case of a weighted graph.

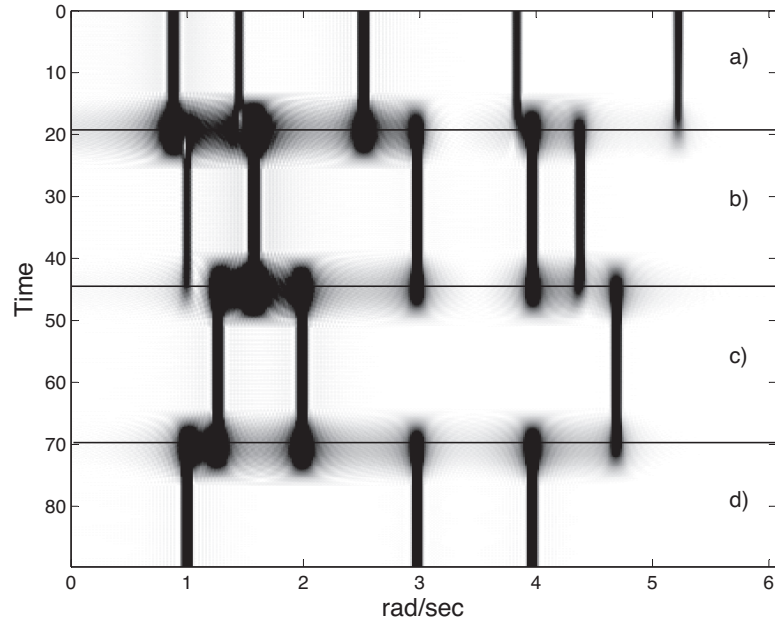


Figure 8.6: Spectrogram of the time varying topology shown in Fig. 9.2 computed by the agent with $ID = 1$ with respect to the output associated to the state variable $x_1(t)$.

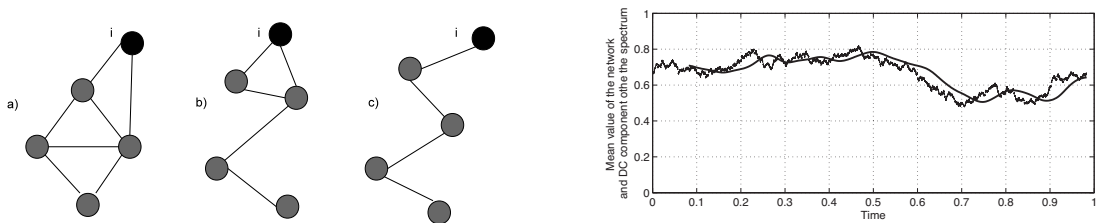


Figure 8.7: Left: Topology variation with respect to time for a network composed of 5 agents. Right: Evolution of the average value of the agents and the estimated average value from the DC component of the FFT.

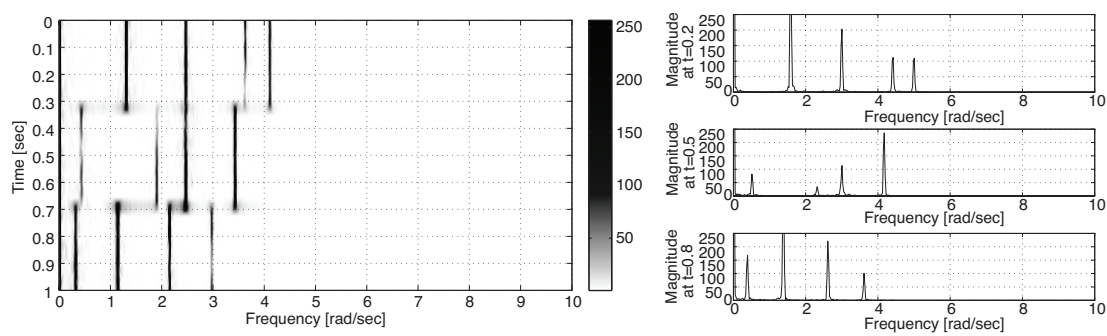


Figure 8.8: Spectrogram of the time varying topology shown in Fig. 9.2 computed the agent with $ID = 1$ with respect to the output associated to the state variable $x_1(t)$.

Chapter 9

Spectrum based Controllability, observability and topology estimation

9.1 Introduction

Multi-agent systems composed by networks of unmanned mobile vehicles are envisioned to perform the most various tasks in the near future. The design of control algorithms for such systems poses several challenges to achieve robustness and scalability. So far such properties are expected to be achieved by decentralized control algorithms that make locally use of available information (7, 16, 24, 50).

A significant example of a multi-agent system is one involving agents with simple integrator dynamics under Laplacian feedback (24). While the model of the agents' dynamics is clearly oversimplified, the network model has just the right complexity to capture several relevant features of a networked system linked to the topology of the network. Furthermore such model is widely accepted to be a good starting point in modeling leader-follower networks of mobile vehicles (127) with the aim of allowing a single pilot to control a multitude of mobile vehicles with limited available information.

In this chapter we build on the idea presented in chapter 8 to use the information about the spectrum of the network to infer in a decentralized fashion properties such as controllability, observability and, more in general, its topology.

The quest for linking controllability and observability of a system under Laplacian feedback has been previously undertaken by (125, 126, 128, 129). In (129) a graph theoretic sufficient condition for controllability has been developed. It turned out that controllability and, by duality, observability depend on the existence of external equitable partitions on the graph representing the network. In this work we show how this graph theoretical characterization is linked to some algebraic facts related to the spectrum of the network. Furthermore our condition is *locally checkable online*.

Controllability and observability are useful properties if the network topology is known. Thankfully another application of the information about the spectrum of a network is the estimation of its topology. In general, the spectrum is not necessarily a unique identifier for a given topology. Moreover, in multi-agent systems we may be interested in the subproblem of estimating when a particular topology known a priori has been achieved. The target topology in which the agents are supposed to be in their nominal state of operations can be built so that it is identifiable by its spectrum. A strong application of this information is the enabling of a simple Luenberger observer to estimate correctly the relative position of each agent in the network with respect to the leader in absence of communication, GPS or common reference frames. In this chapter we show how line and lattice formations composed by a convoy of n agents can be identified by their corresponding spectrum.

We point out that the theory presented in this chapter can be easily extended to heterogeneous networks where a different weight is associated to each link.

This chapter is structured as follows:

- In Section 9.2 we provide some background on leader-follower networks.
- In Section 9.3 we present a decentralized method to check for observability and controllability.
- In Section 9.4 we propose the use of the spectrum of a graph for formation identification and provide an example of application of involving a convoy of vehicles.

9.2 Background on leader-follower networks

In multi-agents systems, it is common to let the nodes of a graph represent the agents, and to let the arcs in the graph represent the inter-agent communication links. In fact, this interaction graph plays a central role in representing the information flow among the agents, and in defining the properties of the system.

Let the undirected graph G be given by the pair (V, \mathcal{E}) , where $V = \{1, \dots, n\}$ is a set of n vertices, and \mathcal{E} is a set of edges. Two nodes j and k are neighbors if $(j, k) \in \mathcal{E}$, and the set of the neighbors of the node j is defined as $N_j = \{k : (j, k) \in \mathcal{E}\}$. The degree of a node is given by the number of its neighbors, and a graph G is *connected* if there is a path between any pair of distinct nodes, where a path $i_0 i_1 \dots i_S$ is a finite sequence of nodes such that $i_{k-1} \in N_{i_k}$ with $k = 1, 3 \dots S$.

We let the state of each node, x_i , be a scalar. (This does not affect the generality of the derived results.) The standard, consensus algorithm consists in each agent performing the following state update law

$$\dot{x}_i(t) = \sum_{j \in N_i} (x_j(t) - x_i(t)), \quad (9.1)$$

or equivalently $\dot{x}(t) = -\mathcal{L}x(t)$, where $x(t)$ is the vector with the states of all nodes at time t , and \mathcal{L} is the graph Laplacian. \mathcal{L} can be obtained as $\mathcal{J}\mathcal{J}^T$, where $\mathcal{J} \in \mathbb{R}^{n \times p}$, (p being the number of edges), is the *incidence matrix* of the graph, defined as

$$[\mathcal{J}]_{kl} = \begin{cases} 1 & \text{if node } k \text{ is the head of the edge } l \\ -1 & \text{if node } k \text{ is the tail of the edge } l \\ 0 & \text{otherwise,} \end{cases}$$

given an arbitrary orientation of the edges.

Under some connectivity conditions, the consensus algorithm (9.1) is guaranteed to converge, i.e. $\lim_{t \rightarrow +\infty} x_i(t) = g$, $i \in \{1, \dots, n\}$, where g is a constant depending on \mathcal{L} , and on the initial conditions $x_0 = x(0)$. See for example (130, 131, 132).

