UNIVERSITÀ DI PISA

Facoltà di Ingegneria

Laurea Specialistica in Ingegneria dell'Automazione

Tesi di laurea

Distributed Intrusion Detection for Secure Consensus Computations

Candidato:

Fabio Pasqualetti _____

Relatori:

Antonio Bicchi _____

Francesco Bullo _____

Sessione di Laurea del 10/07/2007
Archivio tesi di Laurea Specialistica in Ingegneria dell'Automazione etd-06142007-103838
Anno accademico 2006/2007

Abstract

This work focuses on trustworthy computation systems and proposes a novel intrusion detection scheme for consensus networks with misbehaving nodes. This prototypical control problem is relevant in network security applications. The objective is for each node to detect and isolate the misbehaving nodes using only the information flow adopted by standard consensus protocols. We focus mainly on the single misbehaving node problem. Our technical approach is based on the theory of Unknown Input Observability. First, we give necessary and sufficient conditions for the misbehavior to be observable and for the identity of the faulty node to be detectable. Second, we design a distributed unknown input estimator, and we characterize its convergence rate in the "equal-neighbor" model and in the general case. Third and finally, we propose a complete detection and isolation scheme and provide some remarks on the filter convergence time. We also analyze the multiple misbehaving nodes problem, and we propose an algorithm to deal with it. We conclude the document with the numerical study of a consensus problem, of a robot deployment problem, and of an averaging problem.

Sommario

Data una rete di agenti autonomi, si propone un nuovo sistema di monitoraggio per garantire l'efficacia degli algoritmi di consenso in presenza di nodi non cooperanti. Questo problema di controllo è rilevante per migliorare la sicurezza nei sistemi multi agente. L'obiettivo è di permettere ad ogni agente di rilevare, identificare e (possibilmente) isolare gli agenti intrusi dalla rete, facendo uso soltanto delle informazioni previste dall'algoritmo di controllo. Questo documento tratta in dettaglio il caso in cui è presente un solo agente intruso nella rete, e alla fine viene proposto un algoritmo adatto al caso generale, in cui vi sono più intrusi. Il metodo sviluppato si basa sul concetto di osservabilità di un sistema lineare in presenza di un ingresso non noto (Unknown Input Observability). Dopo aver descritto le condizioni necessarie e sufficienti per rilevare il malfunzionamento nella rete, viene sintetizzato uno stimatore decentralizzato che permette di identificare l'identità del nodo intruso. Viene inoltre caratterizzata la velocità di convergenza del filtro per un particolare modello di rete ("equal-neighbor") e nel caso generale. Infine viene descritta una procedura distribuita per la risoluzione del problema di Intrusion Detection, e vengono proposti alcuni esempi numerici.

Contents

1 Introduction **11**

2 Basic Notions **15**
2.1 Preliminary Concepts and Notation 15
2.2 An introduction to Markov Chains 16
2.2.1 Absorbing Markov Chains 22
2.3 Asymptotic (Average) Consensus 23
2.3.1 Convergence Rate and Convergence Time 25

3 Unknown Input Observability: The Geometric Approach **27**
3.1 Geometric Tools 27
3.2 Unknown Input Observation of a Linear Function of the State 30

4 Single Intruder Detection and Isolation **33**
4.1 Solvability Conditions 33
4.2 Convergence Rate Analysis 36
4.2.1 Equal - Neighbor Case 36
4.2.2 Upper Bound for Weighted Digraph 39
4.3 Intrusion Detection Procedure 40
4.3.1 Analysis of the Iteration Error 40
4.3.2 Intrusion Detection Algorithm 41

5 Multi Intruders Detection and Isolation **43**
5.1 An Exact Solution 44
5.1.1 Absorbtion Probability and Iteration Error 45
5.1.2 Constant Unknown Inputs 47
5.2 Distributed Robust Averaging Algorithm 47
5.2.1 Reduced Unknown Input Filter 50
5.2.2 Intrusion Detection and Filter Initialization 51
5.2.3 Convergence Time 52

6 Applications **53**
6.1 Agreement Evaluation 53

6.1.1 Numerical Example 53
6.2 Linear Deployment 55
6.2.1 Numerical Example 58
6.3 Maximum Likelihood Estimation 59
6.3.1 Numerical Example 62

7 Conclusions **67**

Bibliography **69**

List of Figures

3.1 Internal and external stability of an invariant subspace. 29

5.1 Consensus network. 44
5.2 Iteration error. 44
5.3 Filter weights. 50
5.4 Filter initialization. 51
5.5 Error propagation: error detected by node 5 (right), and by node 6 (left). . 51

6.1 Consensus network. 55
6.2 Identification of the misbehaving node. 55
6.3 Modified consensus network. 56
6.4 Deployment initialization 57
6.5 Identification of the misbehaving node. 59
6.6 A sensor network with 12 nodes (blue) and 4 intruders (red) 62
6.7 Network assembly (1) 63
6.8 Network assembly (2) 64
6.9 Network assembly (3) 64
6.10 Network assembly (4) 65

Chapter 1

Introduction

In recent years, mobile ad-hoc networks have received much attention due to their potential applications and the proliferation of mobile devices. Specifically, mobile ad-hoc networks refer to wireless multi-hop networks formed by a set of mobile nodes without relying on a preexisting infrastructure. The problem of controlling a system comprising a large number of autonomous agents has attracted substantial attention, and has lead to the development of several distributed algorithms to solve tasks as varied as parameter estimation, average consensus, rendezvous, sensor coverage and simultaneous localization and mapping ([14]). The term distributed covers a large variety of concurrent algorithms used for a wide range of applications. Originally, this term was used to refer to algorithms that were designed to run on many processors "distributed" over a large geographical area, but over the years, the usage of this term has been broadened, so that it now includes also algorithms that run on autonomous agents networks.

The distributed consensus problem has historically appeared in many diverse areas, such as parallel computation [31],[4], control theory [16],[27], and communication networks [27]. Recently, the problem has attracted significant attention [29],[1], motivated by new contexts and open problems in communications, sensor networks, and networked control theory. We briefly describe some more of the more recent applications.

Reputation management in ad hoc networks: It is often the case that the nodes of a wireless multi-hop network are not controlled by a single authority or do not have a common objective. Selfish behavior among nodes (e.g., refusing to forward traffic meant for others) is possible, and some mechanism is needed to enforce cooperation. One way to detect selfish behavior is reputation management: each node forms an opinion by observing the behavior of its neighbors. One is then faced with the problem of combining these different opinions into a single globally available reputation measure for each node. The use of distributed consensus algorithms for doing this was explored in [21], where a variation of one of the methods we examine here - the "agreement algorithm" - was used as a basis for an empirical investigation.

Sensor networks: A sensor network designed for detection or estimation needs to combine various measurements into a decision or into a single estimate. Distributed computation of this decision/estimate has the advantage of being fault-tolerant (network op-

eration is not dependent a small set of nodes) and self-organizing (network functionality does not require constant supervision) [32, 2, 3, 12].

Control of autonomous agents: It is often necessary to coordinate collections of autonomous agents (e.g., cars or UAVs). For example, one may wish for the agents to agree on a direction or speed. Even though the data related to the decision may be distributed through the network, it is usually desirable that the final decision depend on all the known data, even though most of them are unavailable at each node. A model motivated by such a context was empirically investigated in [33].

Algorithms that solve the distributed consensus problem provide the means by which networks of agents can be coordinated. Although each agent may have access to different local information, the agents can agree on a decision (e.g., on a common direction of motion, on the time to execute a move, etc.). Such synchronized behavior has often been observed in biological systems [11].

For a detailed description and formulation of the distributed consensus problem we refer to [2]. Given a set $N = \{1, 2, \ldots, n\}$ of agents embedded, at each time t, in a directed graph $G = (N, E)$, where E represents the edge set and t lies in some discrete set of times. Each agent i starts with a scalar value $x_i(0)$; the vector with the values of all agents at time t will be denoted by $x(l) = (x_1(l), \ldots, x_n(l))$. The agreement algorithm updates $x(l)$ according to the equation $x(l+1) = F(l)x(l)$, or

$$x_i(l+1) = \sum_{j=1}^{n} f_{ij}(l)x_j(l), \tag{1.1}$$

where $F(l)$ is a nonnegative matrix with entries $f_{ij}(l)$. The row-sums of $F(l)$ are equal to 1, so that $F(l)$ is a stochastic matrix. In particular, $x_i(l+1)$ is the weighted average of the values $x_j(l)$ held by the agents at time t. If there is a positive constant α such that

1. $f_{ii}(l) \geq \alpha$, for all i, t;
2. $f_{ij}(l) \in \{0\} \cup [\alpha, 1]$, for all i, j, t;
3. $\sum_{j=1}^{n} f_{ij}(l) = 1$, for all i, t;

and if, following an arbitrary time t, for any i, j, there is a sequence of communications trough which node i will influence (directly or indirectly) the value held by node j, then the agreement algorithm guarantees asymptotic consensus, that is, there exists some c (depending on $x(0)$ and on the sequence of graphs $G(.)$) such that $\lim_{l\to\infty} x_i(l) = c$, for all i.

The iteration $x := Fx$ that solves the consensus problem can be used in a simple manner to provide a solution to the average consensus problem as well. The (distributed) average consensus problem is to compute the average $(1/n)\sum_{i=1}^{n} x_i(0)$ at every node, via local communication and computation on the graph, as in the agreement algorithm [17]. With the time-invariant agreement algorithm $x(l+1) = Fx(l)$, we have

$$\lim_{l\to\infty} x_i(l) = \sum_{i=1}^{n} \pi_i x_i(0), \quad \forall i, \tag{1.2}$$

where π is the steady-state probability vector of the Markov chain associated with the stochastic matrix F. It follows that we obtain a solution to the averaging problem if and only if $\pi_i = 1/n$ for every i. Since π is a left eigenvector of F, with eigenvalue equal to 1, this requirement translates to the property $\mathbf{1}^T F = \mathbf{1}^T$, where $\mathbf{1}$ is the vector with all components equal to 1. Equivalently, the matrix F needs to be doubly stochastic. An elegant solution to the problem is given by the Metropolis rule [18], while a new method based on the execution of two agreement algorithm is proposed in [2].

For any such algorithm to be practical, the nodes are assumed to cooperate and to follow exactly the control protocol, otherwise the task is not guaranteed to be fulfilled. It is of increasing importance to design distributed systems capable of performing trustworthy computations in the face of failures and intrusions.