As in (133, 134, 135), we imagine that a subset of the agents have superior sensing, computation, or communication abilities. We thus partition the node set V into a leader set L of cardinality n_l , and a follower set F of cardinality n_f , so that $L \cap F = \emptyset$ and $L \cup F = V$.

Leaders differ in their state update law in that they can arbitrarily update their positions, while the followers execute the agreement procedure (9.1), and are therefore controlled by the leaders.

Under the assumption (without loss of generality) that the first n_f agents are followers, and the last $n_l = n - n_f$ are leaders, the introduction of leaders in the network induces a partition of the incidence matrix \mathcal{J} as

$$\mathcal{J} = \begin{bmatrix} \mathcal{J}_f \\ \mathcal{J}_l \end{bmatrix},$$

where $\mathcal{J}_f \in \mathbb{R}^{n_f \times p}$, $\mathcal{J}_l \in \mathbb{R}^{n_l \times p}$, and the subscripts f and l denote respectively the affiliation with the leaders and followers set. As a result, the graph Laplacian \mathcal{L} becomes

$$\mathcal{L} = \left[\begin{array}{c|c} \mathcal{L}_f & \mathcal{L}_{fl} \\ \hline \mathcal{L}_{fl}^T & \mathcal{L}_l \end{array} \right], \quad (9.2)$$

with $\mathcal{L}_f = \mathcal{J}_f \mathcal{J}_f^T \in \mathbb{R}^{n_f \times n_f}$, $\mathcal{L}_l = \mathcal{J}_l \mathcal{J}_l^T \in \mathbb{R}^{n_l \times n_l}$ and $\mathcal{L}_{fl} = \mathcal{J}_f \mathcal{J}_l^T \in \mathbb{R}^{n_f \times n_l}$.

The system we now consider is the controlled agreement dynamics, in which agents evolve through the Laplacian-based dynamics

$$\begin{cases} \dot{x} = -\mathcal{L}_f x - \mathcal{L}_{fl} x_l \\ \dot{x}_l = u \\ y = -\mathcal{L}_{fl}^T x - \mathcal{L}_l x_l \end{cases} \quad (9.3)$$

where x is the state vectors of the followers, and $u(t)$ denotes the exogenous control signal dictated by the leaders.

In the proposed approach all the agents (both followers and the leader) simulate system (8.2) and estimate the eigenvalues of \mathcal{L} as described in chapter 8. We now recall that the leader-followers network evolves according to

$$\begin{cases} \dot{x} = -\mathcal{L}_f x - \mathcal{L}_{fl} x_l \\ \dot{x}_l = u(t) \end{cases} \quad (9.4)$$

The leader has full access to the state of its neighbors and as such it is able to estimate

$$y = -\mathcal{L}_{fl}^T x - \mathcal{L}_l x_l \quad (9.5)$$

It follows that if the leader applies the following feedback control law,

$$u(t) = -\mathcal{L}_{f_l}^T x - \mathcal{L}_l x_l + \hat{u}(t) \quad (9.6)$$

including the state of the leader with the others, the networked system can be described by

$$\begin{cases} \begin{bmatrix} \dot{x} \\ \dot{x}_l \end{bmatrix} = -\mathcal{L} \begin{bmatrix} x \\ x_l \end{bmatrix} + Bu \\ y = C \begin{bmatrix} x \\ x_l \end{bmatrix} \end{cases} \quad (9.7)$$

where $C = B^T = [0, \dots, 0, 1]$.

9.3 Spectrum based decentralized check for observability and controllability

In this section we present a method for the decentralized online verification of observability and controllability in a multi-agent system. In the following it is assumed that the agents execute algorithm 8.2 and thus each agent estimates the eigenvalues (without multiplicity) observable from its position by taking only its own state trajectory as output. The basic idea is to exploit the properties of algorithm (8.2) to locally estimate the spectrum of the network and then link this information to check for observability and controllability. Such link is made possible by the fact that the modes of system (8.2) are observable if and only if the modes of system (9.7) are observable.

We now provide some basic helpful facts of linear system theory.

Lemma 9.3.1 *System (9.4) is controllable iff system (9.7) is controllable.*

Proof:

System (9.4) differs from system (9.7) in that the leader applies the following feedback control law

$$u(t) = -\mathcal{L}_{f_l}^T x - \mathcal{L}_l x_l + \hat{u}(t),$$

where $\hat{u}(t)$ is an input with the same dimensions as $u(t)$. If the system is controllable with such feedback it is controllable also with $u(t) = \hat{u}(t)$ since the input enters only in the row corresponding to x_l . Necessity comes from the fact that if system (9.7) is not controllable from $\hat{u}(t)$ then it is not controllable from any input entering in the row of x_l and thus also

$$\bar{u}(t) = \hat{u}(t) + \mathcal{L}_{f_l}^T x + \mathcal{L}_l x_l = u(t),$$

proving the statement. □

Lemma 9.3.2 *If the Laplacian matrix \mathcal{L} of graph \mathcal{G} has eigenvalues with multiplicity greater than one, then system (9.7) is not observable/controllable.*

Proof:

See (136) chapter 9.5. □

In the following theorem a sufficient and necessary condition for observability and controllability verification is given. Such condition involves only the local information available to agent i if the total number of agents n is known.

Theorem 9.3.3 *Let the network of agents be represented by a connected graph \mathcal{G} . Assume each agent simulates system (8.2) to estimate the Laplacian eigenvalues by applying the FFT algorithm to its state trajectory. Let agent i know the total number of agents n connected to the network. Then the network described by*

$$\begin{cases} \dot{x} = -\mathcal{L}_f x + \mathcal{L}_{f_l} u \\ y = \mathcal{L}_{f_l}^T x \end{cases} \quad (9.8)$$

is observable and controllable from agent i iff agent i observes n distinct eigenvalues,.

Proof:

- *Sufficiency:*

Assume agent i observes n modes of system (8.2) and they are distinct, then by taking as output the matrix $C = [0, \dots, 1, 0, \dots]$ with 1 in the i -th element, we have that observability matrix (C, \mathcal{L}) is full rank due to theorem 8.4.6. Due to lemma 9.3.1 if system (9.7) is controllable so is system (9.8). Furthermore since system (9.8) is symmetric and $C = B^T$, by duality the system is also controllable.

- *Necessity:*

Assume agent i estimates n distinct eigenvalues, assume system (8.2) is initialized with an initial condition not orthogonal to any of its eigenvector. If system (9.8) is not observable, then the observability matrix (C, \mathcal{L}) must be rank deficient and so has to be the observability matrix for system (8.2). It follows that if system (8.2) is not observable, then by definition the number of observable modes must be less than n which is a contradiction. Furthermore observability of system (9.7) is a necessary condition for the observability of system (9.8), the same goes for controllability. \square

The above theorem allows the agents to estimate in a decentralized fashion some relevant properties of the network if the number of agents is known. Note that the necessary condition holds only if system (8.2) is initialized with a proper initial condition so that all the system modes are excited. Now suppose that the total number of agents is not known and that the actual network is eventually not controllable nor observable. We are interested in finding the dimension of the controllable/observable subspace from any given agent. The following theorem characterizes the dimension of the controllable/observable subspace as function of the number of observable eigenvalues estimated by algorithm 8.2.

Theorem 9.3.4 *Assume each agent estimates the eigenvalues of system (8.2), by applying the FFT algorithm to its state trajectory. Assume agent i estimates a number of distinct eigenvalues m_i .*

The dimension of the controllability/observability subspace from agent i is equal to m_i .

Proof:

Assume agent i observes m_i eigenvalues executing algorithm 8.2. Thanks to theorem 8.4.6 we have that

$$\text{rank}(\mathcal{O}(\mathcal{A}, \hat{C})) = 2\text{rank}(\mathcal{O}(\mathcal{L}, C)).$$

Since the eigenvalues of system \mathcal{A} are purely imaginary, pairwise conjugate and equal to the eigenvalues of \mathcal{L} in modulus, we have

$$\text{rank}(\mathcal{O}(\mathcal{L}, C)) = m_i.$$

□

Remark 9.3.5 Theorem 9.3.4 holds if system (8.2) is initialized with a proper initial condition so that each system mode is excited. In the case such condition cannot be guaranteed, then the dimension of the controllability/observability subspace from agent i is clearly greater than or equal to m_i . ■

9.4 Spectrum based Formation Identification

The idea of estimating topological features of a graph from its spectrum has been around for quite some time in algebraic graph theory. Unfortunately it has been shown that the spectrum of a graph is not a unique identifier for its topology. In other words, if two graphs are identical except for a relabeling of their nodes then necessarily the two spectra are identical. On the other hand there exists several graphs which are co-spectral with many others (137, 138, 139). In this section we focus on the practical uses of this notion for the identification of regular structures such as formations of multi-agent systems.

A vast literature that deals with achieving some desired formation, e.g. (7, 140, 141, 142, 143), in a multi-agent system possibly in a decentralized fashion exists. A relevant issue in such decentralized approaches is to understand when such formation has been actually achieved so that the agents can switch mode of operation to something else.