In the literature we can find several results, mostly based on reputation systems, that deal with cooperation issues among nodes of an ad-hoc network. The reputation, as defined in [26], is an index for the reliability of a node in the network, i.e., for the contribution to common network operations, and it is used to exclude uncooperative and misbehaving nodes from the network. Intentional non-cooperation is mainly caused by two types of nodes: selfish ones, that want to save power, and malicious nodes, that are not primarily concerned with power saving but that are interested in attacking the network [5]. In [6], [7], [8], [9], authors develop security systems, where trust relationship and routing decision are based on routing and forwarding behavior of the nodes.

In the fault detection and isolation literature, we find several strategies to construct model-based fault detection systems [13]. One of them is the observer-based technique, whose main idea is to make the decision on possible faults in the process on the basis of the residuals generated by estimating the outputs of the process. The detection system should be immune to faults, such that the differences between the outputs of the process and those of the observer give information about faults in the system.

In [12], a distributed consensus algorithm is implemented by a sensors network to design a distributed fault diagnosis procedure for dynamic system, but the fault diagnosis of the consensus protocol is not considered.

In the agreement algorithms we are considering, agents use the information coming from the neighbors to update their state. Since there are no forwarded messages, reputation systems are not applicable, unless we add a communication structure to the control mechanism. Consider for instance the prototypical problem of uniform deployment on a segment [24]. A malicious agent moving towards an extreme of the segment would push all agents on that side to the extreme point, without them being able to detect the misbehavior from the information they possess (i.e., the distance from their immediate neighbors).

The main contributions of this work are as follows. We apply a technique based on unknown-input observers to the problem of intrusion detection for a linear averaging algorithm. We give conditions for the solvability of the problem in relation to the topology of the network, and we prove that if the network is 2-connected, then a faulty node can be detected, identified and finally isolated from the network, to preserve the functionality of the algorithm. We design an embedded filter, which, only considering the information coming from the neighbors, asymptotically estimates the state of the other nodes of the

network, and we analyze the properties of the estimation error, which is strictly related to the fault induced in the system. The estimation rate of the filter is also considered, and we prove that, in the "equal-neighbor" model, as the number of agents grows, the largest eigenvalue of the observer has the same upper bound of the second largest eigenvalue of the iteration matrix of the algorithm, which determines the convergence rate of the protocol. We also compute an estimation of the convergence rate of the filter in a more general case, i.e., when the associated graph is weighted and directed.

We also address the Intrusion Detection problem in the case of many faulty processors. It is shown that the solution proposed in the single intruder case is not usable, and an algorithm to assemble a secure network for the asymptotic agreement computation is proposed. The algorithm can be used, by properly choosing the weight matrix, to perform the distributed Maximum Likelihood estimation as described in [19].

Chapter 2

Basic Notions

2.1 Preliminary Concepts and Notation

A *directed graph*, in short a *digraph*, of order n is a pair $G = (V, E)$ where $V = \{v_1, \ldots, v_n\}$ is a set with n elements called *vertices* or *nodes*, and E is a set of ordered pair of vertices called *edges*, i.e., $E \subseteq V \times V$. We call V and E the *vertex set* and the *edge set* respectively, and we let $V(G)$ and $E(G)$ denote the vertices and the edges of a graph G. For $u, v \in V$, the ordered pair (u, v) denotes an edge from u to v. A digraph is called *undirected* if $(u, v) \in E$ anytime $(v, u) \in E$.

A *weighted digraph* is a triplet $G = (V, E, A)$ where V and E are a digraph and where A is an $n \times n$ A *weighted adjacency matrix* whose entries a_{ij} correspond to the cost of the edge from i to j, i.e., $a_{ij} > 0$ only if $(i, j) \in E$. A weighted digraph is undirected if $a_{ij} = a_{ji}$. A digraph $G = (V, E)$ can be thought of as a weighted digraph by defining the weighted adjacency matrix A as

$$a_{ij} = \begin{cases} 1, & \text{if } (v_i, v_j) \in E \\ 0, & \text{otherwise.} \end{cases} \tag{2.1}$$

Clearly G is undirected if and only if A is symmetric.

In a weighted digraph $G = (V, E, A)$ with $V = \{v_1, \ldots, v_n\}$ the *weighted out degree* and the *weighted in degree* of vertex v_i are defined by

$$d_{\text{out}}(v_i) = \Sigma_{j=1}^{n} a_{ij}, \quad \textit{and} \quad d_{in}(v_i) = \Sigma_{j=1}^{n} a_{ji}. \tag{2.2}$$

The *weighted out-degree matrix* $D_{\text{out}}(G)$ and *weighted in-degree matrix* $D_{\text{in}}(G)$ are the diagonal matrices defined by $(D_{\text{out}}(G))_{ii} = d_{\text{out}}(v_i)$ and $(D_{\text{in}}(G))_{ii} = d_{\text{in}}(v_i)$. The *graph Laplacian* of the weighted digraph G is

$$L(G) = D_{\text{out}}(G) - A(G). \tag{2.3}$$

Immediate consequences of these definitions include $L(G)\mathbf{1}_n = \mathbf{0}_n$, and the weighted digraph is undirected if and only if $L(G)$ is symmetric.

A *path* in a digraph is a sequence of vertices such that from each of its vertices there is an edge to the next vertex in the sequence.

A digraph is called *strongly connected* if for every pair of vertices $(v_i, v_j) \in V$ there is a path from v_i to v_j.

Let $T \in \mathbb{Z}_{\geq 0}$, a digraph is called $T-$*connected* if and only if it contains T internally disjoint paths between any two vertices.

2.2 An introduction to Markov Chains

A Markov chain is a sequence of random variables $X_1, X_2, X_3, \ldots$, taking values in $\{S_1, \ldots, S_n\}$, that satisfies the Markov property, i.e.,

$$\Pr(X_{t+1} = S_j \mid X_t = S_{i_t},\ X_{t-1} = S_{i_{t-1}}, \ldots, X_0 = S_{i_0}) = \Pr(X_{t+1} = S_j \mid X_t = S_{i_t}). \quad (2.4)$$

The Markov property asserts that the process is memoryless in the sense that the state of the chain at the next time period depends only on the current state and not on the past history of the chain.

Let the value $p_{ij} = Pr(X_t = S_j \mid X_{t-1} = S_i)$ be the probability of being in the state S_j at time t given that the chain is in state S_i at time $t-1$, so $p_{ij}(t)$ is called the transition probability of moving from S_i to S_j at time t. The matrix of transition probabilities $P_{n\times n}(t) = [p_{ij}(t)]$ is clearly a nonnegative row stochastic matrix. When the transition probabilities do not vary with time, the chain is said to be stationary, and the transition matrix is the constant stochastic matrix $P = [p_{ij}]$. Conversely, every stochastic matrix $P_{n\times n}$ defines a $n-$state Markov chain because the entries p_{ij} define a set of transition probabilities, which can be interpreted as a stationary Markov chain on n states.

A probability distribution vector is defined to be a nonnegative vector $p^T = [p_1, p_2, \ldots, p_n]$ such that $\sum_k p_k = 1$. For an $n-$state Markov chain, the k^{th} step probability distribution vector is defined to be

$$p^T = [p_1(k), p_2(k), \ldots, p_n(k)], \quad k = 1, 2, \ldots, \text{ where } p_j(k) = Pr(X_k = S_j). \quad (2.5)$$

In other words, $p_j(k)$ is the probability of being in the j^{th} state after the k^{th} step, but before the $(k+1)^{th}$ step.

The k^{th} step distribution is easily described by using the laws of elementary probability $-$ in particular, recall that $P(E \vee F) = P(E) + P(F)$ when E and F are mutually exclusive events, and the conditional probability of E occurring given that F occurs is $P(E \mid F) = P(E \wedge F)/P(F)$ (its convenient to use $\wedge$ and $\vee$ to denote AND and OR, respectively). To

determine the j^{th} component $p_j(1)$ in $p^T(1)$ for a given $p^T(0)$, write

$$p_j(1) = \Pr(X_1 = S_j) = \Pr\left[X_1 = S_j \wedge \left(X_0 = S_1 \vee X_0 = S_2 \vee \cdots \vee X_0 = S_n\right)\right] \tag{2.6}$$

$$= \Pr\left[(X_1 = S_j \wedge X_0 = S_1) \vee (X_1 = S_j \wedge X_0 = S_2) \vee \cdots \vee (X_1 = S_j \wedge X_0 = S_n)\right] \tag{2.7}$$

$$= \sum_{i=1}^{n} \Pr\left[X_1 = S_j \wedge X_0 = S_i\right] = \sum_{i=1}^{n} \Pr[X_0 = S_i]\Pr[X_1 = S_j \mid X_0 = S_i] \tag{2.8}$$

$$= \sum_{i=1}^{n} p_i(0)p_{ij} \quad \text{for} \quad j = 1, 1, \ldots, n. \tag{2.9}$$

Consequently, $p^T(1) = p^T(0)P$. This tells us what to expect after one step when we start with $p^T(0)$. But the no memory Markov property tells us that the state of affairs at the end of two steps is determined by where we are at the end of the first step - it is like starting over but with $p^T(1)$ as the initial distribution. In other words, it follows that $p^T(2) = p^T(1)P$, and $p^T(3) = p^T(2)P$, etc. Therefore, successive substitution yields

$$p^T(k) = p^T(k-1)P = p^T(k-2)P^2 = \ = p^T(0)P^k, \tag{2.10}$$

and thus the k^{th} step distribution is determined from the initial distribution and the transition matrix by the vector - matrix product

$$p^T(k) = p^T(0)P^k. \tag{2.11}$$

Notice that if we adopt the notation $P^k = [p_{ij}^{(k)}]$, and if we set $p^T(0) = e_i^T = \mathbf{1}$ in the equation above, then we get $p_j(k) = p_{ij}^{(k)}$ for each $i = 1, 2, \ldots, n$, and thus we arrive at the following conclusion.

- The $(i, j)-$entry in P^k represents the probability of moving from S_i to S_j in exactly k steps. For this reason, P^k is often called the $k-$step transition matrix.

Since P is a probability transition matrix, its entries are nonnegative. An important result in the theory of nonnegative matrices is the Perron-Frobenius Theorem, as follows:

Theorem 2.2.1 (Perron-Frobenius)**.** *If $A_{n\times n} \geq 0$ is irreducible, then each of the following is true.*

- $r = \rho(A) > 0$.
- $\rho(A)$ *is an eigenvalue of* A
- *There exists an eigenvector $x > 0$ such that $Ax = rx$*
- *The unique vector defined by*

$$Ap = rp, \quad p > 0, \quad \textit{and} \quad \|p\|_1 = 1, \tag{2.12}$$

is called the Perron vector. There are no nonnegative eigenvectors for A except for positive multiples of p, regardless of the eigenvalue.