It is clear that if the achievement of a formation could be linked directly to the spectrum of its topology then algorithm 8.2 could provide an instance of solution to such problem.

A relevant class of graph topologies that serve our cause are those structured graphs whose eigenvalues are known analytically as function of the number of nodes.

The first of such graphs is the line graph, or path P_n of n agents whose eigenvalues are

$$\lambda(P_n) = 4 \sin\left(\frac{\pi i}{2n}\right)^2, \quad \forall i = 0, \dots, n-1. \quad (9.9)$$

This fact is relevant to practical applications in that the line graph is both controllable and observable for leader-follower networks. Furthermore it has obvious applications in the control of convoys of ground vehicles.

Since the cartesian product of graphs has eigenvalues equal to any combination of summation of the eigenvalues of the original graphs (144), we have that the $n \times m$ grid has eigenvalues given by

$$\lambda(G_{n \times m}) = 4 \sin\left(\frac{\pi i}{2n}\right)^2 + 4 \sin\left(\frac{\pi j}{2m}\right)^2, \quad \forall i, j = 0, \dots, n-1.$$

The grid graph has significant applications in the coverage problem for both multi-agents systems and sensor networks.

Consider a simpler application in which agents are organizing as a convoy for moving toward a goal. Suppose that the leader knows the number of agents of the network and the desired topology which is determined by the eigenvalues of the Laplacian Matrix. Furthermore, suppose that each agent is provided with a decentralized controller which is able to chose its neighbors in order to reach the desired topology.

Starting from the initial point and structure of Figure 9.1(a), the communication links among nodes are changing (Figure 9.1(b)) to the final structure of Figure 9.1(c).

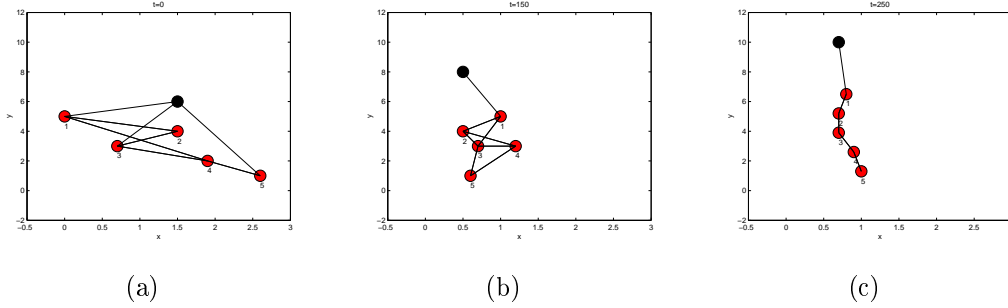


Figure 9.1: The initial structure of the convoy (a) and its modifications (b) toward the desired topology (c).

Figure 9.2 shows the evolution of the eigenvalues of the Laplacian matrix associated with networks of Figure 9.1(a), 9.1(b), 9.1(c). For every t it reports the Fast Fourier Transform (FFT) to a sufficiently long time window (of size T_w) of the trajectory of the state of the system (8.2) in the interval $[t - T_w, t]$ which is composed by a linear combination of sinusoids with frequencies corresponding to the Laplacian eigenvalues of the network. It is clear that, since the window is sliding, we are able to capture the eigenvalues of the Laplacian matrix associated to the network Figure 9.1(c) $\forall t \in [0, T_f]$ with all their modification to the final set-up. In particular from Figure 9.2, we can see that the topology in Figure 9.1(a) is not completely controllable and observable from the leader. Since its has eigenvalues located in $\lambda(\mathcal{G}_1) = [0, 1.4, 3, 3, 3, 5.5]$ the lack in the observability and controllability is due to the multiplicity of the eigenvalues 3. Indeed, the topology in Figure 9.1(b) is completely controllable and observable from the leader since all the eigenvalues of this structure are distinct. As it is not a line-graph its spectrum cannot be calculated analytically and it results $\lambda(\mathcal{G}_2) = [0, 0.7, 2.1, 3.4, 4.5, 5.1]$. At last, Figure 9.2 shows that the network in Figure 9.1(c) is completely controllable and observable. Since it is a line-graph its eigenvalues can be calculated according to (9.9) and they result $\lambda(\mathcal{G}_3) = [0, 0.2, 1, 2, 3, 3.7]$.

It is clear that this context emphasize the importance of the proposed method: by executing the decentralized check all agents are able to investigate about the eigenvalues of the network 9.2 and to settle whether the network is changed and whether the actual configuration is the desired one, for example observable. Only in the latter case, the leader, from which the network is completely observable, is

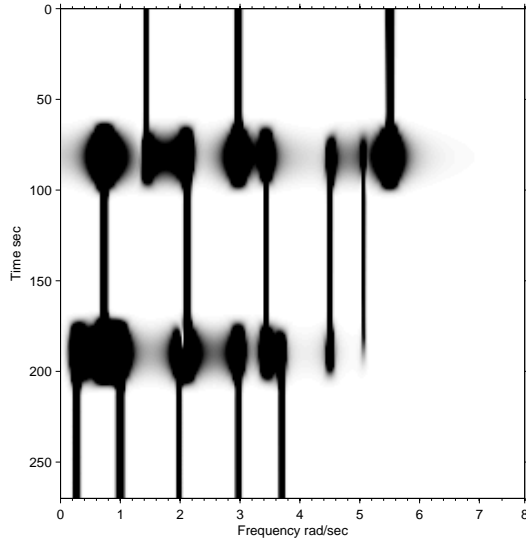


Figure 9.2: Spectrogram of the switching topology in figure 9.1(a),9.1(b),9.1(c) obtained by algorithm 8.2.

interested in reconstruct the connection scheme through which it is able to know all information regarding the other node in the network.

9.5 Conclusions

In this chapter we proposed a decentralized method for online checking of controllability and observability of a network of single integrators with Laplacian feedback under the assumption of unknown network topology. The method exploits the knowledge of the eigenvalues of the linear dynamics made available by the algorithm proposed in (117). We proposed the use of the spectrum of the network of a multi-agent system to identify when a desired formation has been achieved. We introduced examples of formations whose eigenvalues are known analytically and can be identified through such method.

Chapter 10

Conclusions

In this thesis several algorithms that address consensus related problems for several different applications have been presented. They represent improvements with respect to the state of the art in the exploitation of emergent behavior to solve problems arising in the multi-agent systems. The several contributions consists in:

- The algorithms for *discrete consensus* presented in chapter 3 and 4 to address particular quantization issues of consensus algorithms with assumptions borrowed from load balancing problems and applications to sensor networks. These algorithms are superior in that address problems statements with more general assumptions and improvements on the convergence properties respect to the state of the art
- The algorithm presented in **Chapter 5** is the first known to provide a solution to the consensus on the average problem on arbitrary directed graphs for real valued scalar agents.
- The algorithms presented in **Chapter 6** are the first to provide a method for agreement toward a common point in space in absence of a common reference frame and pairwise asynchronous communications with agents able only to sense the distance between themselves and their neighbors and the direction in which they see their neighbors with respect to their local reference frame.

- The fault diagnosis and recovery algorithms presented in **Chapter 7** are the first to address fault diagnosis for sensor networks implementing the consensus algorithm with the assumption of unknown network topology and only local state information in a decentralized way.
- The algorithm presented in **Chapter 8** is a significant improvement in the decentralized estimation of the Laplacian spectrum of a network in that it allows to estimate all the eigenvalues of the Laplacian matrix by means of the standard FFT algorithm in finite time. The state of the art consisted in the use of several algorithms to first estimate asymptotically all the eigenvectors and then all the corresponding eigenvalues separately. Furthermore in **Chapter 9** such algorithm is exploited to infer properties of leader-follower networks such as controllability and observability of a leader-follower network. Finally the use of the Laplacian spectrum of the network of a multi-agent system is proposed as tool to identify when a desired formation is achieved.

The research problems presented in this thesis have left many open problems and paved the way for novel research directions in multi-agent systems.

- The improvement of the convergence properties of the algorithms for *discrete consensus* may possibly be extended to more general graph structures such as trees instead of hamiltonian graphs.
- The algorithm presented in **Chapter 5** for consensus on the average problem on arbitrary directed graphs presents great challenges in the proof of its convergence properties. In particular it is an instance of randomized algorithm which almost surely converge in probability but that does not satisfy any of the common properties exploited in the study of stability switched linear systems.
- The algorithms presented in **Chapter 6** for agreement in absence of common reference frame solve the problem in a 2-D space if the network is connected. On the other hand for the d -D case network connectivity is merely necessary but not sufficient. The characterization of all the graph topologies that allow convergence would be a great advance, especially for the 3-D case.