Recall that a nonnegative irreducible matrix A having only one eigenvalue, $r = \rho(A)$, on its spectral circle is said to be a primitive matrix, and that for a primitive matrix A yields

$$\lim_{k\to\infty}\left(\frac{A}{r}\right)^k = \frac{pq^T}{q^Tp} > 0, \tag{2.13}$$

where p and q are the respective Perron vectors for A and A^T.

We are now ready to analyze the limiting properties of certain Markov chains. In case the matrix P is primitive, i.e., irreducible, we know exactly what $\lim_{k\to\infty} P^k$ looks like. The Perron vector for P is e/n (the uniform distribution vector), so if $\pi = [\pi_1, \pi_2, \ldots, \pi_n]^T$ is the Perron vector for P^T, then

$$\lim_{k\to\infty} P^k = \frac{(e/n)\pi^T}{\pi^T(e/n)} = \frac{e\pi^T}{\pi^Te} = \begin{bmatrix} \pi_1 & \pi_2 & \cdots & \pi_n \\ \pi_1 & \pi_2 & \cdots & \pi_n \\ \vdots & \vdots & & \vdots \\ \pi_1 & \pi_2 & \cdots & \pi_n \end{bmatrix} > 0, \tag{2.14}$$

because $\pi^Te = 1$. Therefore, if P is primitive, then a limiting probability distribution exists, and it is given by

$$lim_{k\to\infty}p^T(k) = lim_{k\to\infty}p^T(0)P^k = p^T(0)e\pi^T = \pi^T. \tag{2.15}$$

Notice that because $\sum_k p_k(0) = 1$, the term $P^T(0)e$ drops away, so we have the conclusion that the value of the limit is independent of the value of the initial distribution $p^T(0)$.

In the case the matrix P is imprimitive, i.e., P is irreducible and has $h > 1$ eigenvalues on the unit spectral circle, the limit $\lim_{k\to\infty} P^k$ does not exist, and hence $\lim_{k\to\infty} p^T(k)$ cannot exist [25]. However, each eigenvalue on the unit circle is simple, and this means that P is Cesàro summable. Moreover, e/n is the Perron vector for P, and, if $\pi^T = (\pi_1, \pi_2, \ldots, \pi_n)$ is the left-hand Perron vector, then

$$\lim_{k\to\infty}\frac{I + P + \cdots + P^{k-1}}{k} = \frac{(e/n)\pi^T}{\pi^T(e/n)} = e\pi^T = \begin{bmatrix} \pi_1 & \pi_2 & \cdots & \pi_n \\ \pi_1 & \pi_2 & \cdots & \pi_n \\ \vdots & \vdots & & \vdots \\ \pi_1 & \pi_2 & \cdots & \pi_n \end{bmatrix}, \tag{2.16}$$

which is exactly the same form as the limit for the primitive case. Consequently, the k^{th} step distributions have a Cesàro limit given by

$$\lim_{k\to\infty}\left[\frac{p^T(0) + p^T(1 + \cdots + p^T(k-1))}{k}\right] = \lim_{k\to\infty} p^T(0)\left[\frac{I + P + \cdots + P^{k-1}}{k}\right] \tag{2.17}$$

$$= p^T(0)e\pi^T = \pi^T, \tag{2.18}$$

and, just as in the primitive case, this Cesàro limit is independent of the initial distribution.

We can summarize the properties of irreducible Markov chains as follows:

Remark 1. *Let P be the transition probability matrix for an irreducible Markov chain on states $S_1, S_2, \ldots, S_n$ (i.e., P is an $n \times n$ irreducible stochastic matrix), and let π^T denote the left-hand Perron vector for P. The following statements are true for every initial distribution $p^T(0)$.*

- *The k^{th} step transition matrix is P^k because the (i, j) -entry in P^k is the probability of moving from S_i to S_j in exactly k steps.*
- *The k_{th} step distribution vector is given by $p^T(k) = p^T(0)P^k$.*
- *If P is primitive, and if e denotes the column of all 1s, then*

$$\lim_{k\to\infty} P^k = e\pi^T \quad \textit{and} \quad \lim_{k\to\infty} p^T(k) = \pi^T. \tag{2.19}$$

- *If P is imprimitive, then*

$$\lim_{k\to\infty} \frac{I + P + \cdots + P^{k-1}}{k} = e\pi^T \tag{2.20}$$

and

$$\lim_{k\to\infty} \left[\frac{p^T(0) + p^T(1 + \cdots + p^T(k-1))}{k}\right] = \pi^T. \tag{2.21}$$

- *π^T is often called the stationary distribution vector for the chain because it is the unique distribution vector satisfying $\pi^T P = \pi^T$.*

Because the PerronFrobenius theorem is not directly applicable to reducible chains (chains for which P is a reducible matrix), the strategy for analyzing reducible chains is to deflate the situation, as much as possible, back to the irreducible case as described below. If P is reducible, then, by definition, there exists a permutation matrix Q and square matrices X and Z such that $Q^T P Q = \begin{bmatrix} X & Y \\ 0 & Z \end{bmatrix}$. For convenience, denote this by writing $P \sim \begin{bmatrix} X & Y \\ 0 & Z \end{bmatrix}$. If X or Z is reducible, then another symmetric permutation can be performed to produce

$$\begin{bmatrix} X & Y \\ 0 & Z \end{bmatrix} \sim \begin{bmatrix} R & S & T \\ 0 & U & V \\ 0 & 0 & W \end{bmatrix}, \quad \text{where } R, U, W \text{ are square.} \tag{2.22}$$

Repeating the process eventually yields

$$P \sim \begin{bmatrix} X_{11} & X_{12} & \cdots & X_{1k} \\ 0 & X_{22} & \cdots & X_{2k} \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & X_{kk} \end{bmatrix}, \quad \text{where each } X_{ii} \text{ is irreducible or } X_{ii} = [0]_{1\times 1}. \tag{2.23}$$

Finally, if there exist rows having nonzero entries only in diagonal blocks, then symmetrically permute all such rows to the bottom to produce

$$P \sim \begin{bmatrix} P_{11} & P_{12} & \cdots & P_{1r} & P_{1,r+1} & P_{1,r+2} & \cdots & P_{1m} \\ 0 & P_{22} & \cdots & P_{2r} & P_{2,r+1} & P_{2,r+2} & \cdots & P_{2m} \\ \vdots & & \ddots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & P_{rr} & P_{r,r+1} & P_{r,r+2} & \cdots & P_{rm} \\ 0 & 0 & \cdots & 0 & P_{r+1,r+1} & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & P_{r+2,r+2} & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & P_{mm} \end{bmatrix} \tag{2.24}$$

where each $P_{11}, \ldots, P_{rr}$ is either irreducible or $[0]_{1\times 1}$, and $P_{r+1,r+1}, \ldots, P_{mm}$ are irreducible (they cant be zero because each has row sums equal to 1). The effect of a symmetric permutation is simply to reorder the states in the chain. When the states of a chain have been reordered so that P assumes the form above, we say that P is in the canonical form for reducible matrices. When P is in canonical form, the subset of states corresponding to P_{kk} for $1 \leq k \leq r$ is called the k_{th} transient class (because once left, a transient class cant be reentered), and the subset of states corresponding to $P_{r+j,r+j}$ for $j \geq 1$ is called the j^{th} ergodic class. Each ergodic class is an irreducible Markov chain unto itself that is imbedded in the larger reducible chain. From now on, we will assume that the states in our reducible chains have been ordered so that P is in canonical form.

Suppose now that $P = \begin{bmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{bmatrix}$ is a reducible stochastic matrix that is in the canonical form, where

$$T_{11} = \begin{bmatrix} P_{11} & \cdots & P_{1r} \\ & \ddots & \vdots \\ & & P_{rr} \end{bmatrix}, \quad T_{12} = \begin{bmatrix} P_{1,r+1} & \cdots & P_{1m} \\ \vdots & & \vdots \\ P_{r,r+1} & \cdots & P_{rm} \end{bmatrix} \tag{2.25}$$

and

$$T_{22} = \begin{bmatrix} P_{r+1,r+1} & & \\ & \ddots & \\ & & P_{mm} \end{bmatrix} \tag{2.26}$$

Observe that

$$\rho(P_{kk}) < 1 \quad \text{for each } k = 1, 2, \ldots, r. \tag{2.27}$$

This is certainly true when $P_{kk} = [0]_{1\times 1}$, so suppose that P_{kk} $(1 \leq k \leq r)$ is irreducible. Because there must be blocks P_{kj}, $j \neq k$, that have nonzero entries (otherwise P_{kk} would be an ergodic class), it follows that

$$P_{kk}e < e \quad \text{where } e \text{ is the column of all 1's.} \tag{2.28}$$

If $\rho(P_{kk}) = 1$, then, for the Perron-Frobenius Theorem, would be $P_{kk}e = e$, which is impossible, and thus $\rho(P_{kk}) < 1$. It follows that $\rho(T_{11}) < 1$, and hence [25]

$$\lim_{k\to\infty} \frac{I + T_{11} + \cdots + T_{11}^{k-1}}{k} = \lim_{k\to\infty} T_{11}^k = 0. \tag{2.29}$$

Furthermore, $P_{r+1,r+1}, \ldots, P_{mm}$ are each irreducible stochastic matrices, so if π_j^T is the left-hand Perron vector for P_{jj}, $r+1 \leq j \leq m$, then our previous results tell us that

$$\lim_{k\to\infty} \frac{I + T_{22} + \cdots + T_{22}^{k-1}}{k} = \begin{bmatrix} e\pi_{r+1}^T & & \\ & \ddots & \\ & & e\pi_m^T \end{bmatrix} = E. \tag{2.30}$$

Furthermore, $\lim_{k\to\infty} T_{22}^k$ exists if and only if $P_{r+1,r+1}, \ldots, P_{mm}$ are each primitive, in which case $\lim_{k\to\infty} T_{22}^k = E$. Therefore, the limits, be they Cesàro or ordinary (if it exists), all have the form

$$\lim_{k\to\infty} \frac{I + P + \cdots + P^{k-1}}{k} = G = \begin{bmatrix} 0 & Z \\ 0 & E \end{bmatrix} = \lim_{k\to\infty} P^k \text{ when it exists.} \tag{2.31}$$