- The fault diagnosis and recovery algorithms presented in **Chapter 7** perform the best on static network topologies. Extensions to the switching topology case would be challenging.
- The algorithm presented in **Chapter 8** allows the Laplacian spectrum of a network to be known efficiently by the agents, thus further ideas on how to exploit this information, of which some were presented in chapter 9, are expected to be proposed, especially regarding the inferring of topological properties of the network.

List of papers by the author

Journal articles

1. N. Orani, A. Pisano, M. Franceschelli, A. Giua, E. Usai "Robust reconstruction of the discrete state for a class of nonlinear uncertain switched systems" *Nonlinear Analysis: Hybrid Systems*, accepted, to appear 2011 (available online).
2. M.Franceschelli, A. Giua, C. Seatzu "Distributed Averaging in Sensor Networks Based on Broadcast Gossip Algorithms" *IEEE Sensor Journal*, Special Issue On Cognitive Sensor Networks, accepted, to appear 2011.
3. M.Franceschelli, A. Giua, C. Seatzu "Decentralized Estimation of Laplacian Eigenvalues in Multi-Agent Systems" *IEEE Transactions on Automatic Control*, submitted, (second revision)
4. M.Franceschelli, A. Giua, C. Seatzu "Quantized Consensus in Hamiltonian Graphs" *Automatica*, conditionally accepted (third revision)
5. M.Franceschelli, A. Giua, C. Seatzu "A gossip-based algorithm for discrete consensus over heterogeneous networks" *IEEE Transactions on Automatic Control*, vol. 55, n 5, May 2010, page 1244-1249.

Conference Proceedings with acceptance based on submission of full paper and peer-review.

1. M. Franceschelli, S. Martini, M. Egerstedt, A. Bicchi, A. Giua "Observability and Controllability Detection of Multi-Agent Systems through Decentralized Laplacian Spectrum Estimation" 49th IEEE Conference on Decision and Control, Atlanta,US, Dec 2010.

2. M. Franceschelli, A. Gasparri "On Agreement Problems with Gossip Algorithms in absence of common reference frames" IEEE International Conference on Robotics and Automation, Anchorage, Alaska, USA, May 4 - 8, 2010.
3. A. Gasparri, M. Franceschelli, G. Ulivi, A. Giua "On the stabilization of heterogeneous multi-agent robotic systems" IEEE International Conference on Robotics and Automation, Anchorage, Alaska, USA, May 4 - 8, 2010.
4. M. Franceschelli, A. Gasparri, A. Giua, C. Seatzu "Decentralized Laplacian Eigenvalues Estimation for Networked Multi-Agent Systems" 48th IEEE Conference on Decision and Control, Shanghai, China, Dec 2009.
5. N. Orani, A. Pisano, M. Franceschelli, A. Giua, E. Usai "Robust reconstruction of the discrete state for a class of nonlinear uncertain switched systems" 3rd IFAC Conference on Analysis and Design of Hybrid Systems, Zaragoza, Spain, Sep. 2009.
6. M. Franceschelli, A. Giua, C. Seatzu "Decentralized Fault Diagnosis for Sensor Networks" 5th Annual IEEE Conference on Automation Science and Engineering, Bangalore, India, Aug. 2009.
7. M. Franceschelli, A. Giua, C. Seatzu "Consensus on the Average on Arbitrary Strongly Connected Digraphs Based on Broadcast Gossip Algorithms" 1st IFAC Workshop on Estimation and Control of Networked Systems, Venice, Italy, Sep. 2009.
8. M. Franceschelli, A. Giua, C. Seatzu "Hamiltonian Quantized Gossip" IEEE Multi-conference on Systems and Control, St. Petersburg, Russia, July 2009.
9. M. Franceschelli, M. Egerstedt, A. Giua, C. Mahulea, "Constrained Invariant Motions for Networked Control Systems" American Control Conference, St. Louis, Missouri, USA, June 2009.
10. M. Franceschelli, A. Gasparri, "Decentralized Centroid Estimation for Multi-Agent Systems in Absence of Any Global Reference Frame" American Con-

trol Conference, St. Louis, Missouri, USA, June 2009. (Best Presentation in Session Award).

11. M. Franceschelli, A. Giua, C. Seatzu, "Load balancing over heterogeneous networks with gossip-based algorithms" American Control Conference, St. Louis, Missouri, USA, June 2009.
12. M. Franceschelli, M. Egerstedt, A. Giua, "Motion probes for fault detection and recovery in networked control systems" American Control Conference, Seattle, WA, USA, June 2008.
13. M. Franceschelli, A. Giua, C. Seatzu, "Load Balancing on Networks with Gossip-Based Distributed Algorithms" 46th IEEE Conf. on Decision and Control , New Orleans, LA, USA, December 2007.

Appendix A

Appendix

A.1 Algebraic graph theory

In this thesis we make extensive use of notation from algebraic graph theory. In the description of a multi-agent system we usually describe the pattern of couplings between the agents with a graph $G = \{V, E\}$ where $V = \{1, \dots, n\}$ is the set of n nodes or vertices that represent the agents. In this dissertation "nodes" and "agents" are used as synonyms we referring to a graph. For convenience the generic vertex i may sometime be referred to as v_i . The set of edges $E \subseteq \{V \times V\}$ represents the existence of an interaction between any given couple of nodes. If not otherwise stated, graph G is considered to be a directed graph (digraph in short), i.e., to each edge (i, j) we associate a direction, we call *head* of the edge node i and *tail* node j , finally we say that edge (i, j) , which sometime is referred as $e_{i,j}$ in short, goes from node j to node i .

A *loop* is an edge whose endpoints are the same, in this dissertation we always assume that there are no loops in G . A *walk* $w_{i,j}$ from node i to node j in G is an alternate sequence of vertices and edges, for instance

$$w_{1,3} = v_1, e_{1,2}, v_2, e_{3,2}, v_3.$$

In a walk we consider a sequence of vertices who may be head or tail of the incident edge, disregarding its direction, as long as there exist an edge connectivity any two consecutive nodes in the walk.

A *path* $p_{i,j}$ from node i to node j in G is an alternate sequence of vertices and

edges, for instance

$$p_{1,3} = v_1, e_{1,2}, v_2, e_{2,3}, v_3.$$

In a path we consider a sequence of vertices who are tail of the sequent edge and whose next vertex is a head for the same edge. A path between node i and j exists only if there exists a sequence of nodes and edges that can be visited respecting the direction of the edges.

In an undirected graph in which edges do not have a direction, a walk is equivalent to a path.

We now give various kinds of connectivity definitions depending on some properties of graph G .

- *Disconnected* - We say that graph G is disconnected if there exists two nodes i and j and there does not exist a walk from i to j .
- *Weakly connected* - We say that graph G is weakly connected if for any couple of nodes $i, j \in V$ there exists a walk between i and j .
- *Quasi-strongly connected* - We say that graph G is quasi-strongly connected if from each node $i \in V$ there exist a path to node w .
- *Strongly connected* - We say that graph G is strongly connected if there exists a path between each pair of nodes $i, j, \in V$.

If graph G is undirected, we only distinguish between the case in which it is disconnected or connected.

Dynamic case

Since we are interested in multi-agent systems with a time-varying topology we often make use of time-varying graphs $G(t) = \{V, \mathcal{E}(t)\}$ where $V = \{1, \dots, n\}$ is the set of nodes and $\mathcal{E}(t) \subseteq \{V \times V\}$ is the time-varying set of edges that map each instant of time into a set of edges $\mathcal{E} : \mathbb{R} \rightarrow E$. We define the union of graph $G_1 = \{V_1, E_1\}$ and $G_2 = \{V_2, E_2\}$ as the graph $G = G_1 \cup G_2 = \{V_1 \cup V_2, E_1 \cup E_2\}$ whose vertex and edge set is the union of those of G_1 and G_2 . Given an interval of time $[t, t']$ we define the union graph $\mathcal{G}[t, t']$ over an interval of time as

$$\mathcal{G}[t, t'] = \bigcup_{\tau=t}^{t'} G(\tau).$$

A property \mathcal{P} of a dynamic graph $G(t)$ is said to be *uniform* if for any t , there exists $T > 0$ such that in the union graph

$$\mathcal{G}[t, t + T] = \bigcup_{\tau=t}^{t+T} G(\tau),$$

\mathcal{P} holds.

A dynamic graph $G(t)$ is thus uniformly strongly connected if for any t there exists T in which $\mathcal{G}[t, t + T]$ is strongly connected.

A.2 Graph spectral analysis and Multi-agent systems

Graph spectral analysis is a well established research area concerned with the characterization of the properties of a graph by its spectrum. As an example, the topology of a graph can be *classified* by its spectrum (145, 146, 147) and this may have practical applications, such as matching the actual spectrum of a network representing a formation to the one of a target formation to check whether it has been achieved. The reader is referred to (42, 144) for a detailed overview of results on this topic.

In this appendix we list some facts of graph spectral analysis that allow topological properties of a graph to be inferred by its spectrum, see (42, 144) for further details.