To determine the precise nature of Z, use the fact that $\mathrm{Im}(G) = \mathrm{Ker}(I - P)$ (because G is the projector onto $\mathrm{Ker}(I - P)$ along $\mathrm{Im}(I - P)$) to write

$$(I - P)G = 0 \Rightarrow \begin{bmatrix} I - T_{11} & -T_{12} \\ 0 & I - T_{22} \end{bmatrix} \begin{bmatrix} 0 & Z \\ 0 & E \end{bmatrix} = 0 \Rightarrow (I - T_{11})Z = T_{12}E. \tag{2.32}$$

Since $I - T_{11}$ is nonsingular (because $\rho(T11) < 1$), it follows that

$$Z = (I - T_{11})^{-1}T_{12}E. \tag{2.33}$$

Remark 2. *If the states in a reducible Markov chain have been ordered to make the transition matrix assume the canonical form*

$$P = \begin{bmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{bmatrix}, \tag{2.34}$$

and if π_j^T is the left-hand Perron vector for P_{jj} $(r+1 \leq j \leq m)$, then $(I - T_{11})$ is nonsingular, and

$$\lim_{k\to\infty} \frac{I + P + \cdots + P^{k-1}}{k} = \begin{bmatrix} 0 & (I - T_{11})^{-1}T_{12}E \\ 0 & E \end{bmatrix}, \tag{2.35}$$

where

$$E = \begin{bmatrix} e\pi_{r+1}^T & & \\ & \ddots & \\ & & e\pi_m^T \end{bmatrix}. \tag{2.36}$$

Furthermore, $\lim_{k\to\infty} P^k$ exists if and only if the stochastic matrices of the ergodic classes are primitive, in which case

$$\lim_{k\to\infty} P^k = \begin{bmatrix} 0 & (I - T_{11})^{-1}T_{12}E \\ 0 & E \end{bmatrix}. \tag{2.37}$$

2.2.1 Absorbing Markov Chains

The preceding analysis shows that every reducible chain eventually gets absorbed (trapped) into one of the ergodic classes, i.e., into a subchain defined by $P_{r+j,r+j}$ for some $j \geq 1$. If $P_{r+j,r+j}$ is primitive, then the chain settles down to a steady-state defined by the left-hand Perron vector of $P_{r+j,r+j}$, but if $P_{r+j,r+j}$ is imprimitive, then the process will oscillate in the j^{th} ergodic class forever. There is not much more that can be said about the limit, but there are still important questions concerning which ergodic class the chain will end up in and how long it takes to get there. This time the answer depends on where the chain startsi.e., on the initial distribution.

For convenience, let Γ_i denote the i^{th} transient class, and let Λ_j be the j^{th} ergodic class. Suppose that the chain starts in a particular transient state, say we start in the p^{th} state of Γ_i. Since the question at hand concerns only which ergodic class is hit but not what happens after it is entered, we might as well convert every state in each ergodic class into a trap by setting $P_{r+j,r+j} = I$ for each $j \geq 1$. The transition matrix for this modified chain is $\tilde{P} = \begin{bmatrix} T_{11} & T_{12} \\ 0 & I \end{bmatrix}$, and we know that $\lim_{k\to\infty} \tilde{P}^k$ exists and has the form

$$\lim_{k\to\infty} \tilde{P}^k = \begin{bmatrix} 0 & (I - T_{11})^{-1}T_{12} \\ 0 & I \end{bmatrix} = \begin{bmatrix} 0 & 0 & \cdots & 0 & L_{1,1} & L_{1,2} & \cdots & L_{1,s} \\ 0 & 0 & \cdots & 0 & L_{2,1} & L_{2,2} & \cdots & L_{2,s} \\ \vdots & & \ddots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 0 & L_{r,1} & L_{r,2} & \cdots & L_{r,s} \\ 0 & 0 & \cdots & 0 & I & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & I & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & I \end{bmatrix}. \tag{2.38}$$

Consequently, the (p, q)-entry in block L_{ij} represents the probability of eventually hitting the q^{th} state in Λ_j given that we start from the p^{th} state in Γ_i. Therefore, if e is the vector of all 1 s, then the probability of eventually entering somewhere in Γ_j is given by

- Pr(absorbtion into Λ_j | start in p^{th} state of Γ_i)=$\sum_k [L_{ij}]_{pk} = [L_{ij}e]_p$.

If $p_i^T(0)$ is an initial distribution for starting in the various states of Γ_i, then

- Pr(absorbtion into Λ_j | $p_i^T(0)$)=$p_i^T(0)L_{ij}e$.

Its often the case in practical applications that there is only one transient class, and the ergodic classes are just single absorbing states (states such that once they are entered, they are never left). If the single transient class contains r states, and if there are s absorbing

states, then the canonical form for the transition matrix is

$$P = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1r} & p_{1,r+1} & p_{1,r+2} & \cdots & p_{1m} \\ 0 & p_{22} & \cdots & p_{2r} & p_{2,r+1} & p_{2,r+2} & \cdots & p_{2m} \\ \vdots & & \ddots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & p_{rr} & p_{r,r+1} & p_{r,r+2} & \cdots & p_{rm} \\ 0 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \quad \text{and} \quad L_{ij} = [(I - T_{11})^{-1} T_{12}]_{ij}. \tag{2.39}$$

The preceding analysis specializes to say that every absorbing chain must eventually reach one of its absorbing states. The probability of being absorbed into the j^{th} absorbing state (which is state S_{r+j}) given that the chain starts in the i^{th} transient state (which is S_i) is

- Pr(absorbtion into S_{r+j} | start in S_i for $1 \leq i \leq r$)=$[(I - T_{11})^{-1} T_{12}]_{ij}$.