Note that in the following, for simplicity of presentation, we will denote as $0 = \lambda_1 < \lambda_2 < \dots < \lambda_m$, $m \leq n$, the set of distinct eigenvalues of the Laplacian matrix \mathcal{L} . Moreover, we denote as $\hat{\lambda}_1 \leq \hat{\lambda}_2 \leq \dots \leq \hat{\lambda}_n$ the set of eigenvalues of \mathcal{L} , where two or more $\hat{\lambda}_i$'s coincide if their multiplicity is greater than 1.

- The graph *diameter*. It is a critical parameter in telecommunication networks since it influences the maximum number of hops that a packet needs to perform before reaching a destination. Let m be the total number of

distinct eigenvalues of \mathcal{L} . Then its diameter D is bounded by:

$$D \leq m - 1.$$

Furthermore, we have the following upper bound

$$D \leq \frac{4}{n\lambda_2},$$

where n is the total number of agents.

- The number of nodes, which is usually assumed not to be known in the context of multi-agent systems, is clearly equal to the number of eigenvalues with their multiplicity. Thus, given the number of distinct eigenvalues m , we have the following bound:

$$n \geq m.$$

- Let $E \subseteq \{V \times V\}$ be the set of edges and let Δ_i be the degree of agent i . Then the total number of links in the network is bounded by:

$$|E| = \frac{1}{2}Tr(\mathcal{L}) \leq \frac{1}{2} \sum_{i=1}^m \lambda_i = \frac{1}{2} \sum_{i=1}^m \Delta_i$$

where $Tr(\mathcal{L})$ denotes the trace of \mathcal{L} . Note that if all the eigenvalues are distinct, i.e., $m = n$, then the inequality becomes an equality. Conversely, in the case of some eigenvalues with multiplicity higher than one it simply gives a lower bound to the number of edges.

- Let \mathcal{K}_0 be the vertex connectivity, namely the minimal number of vertices (agents) whose removal disconnects the network. Let \mathcal{K}_1 be the edge connectivity, namely the minimal number of edges whose removal disconnects the network. Let δ_{min} be the minimum degree, then:

$$\lambda_2 \leq \mathcal{K}_0 \leq \mathcal{K}_1 \leq \delta_{min}.$$

- Let a *bridge* be an edge whose removal disconnects the network. A graph \mathcal{G} has edge connectivity 1 if and only if it has a bridge. Thus the graph does not have a bridge if $\lambda_2 > 1$ (42).

- Given a graph \mathcal{G} , if we add an edge to form \mathcal{G}' then:

$$\lambda_2(\mathcal{G}) \leq \lambda_2(\mathcal{G}') \leq \lambda_2(\mathcal{G}) + 2.$$

- Let Δ_{ave} be the average degree in the network, computed by executing a simple consensus algorithm. Then if all the eigenvalues are distinct:

$$n = \frac{\sum_{i=1, \dots, n} \lambda_i}{\Delta_{ave}}.$$

It follows that if we can monitor the number of agents connected to the network online, we can easily detect network disconnections in a decentralized way.

- The graph topology can be *classified* by its spectrum. In (145, 146, 147) and therein references, the shape of the spectrum of a given network is used to characterize its topology. For instance given a formation, we can match the actual spectrum of the network to the spectrum of the desired formation to check whether it has been achieved. This can only verify that the formation graph is *co-spectral* with the desired one.
- Since topological changes of the network influence directly the graph spectrum, an interesting research problem is to study spectral variations induced by adding or removing edges. In (148) for instance, authors investigate how the Laplacian spectral radius changes when rewiring a network by adding and removing some edges from one vertex to another.
- Complex networks are renown to be characterized by their spectra. The ability to estimate the whole spectra of a network allows to assess the kind of complex network in place, such information is specially helpful when the objective is to rewire a network such that it matches some particular kind of complex network to achieve its properties like robustness to random failure or high algebraic connectivity.

Bibliography

- [1] R. Beckers, J. Deneubourg, and S. Goss, "Trails and u-turns in the selection of a path by the ant *Lasius niger*," *Journal of Theoretical Biology*, vol. 159, no. 4, pp. 397–415, 1992. [2](#)
- [2] S. Goss, S. Aron, J. Deneubourg, and J. Pasteels, "Self-organized shortcuts in the Argentine ant," *Naturwissenschaften*, vol. 76, no. 12, pp. 579–581, 1989. [2](#)
- [3] M. Dorigo and L. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *Evolutionary Computation, IEEE Transactions on*, vol. 1, no. 1, pp. 53–66, 2002. [2](#)
- [4] J. Parrish, S. Viscido, and D. Grunbaum, "Self-organized fish schools: an examination of emergent properties," *The biological bulletin*, vol. 202, no. 3, p. 296, 2002. [2](#)
- [5] C. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*. ACM, 1987, pp. 25–34. [2](#)
- [6] V. Blondel, J. Hendrickx, A. Olshevsky, and J. Tsitsiklis, "Convergence in multiagent coordination, consensus, and flocking," in *Proc. 44th IEEE Conf. on Decision and Control*, Seville, Spain, Dec. 2005, pp. 2996–3000. [2](#), [8](#), [14](#), [143](#)
- [7] R. Olfati-Saber, "Flocking for multi-agent dynamic system: Algorithms and theory," *IEEE Transactions on Automatic Control*, vol. 51, pp. 401–420, 2006. [2](#), [14](#), [41](#), [65](#), [143](#), [165](#), [173](#)
- [8] H. G. Tanner and A. Jadbabaie, "Stable flocking of mobile agents, part ii: Dynamic topology," in *In IEEE Conference on Decision and Control*, 2003, pp. 2016–2021. [2](#), [93](#)
- [9] T. Vicsek, A. Czirók, E. Ben-Jacob, I. Cohen, and O. Shochet, "Novel type of phase transition in a system of self-driven particles," *Physical Review Letters*, vol. 75, no. 6, pp. 1226–1229, 1995. [2](#), [14](#)
- [10] H. Levine, W. Rappel, and I. Cohen, "Self-organization in systems of self-propelled particles," *Physical Review E*, vol. 63, no. 1, p. 17101, 2000. [2](#)
- [11] D. Helbing, I. Farkas, and T. Vicsek, "Simulating dynamical features of escape panic," *Nature*, vol. 407, no. 6803, pp. 487–490, 2000. [2](#)
- [12] M. DeGroot, "Reaching a consensus," *Journal of the American Statistical Association*, vol. 69, no. 345, pp. 118–121, 1974. [2](#)
- [13] J. Tsitsiklis, "Problems in decentralized decision making and computation," Ph.D. dissertation, M. I. T., Dept. of Electrical Engineering and Computer Science, 1984. [2](#)
- [14] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Communications of the ACM*, vol. 21, no. 7, pp. 558–565, 1978. [2](#)
- [15] R. Olfati-Saber, J. Fax, and R. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, Jan. 2007. [3](#), [8](#), [12](#), [93](#)
- [16] J. Cortes, S. Martinez, and F. Bullo, "Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions," *IEEE Transactions on Automatic Control*, vol. 51, no. 8, pp. 1289–1298, 2006. [3](#), [143](#), [165](#)
- [17] J. Cortes, S. Martinez, and F. Bullo, "Robust rendezvous for mobile autonomous agents via proximity graphs in d dimensions," *IEEE Trans. Robot. Automat.*, vol. 51, no. 8, pp. 1289 – 1298, 2006. [3](#), [93](#)
- [18] D. Dimarogonas and K. Kyriakopoulos, "On the rendezvous problem for multiple nonholonomic agents," *IEEE Transactions on Automatic Control*, vol. 52, no. 5, pp. 916–922, 2007. [3](#), [93](#)
- [19] W. Ren, R. Beard, and E. Atkins, "A survey of consensus problems in multi-agent coordination," in *American Control Conference, 2005. Proceedings of the 2005*. IEEE, 2005, pp. 1859–1864. [3](#), [8](#)
- [20] L. Schenato and G. Gamba, "A distributed consensus protocol for clock synchronization in wireless sensor network," in *Decision and Control, 2007 46th IEEE Conference on*. IEEE, 2008, pp. 2289–2294. [3](#)
- [21] J. Widder, "Bootstrapping clock synchronization in partially synchronous systems," *Distributed Computing*, pp. 121–135, 2003. [3](#)
- [22] R. Carli, A. Chiuso, L. Schenato, and S. Zampieri, "A PI consensus controller for networked clocks synchronization," in *IFAC World Congress on Automatic Control (IFAC 08)*, 2008. [3](#)
- [23] X. Lin and S. Boyd, "Fast linear iterations for distributed averaging," *Systems and Control Letters*, vol. 53, pp. 65–78, 2004. [3](#), [8](#), [41](#), [65](#), [66](#), [144](#)
- [24] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Trans. on Automatic Control*, vol. 49, pp. 1520–1533, 2004. [3](#), [8](#), [9](#), [10](#), [41](#), [143](#), [147](#), [165](#)
- [25] T. Aysal, M. Coates, and M. Rabbat, "Distributed average consensus with dithered quantization," *IEEE Trans. on Signal Processing*, vol. 56, no. 10, Part 1, pp. 4905–4918, 2008. [3](#), [41](#), [43](#)

- [26] R. Carli and S. Zampieri, "Efficient quantization in the average consensus problem," *Advances in Control Theory and Applications*, pp. 31–49, 2007. [3](#), [20](#), [21](#), [41](#), [43](#)
- [27] G. Cybenko, "Dynamic load balancing for distributed memory multiprocessors," *J. of Parallel and Distributed Computing*, vol. 7, pp. 279–301, 1989. [3](#), [18](#), [41](#)
- [28] B. Ghosh and S. Muthukrishnan, "Dynamic load balancing by random matchings," *J. of Computer and Systems Sciences*, vol. 53, no. 3, pp. 357–370, 1996. [3](#), [18](#), [41](#), [43](#)
- [29] A. Kashyap, T. Başar, and R. Srikant, "Quantized consensus," *Automatica*, vol. 43,7, pp. 1192–1203, 2007. [3](#), [4](#), [18](#), [19](#), [21](#), [22](#), [41](#), [42](#), [44](#), [45](#), [49](#), [51](#), [56](#), [58](#), [61](#), [62](#), [63](#)
- [30] A. Cortes, A. Ripoll, M. Senar, P. Pons, and E. Luque, "On the performance of nearest-neighbors load balancing algorithms in parallel systems," in *Proc. of the 7th Euromicro Workshop on Parallel and Distributed Processing*, Funchal, Portugal, Feb. 1999, pp. 170–177. [3](#), [43](#)
- [31] Y. Rabani, A. Sinclair, and R. Wanka, "Local divergence of Markov chains and the analysis of iterative load-balancing schemes," in *Proc. of the 39th Annual Symposium on Foundations of Computer Science*, Palo Alto, CA, USA, Nov. 1998, pp. 694–703. [3](#)
- [32] W. Aiello, B. Awerbuch, B. Maggs, and S. Rao, "Approximate load balancing on dynamic and asynchronous networks," in *Proc. of the 25th ACM Symposium on Theory of Computing*, San Diego, CA, US, May 1993, pp. 632–641. [3](#)
- [33] R. Subramanian and I. Scherson, "An analysis of diffusive load-balancing," in *Proc. of the 6th annual ACM Symposium on Parallel Algorithms and Architectures*, Cape May, New Jersey, USA, 1994, pp. 220–225. [3](#)
- [34] M. Franceschelli, A. Giua, and C. Seatzu, "Load balancing on networks with gossip based distributed algorithms," in *Proc. 46th IEEE Conf. on Decision and Control*, New Orleans, LA, USA, Dec. 2007. [4](#), [22](#), [31](#), [41](#), [42](#), [43](#), [44](#), [45](#), [46](#), [51](#), [58](#), [61](#), [63](#)
- [35] W. Ren and R. Beard, "Consensus seeking in multi-agent systems under dynamically changing interaction topologies," *IEEE Trans. on Automatic Control*, vol. 50 (5), pp. 655–661, 2005. [8](#)
- [36] I. Daubechies and J. Lagarias, "Sets of matrices all infinite products of which converge," *Linear algebra and its applications*, vol. 161, pp. 227–263, 1992. [8](#), [16](#), [67](#)
- [37] A. Dimakis, S. Kar, J. Moura, M. Rabbat, and A. Scaglione, "Gossip algorithms for distributed signal processing," *Proceedings of the IEEE*, no. 99, pp. 1–18, 2010. [8](#), [15](#)
- [38] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Gossip algorithms: design, analysis and applications," in *Proc. IEEE Infocom 2005*, Miami, USA, Mar. 2005. [8](#), [22](#), [66](#), [86](#)
- [39] T. Aysal, M. Yildiz, A. Sarwate, and A. Scaglione, "Broadcast gossip algorithms for consensus," in *IEEE Transactions on Signal processing*, vol. 57, no. 7, 2009, pp. 2748–2761. [8](#), [15](#), [71](#), [81](#)
- [40] R. Merris, "Laplacian matrices of a graph: A survey," *Linear Algebra and its Appl.*, vol. 197, pp. 143–176, 1994. [9](#)
- [41] N. Biggs, "Algebraic Graph Theory. Cambridge Tracks in Mathematics," 1974. [9](#)
- [42] C. Godsil and G. Royle., "Algebraic graph theory," *Springer*, 2001. [10](#), [102](#), [119](#), [143](#), [187](#), [188](#)
- [43] R. Olfati-Saber and R. M. Murray, "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. [11](#), [65](#), [66](#), [119](#)
- [44] W. Ren and R. Beard, "Consensus seeking in multi-agent systems under dynamically changing interaction topologies," *Automatic Control, IEEE Transactions on*, vol. 50, no. 5, pp. 655–661, May 2005. [11](#)
- [45] G. Xie and L. Wang, "Consensus control for networks of dynamic agents via active switching topology," in *Advances in Natural Computation*, ser. Lecture Notes in Computer Science. Springer Berlin/Heidelberg, 2005. [11](#)
- [46] Z. Lin, B. Francis, and M. Maggiore, "State agreement for continuous-time coupled nonlinear systems," *SIAM Journal on Control and Optimization*, vol. 46, no. 1, pp. 288–307, 2008. [11](#), [12](#)
- [47] L. Moreau, "Stability of multiagent systems with time-dependent communication links," *Automatic Control, IEEE Transactions on*, vol. 50, no. 2, pp. 169–182, 2005. [12](#)
- [48] R. Olfati-Saber and R. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *Automatic Control, IEEE Transactions on*, vol. 49, no. 9, pp. 1520–1533, Sept. 2004. [13](#), [93](#)
- [49] J. Fax and R. Murray, "Information flow and cooperative control of vehicle formations," *Automatic Control, IEEE Transactions on*, vol. 49, no. 9, pp. 1465–1476, 2004. [13](#)
- [50] 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, pp. 988–1001, 2003. [14](#), [41](#), [65](#), [143](#), [165](#)
- [51] D. Bertsekas and J. Tsitsiklis, "Comments on §Coordination of groups of mobile autonomous agents using nearest neighbor rules," *Automatic Control, IEEE Transactions on*, vol. 52, no. 5, pp. 968–969, 2007. [14](#)
- [52] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *Automatic Control, IEEE Transactions on*, vol. 31, no. 9, pp. 803–812, 2002. [14](#)

- [53] R. Olfati-Saber, "Distributed kalman filtering for sensor networks," in *46th IEEE Conference on Decision and Control*, New Orleans, LA, USA, 2007. 14, 66
- [54] U. A. Khan and J. M. F. Moura, "Distributed kalman filters in sensor networks: Bipartite fusion graphs," in *IEEE/SP 14th Workshop on Statistical Signal Processing, Madison, Wisconsin, USA*, 2007. 14, 66
- [55] M. Alighanbari and J. How, "An unbiased kalman consensus algorithm," in *American Control Conference, Minneapolis, Minnesota, USA*, 2006. 14, 66
- [56] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Trans. on Information Theory*, vol. 52 (6), pp. 2508–2530, 2006. 15, 17, 18, 42
- [57] —, "Randomized gossip algorithms," *IEEE/ACM Trans. Netw.*, vol. 14, no. S1, pp. 2508–2530, 2006. 15
- [58] G. Picci and T. Taylor, "Almost sure convergence of random gossip algorithms," in *Proc. 46th IEEE Conf. on Decision and Control*, New Orleans, LA, USA, Dec. 2007. 15, 66
- [59] A. Dimakis, A. Sarwate, and M. Wainwright, "Geographic gossip: efficient aggregation for sensor networks," in *Information Processing in Sensor Networks, 2006. IPSN 2006. The Fifth International Conference on*. IEEE, 2006, pp. 69–76. 15
- [60] H. Bauschke, "A norm convergence result on random products of relaxed projections in hilbert space," *Trans. Amer. Math. Soc.*, vol. 347, pp. 1365–1373, 1994. 16, 101
- [61] T. Aysal, M. Coates, and M. Rabbat, "Distributed average consensus with dithered quantization," *IEEE Trans. on Signal Processing*, vol. 56, no. 10, pp. 4905–4918, 2008. 20, 21
- [62] N. Michael, M. Zavlanos, V. Kumar, and G. Pappas, "Distributed multi-robot task assignment and formation control," in *Proc. 2008 IEEE International Conference on Robotics and Automation*, Pasadena, California, May 2008. 21
- [63] K. Lerman, C. Jones, A. Galstyan, and M. Mataric, "Analysis of dynamic task allocation in multi-robot systems," *The Int. Journal of Robotics Research*, vol. 25 (3), pp. 225–242, 2006. 21
- [64] M. Zavlanos and G. Pappas, "Dynamic assignment in distributed motion planning with local coordination," *IEEE Trans. on Robotics*, vol. 24 (1), pp. 232–242, 2008. 21
- [65] M. Ji, S. Azuma, and M. Egerstedt, "Role-assignment in multi-agent coordination," *Int. Journal of Assistive Robotics and Mechatronics*, vol. 7 (1), pp. 32–40, 2006. 21
- [66] Y.-W. Leung, "Processor assignment and execution sequence for multiversion software," *IEEE Tran. on Computers*, vol. 14 (12), pp. 1371–1377, 1997. 22
- [67] M. Franceschelli, A. Giua, and C. Seatzu, "Load balancing over heterogeneous networks with gossip-based algorithms," in *2009 American Control Conference*, St. Louis, Missouri, USA, Jun. 2009. 22
- [68] D. Bauso, L. Giarré, and R. Pesenti, "Non-linear protocols for optimal distributed consensus in networks of dynamic agents," *Systems and Control Letters*, vol. 55, no. 11, pp. 918–928, 2006. 41
- [69] M. Herlihy and S. Tirthapura, "Self-stabilizing smoothing and balancing networks," *Distributed Computing*, 2005. 41, 43
- [70] M. Houle, E. Tempero, and G. Turner, "Optimal dimension exchange token distribution on complete binary trees," *Theoretical Computer Science*, vol. 220, pp. 363–377, 1999. 41, 43
- [71] R. Carli, F. Fagnani, A. Speranzon, and S. Zampieri, "Communication constraints in the average consensus problem," *Automatica*, vol. 44, no. 3, pp. 671–684, 2008. 41
- [72] T. Aysal, M. Coates, and M. Rabbat, "Distributed average consensus using probabilistic quantization," *IEEE/SP 14th Workshop on Statistical Signal Processing*, pp. 640–644, August 2007. 41
- [73] R. Carli, F. Fagnani, P. Frasca, and S. Zampieri, "Gossip consensus algorithms via quantized communication," *Automatica*, vol. 46, no. 1, pp. 70–80, 2010. 42
- [74] M. Zhu and S. Martinez, "On the convergence time of distributed quantized averaging algorithms," in *47th IEEE Conf. on Decision and Control, Cancun, Mexico*, 2008, pp. 3971–3976. 42
- [75] J. Lavaei and R. Murray, "On quantized consensus by means of gossip algorithm—Part I: convergence proof," in *American Control Conference, St. Louis, MO, USA*, 2009, pp. 394–401. 42
- [76] —, "On quantized consensus by means of gossip algorithm—Part II: convergence time," in *American Control Conference, St. Louis, MO, USA*, 2009, pp. 2958–2965. 42
- [77] M. Franceschelli, A. Giua, and A. Seatzu, "A gossip-based algorithm for discrete consensus over heterogeneous networks," *IEEE Trans. on Automatic Control*, vol. 55, no. 5, pp. 1244–1249, 2010. 42
- [78] M. Franceschelli, A. Giua, and C. Seatzu, "Hamiltonian quantized gossip," in *Proc. 2009 IEEE Multi-conference on Systems and Control*, St. Petersburg, Russia, July 2009, pp. 648–654. 43
- [79] B. Ghosh, F. Leighton, B. Maggs, S. Muthukrishnan, C. Plaxton, R. Rajaraman, A. Richa, R. Tarjan, and D. Zuckerman, "Tight analyses of two local load balancing algorithms," *SIAM J. on Computing*, vol. 29, no. 1, pp. 29–64, 2000. 43
- [80] M. Houle, A. Symvonis, and D. Wood, "Dimension exchange algorithms for token distribution on tree-connected architectures," *J. of Parallel and Distributed Computing*, vol. 64, pp. 591–605, 2004. 43

- [81] F. Meyer Auf Der Heide, B. Oesterdiekhoff, and R. Wanka, "Strongly adaptive token distribution," in *Lecture Notes in Computer Science*, vol. 700, 1993, pp. 398–409. 43
- [82] G. Turner and H. Schroder, "Token distribution on reconfigurable d-dimensional meshes," in *Proc. 1st IEEE Int. Conf. on Algorithms and Architectures for Parallel Processing*, vol. 1, 1995, pp. 335–344. 43
- [83] T. Fenner and A. Frieze, "On the existence of hamiltonian cycles in a class of random graphs," *Discrete Mathematics*, vol. 45, no. 2-3, pp. 301–305, 1983. 47
- [84] M. Garey and D. Johnson, *Computers and Intractability; A Guide to the Theory of NP-completeness*. W. H. Freeman and Co., New York, NY, USA, 1990. 47
- [85] K. Day and A. Tripathi, "Embedding of cycles in arrangement graphs," *IEEE Trans. on Computers*, vol. 42, no. 8, pp. 1002–1006, 1993. 47
- [86] E. Levy, G. Louchard, and J. Petit, "A distributed algorithm to find hamiltonian cycles in random graphs," in *Combinatorial and Algorithmic Aspects of Networking*, pp. 63–74. 47
- [87] R. Dixon, N. Strole, and J. Markov, "A token-ring network for local data communications," *IBM Systems J.*, vol. 22, no. 1-2, pp. 47–62, 1983. 47
- [88] N. Bshouty, L. Higham, and J. Warpechowska-Gruca, "Meeting times of random walks on graphs," *Information Processing Letters*, vol. 69, no. 5, pp. 259–265, 1999. 59
- [89] P. Tetali and P. Winkler, "On a random walk problem arising in self-stabilizing token management," in *PODC '91: Proc. 10th Annual ACM Symposium on Principles of Distributed Computing*, New York, NY, USA, 1991, pp. 273–280. 59
- [90] F. Fagnani and S. Zampieri, "Randomized consensus algorithms over large scale networks," *IEEE J. on Selected Areas of Communications*, 2008 (to appear). 65, 66
- [91] M. Porfiri and D. Stilwell, "Consensus seeking over random weighted directed graphs," *IEEE Transactions on Automatic Control*, vol. 52, no. 9, pp. 1767–1773, 2007. 65, 66
- [92] M. Rabbat, "On spatial gossip algorithms for average consensus," in *IEEE/SP 14th Workshop on Statistical Signal Processing, Madison, Wisconsin, US*, 2007, pp. 705–709. 66, 86
- [93] S. Kar and J. Moura, "Distributed consensus algorithms in sensor networks with imperfect communication: Link failures and channel noise," *IEEE Transactions on Signal Processing*, vol. 57, no. 1, pp. 355–369, 2009. 66, 86
- [94] —, "Sensor networks with random links: Topology design for distributed consensus," *IEEE Transactions on Signal Processing*, vol. 56, no. 7 Part 2, pp. 3315–3326, 2008. 66, 86, 88
- [95] M. Franceschelli, A. Giua, and C. Seatzu, "Consensus on the average on arbitrary strongly connected digraphs based on broadcast gossip algorithms," in *1st IFAC Workshop on Estimation and Control of Networked Systems*, Venice, Italy, Sep. 2009. 66, 67
- [96] K. Cai and H. Ishii, "Convergence time analysis of quantized gossip algorithms on digraphs," in *Proc. 49th IEEE Conf. on Decision and Control*, Atlanta, GA, USA, december 2010. 67
- [97] —, "Gossip consensus and averaging algorithms with quantization," in *American Control Conference (ACC), 2010*. IEEE, 2010, pp. 6306–6311. 67
- [98] —, "Further results on randomized quantized averaging: A surplus-based approach," in *Computer-Aided Control System Design (CACSD), 2010 IEEE International Symposium on*. IEEE, pp. 1558–1563. 67
- [99] M. Franceschelli, M. Egerstedt, and G. A., "Motion probes for fault detection and recovery in networked control systems," in *Proceedings of the American Control Conference, Seattle, WA, USA*, 2008, pp. 2958–2965. 67, 133
- [100] M. Franceschelli, M. Egerstedt, A. Giua, and C. Mahulea, "Constrained invariant motions for networked control systems," in *Proceedings of the American Control Conference, St. Louis, Missouri, USA*, 2009. 67
- [101] M. Franceschelli, A. Giua, and C. Seatzu, "Distributed averaging in sensor networks based on broadcast gossip algorithms," *IEEE Sensor Journal, Special Issue On Cognitive Sensor Networks*, 2011, to appear. 67
- [102] T. Aysal, M. Yildiz, A. Sarwate, and A. Scaglione, "Broadcast gossip algorithms: Design and analysis for consensus," in *47th IEEE Conference on Decision and Control, Cancun, Mexico*, 2008, pp. 4843–4848. 71, 81
- [103] A. Jadbabaie, J. Lin, and A. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *Automatic Control, IEEE Transactions on*, vol. 48, no. 6, pp. 988–1001, June 2003. 93
- [104] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Trans. Automat. Contr.*, vol. 51, no. 3, pp. 401 – 420, 2006. 93
- [105] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Trans. Automat. Contr.*, vol. 49, no. 9, pp. 1520 – 1533, 2004. 93
- [106] G. Ferrari-Trecate, M. Egerstedt, A. Buffa, and M. Ji, "Laplacian sheep: A hybrid, stop-go policy for leader-based containment control." in *HSCC*, vol. 3927, 2006, pp. 212–226. 93
- [107] J. Rodrigues, D. Figueira, C. Neves, and M. I. Ribeiro, "Leader-following graph-based distributed formation control," in *Proc. of Robotica 2008 - 8th Conference on Autonomous Robot Systems and Competitions*, 2008. 93

- [108] D. Dimarogonas, P. Tsiotras, and K. Kyriakopoulos, "Laplacian cooperative attitude control of multiple rigid bodies," *Intelligent Control, 2006. IEEE International Symposium on*, pp. 3064–3069, Oct. 2006. 93
- [109] M. Ji and M. Egerstedt, "Distributed coordination control of multi-agent systems while preserving connectedness," *IEEE Transactions on Robotics*, vol. 23, no. 4, pp. 693–703, August 2007. 93
- [110] Z. Lin, M. Broucke, and B. Francis, "Local control strategies for groups of mobile autonomous agents," *IEEE Trans. Automat. Contr.*, vol. 49, no. 4, pp. 622 – 629, 2004. 93
- [111] W. Ren, R. Beard, and E. Atkins, "A survey of consensus problems in multi-agent coordination," *American Control Conference*, vol. 3, pp. 1859–1864, 2005. 93
- [112]
- [113] A. Bensoussan and J. Menaldi, "Difference equations on weighted graphs," *Journal of Convex Analysis (Special issue in honor of Claude Lemarechal)*, vol. 12, no. 1, pp. 13–44, 2005. 117
- [114] A. B. G. Ferrari-Trecate and M. Gati, "Analysis of coordination in multi-agent systems through partial difference equations. part i: The laplacian control." *16th IFAC World Congress on Automatic Control*, 2005. 117
- [115] A. B. G. Ferrari-Trecate, M. Egerstedt and M. Ji, "Laplacian sheep: A hybrid, stop-go policy for leader-based containment control." *Hybrid Systems: Computation and Control, Santa Barbara, CA, March*, pp. 212–226, 2006. 117, 134
- [116] A. Muhammad and M. Egerstedt, "Connectivity graphs as models of local interactions," *CDC04*, vol. 1, pp. 124–129, 2004. 143
- [117] M. Franceschelli, A. Gasparri, A. Giua, and C. Seatzu, "Decentralized laplacian eigenvalues estimation for networked multi-agent systems," in *48th IEEE Conference on Decision and Control*, 2009. 144, 146, 150, 175
- [118] D. Spanos and R. Murray, "Robust connectivity of networked vehicles," in *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, vol. 3, 14–17 2004, pp. 2893 – 2898. 144
- [119] M. De Gennaro and A. Jadbabaie, "Decentralized control of connectivity for multi-agent systems," in *Decision and Control, 2006 45th IEEE Conference on*, 13–15 2006, pp. 3628 –3633. 144
- [120] M. Zavlanos and G. Pappas, "Potential fields for maintaining connectivity of mobile networks," *Robotics, IEEE Transactions on*, vol. 23, no. 4, pp. 812 –816, aug. 2007. 145
- [121] —, "Distributed connectivity control of mobile networks," *Robotics, IEEE Transactions on*, vol. 24, no. 6, pp. 1416 –1428, dec. 2008. 145
- [122] P. Yang, R. A. Freeman, G. Gordon, K. Lynch, S. Srinivasa, and R. Sukthankar, "Decentralized estimation and control of graph connectivity in mobile sensor networks," in *American Control Conference*, June 2008. 145, 156, 157
- [123] K. Savla, G. Notarstefano, and F. Bullo, "Maintaining limited-range connectivity among second-order agents," *SIAM J. Control Optim.*, vol. 48, no. 1, pp. 187–205, 2009. 145
- [124] J. R. Silvester, "Determinants of block matrices," *The Mathematical Gazette*, vol. 84, no. 501, pp. 460 – 467, November 2000. 149
- [125] M. Ji and M. Egerstedt, "Observability and estimation in distributed sensor networks," in *46th IEEE Conference on Decision and Control*, New Orleans, LA, December 2007, pp. 4221–4226. 154, 155, 166
- [126] S. Martini, M. Egerstedt, and A. Bicchi, "Controllability decompositions of networked systems through quotient graphs," in *47th IEEE Conference on Decision and Control*, Cancun, Mexico, December 2008, pp. 5244–5249. 154, 166
- [127] M. Ji and M. Egerstedt, "Distributed coordination control of multi-agent systems while preserving connectedness," vol. 23, no. 4, pp. 693–703, Aug. 2007. 165
- [128] M. M. A. Rahmani, M. Ji and M. Egerstedt, "Controllability of multi-agent systems from a graph-theoretic perspective," *SIAM Journal on Control and Optimization*, 2008. 166
- [129] S. Martini, M. Egerstedt, and A. Bicchi, "Controllability analysis of networked systems using relaxed equitable partitions," *International Journal of Systems, Control and Communications*, vol. 2, 2010. 166
- [130] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Trans. on Automatic Control*, vol. 48, no. 6, pp. 988–1001, 2003. 167
- [131] R. Olfati-Saber and R. M. Murray, "Consensus protocols for networks of dynamic agents," in *Proc. American Contr. Conf.*, Denver, CO, Jun. 2003, pp. 951–956. 167
- [132] A. Olshevsky and J. N. Tsitsiklis, "Convergence rates in distributed consensus and averaging," San Diego, CA, Dec. 2006, pp. 3387–3392. 167
- [133] H. Tanner, A. Jadbabaie, and G. J. Pappas, "Flocking in fixed and switching networks," *IEEE Trans. on Automatic Control*, vol. 52, no. 5, pp. 863–868, 2007. 167
- [134] A. Rahmani, M. Ji, M. Mesbahi, and M. Egerstedt, "Controllability of multi-agent systems from a graph-theoretic perspective," vol. 48, no. 1, pp. 162–186, Feb 2009. 167
- [135] M. Ji, A. Muhammad, and M. Egerstedt, "Leader-based multi-agent coordination: Controllability and optimal control," in *American Control Conference*, Minneapolis, MN, Jun. 2006, pp. 1358–1363. 167

- [136] K. Ogata, *Modern control engineering*. Prentice Hall, 2009. [170](#)
- [137] E. van Dam and W. Haemers, "Which graphs are determined by their spectrum?" *Linear Algebra and its Applications*, vol. 373, pp. 241–272, 2003. [172](#)
- [138] W. Haemers and E. Spence, "Enumeration of cospectral graphs," *European Journal of Combinatorics*, vol. 25, no. 2, pp. 199–211, 2004. [172](#)
- [139] W. Wang and C. Xu, "A sufficient condition for a family of graphs being determined by their generalized spectra," *European Journal of Combinatorics*, vol. 27, no. 6, pp. 826–840, 2006. [172](#)
- [140] J. A. Fax and R. M. Murray, "Information flow and cooperative control of vehicle formations," *IEEE Trans. on Automatic Control*, vol. 49, no. 9, pp. 1465–1476, 2004. [173](#)
- [141] M. Egerstedt and X. Hu, "Formation constrained multi-agent control," *IEEE Trans. on Automation*, vol. 17, no. 6, pp. 947–951, 2001. [173](#)
- [142] M. Mesbahi and F. Y. Hadaegh, "Formation flying control of multiple spacecraft via graphs, matrix inequalities, and switching," vol. 24, no. 2, pp. 369–377, 2001. [173](#)
- [143] Z. Lin, B. Francis, and M. Maggiore, "Necessary and sufficient graphical conditions for formation control of unicycles," *IEEE Trans. on Automatic Control*, vol. 50, no. 1, pp. 121–127, Jan 2005. [173](#)
- [144] B. Mohar, "The laplacian spectrum of graphs," in *Graph Theory, Combinatorics, and Applications*. Wiley, 1991, pp. 871–898. [173](#), [187](#)
- [145] R. Grone, R. Merris, and V. S. Sunder, "The laplacian spectrum of a graph," *SIAM Journal on Matrix Analysis and Applications*, vol. 11, no. 2, pp. 218–238, 1990. [187](#), [189](#)
- [146] R. Grone and R. Merris, "The laplacian spectrum of a graph ii," *SIAM J. Discret. Math.*, vol. 7, no. 2, pp. 221–229, 1994. [187](#), [189](#)
- [147] J. V. D. Heuvel, "Using laplacian eigenvalues and eigenvectors in the analysis of frequency assignment problems," 2001. [187](#), [189](#)
- [148] J.-M. Guo, "The laplacian spectral radius of a graph under perturbation," *Comput. Math. Appl.*, vol. 54, no. 5, pp. 709–720, 2007. [189](#)