2.3 Asymptotic (Average) Consensus

In this work, we focus on a particular class of iterative algorithms for consensus, widely used in the applications cited above. Each node updates itself with the weighted sum of the states of its immediate neighbors:

$$w_i(l+1) = \sum_{j \in N_i(l)} f_{ij}(l) w_j(l), \quad i = \{1, \ldots, n\}, \quad l \in \mathbb{Z}_{\geq 0}. \tag{2.40}$$

Here, $f_{ij}(l)$ is a weight associated with the edge j, i. These weights are algorithm parameters, and can be chosen according to the particular problem to be solved. Setting $f_{ij}(l) = 0$ for $j \notin N_i(l)$ and $f_{ii}(l) = 1 - \sum_{j \in N_i(l)} f_{ij}(l)$, the iterative method can be expressed as the simple linear iteration

$$w(l+1) = F(l) w(l), \quad l \in \mathbb{Z}_{\geq 0}. \tag{2.41}$$

By construction, the weight matrix $F(l)$ is row stochastic, i.e., satisfies the properties

$$\sum_{j=1}^{n} f_{ij}(l) = 1 \quad \text{and} \quad f_{ij}(l) \geq 0, \quad i, j = \{1, \ldots, n\}. \tag{2.42}$$

If the weights are assigned such that $f_{ij}(l) = f_{ji}(l)$, then the weight matrix is symmetric and the asymptotic value of each node correspond to the average $(1/n) \sum_{i=1}^{n} w_i(0)$ (Section 2.2).

In the literature there are at least two alternative ways in which consensus algorithms are introduced. Let $G = (V, E, A)$ be a weighted digraph. Then,

i) a first form of agreement algorithm is given by

$$w(l+1) \;=\; [I - \varepsilon L(G)]w(l), \quad l \in \mathbb{Z} \geq 0 \tag{2.43}$$

where $L(G)$ is the weighted Laplacian matrix associated with G. In order for the matrix $[I - \varepsilon L(G)]$ to be stochastic, the constant ε must satisfy the bound $0 < \varepsilon \leq min_{i \in I} 1/d_{out}(i)$. We term this type of agreement algorithm *Laplacian-based*.

ii) a second form of agreement algorithm is given by

$$w(l+1) \;=\; [I_n + D_{out}(G)]^{-1}[I_n + A(G)]w(l), \quad l \in \mathbb{Z} \geq 0 \tag{2.44}$$

where $D_{out}(G)$ and $A(G)$ are respectively the out-degree and the adjacency matrices of G. Observe that the resulting stochastic matrix has always non-zero diagonal entries. We term this type of agreement algorithm *adjacency based*.

We next state some conditions under which the agreement algorithm is guaranteed to converge.

Assumption 1. *There exists a positive constant α such that:*

1. $f_{ii}(l) \geq \alpha$, *for all i,j,l.*
2. $f_{ij}(l) \in \{0\} \cup [\alpha, 1]$,*for all i,j,l.*
3. $\sum_{j=1}^{n} f_{ij}(l) = 1$,*for all i,j,l.*

Intuitively, whenever $f_{ij}(l) > 0$, node j communicates its current value $x_j(l)$ to node i. Each node i updates its own value, by forming a weighted average of its own value and the values it has just received from other nodes. We represent the sequence of communications between nodes by a sequence $G(l) = (N, E(l))$ of directed graphs, where $(j,i) \in E(l)$ if and only if $f_{ij}(l) > 0$. Note that $(i,i) \in E(l)$ for all l. Our next assumption requires that following an arbitrary time l, and for any i, j, there is a sequence of communications through which node i will influence (directly or indirectly) the value held by node j, i.e., the graph $G = (N, E(l))$ is strongly connected.

We note various special cases of possible interest.

Time-invariant model: In this model the set of arcs $E(l)$ is the same for all l; furthermore, the matrix $F(l)$ is the same for all l. In this case, we are dealing with the iteration $x := Fx$, where F is a stochastic matrix; in particular, $x(l) = F^l x(0)$. Under the assumptions above, F is the transition probability matrix of an irreducible and aperiodic Markov chain. Thus, F^l converges to a matrix all of whose rows are equal to the (positive) vector $\pi = (\pi_1, \ldots, \pi_n)$ of steady-state probabilities of the Markov chain. Accordingly, we have $\lim_{l \to \infty} x_i(l) = \sum_{i=1}^{n} \pi_i x_i(0)$.

Bidirectional model: In this case, we have $(i,j) \in E(l)$ if and only if $(j,i) \in E(l)$, and we say that the graph G is symmetric. Intuitively, whenever i communicates to j, there is a simultaneous communication from j to i.

Equal-neighbor model: Here

$$f_{ij} = \begin{cases} 1/d_i(l), & \text{if } j \in N_i(l), \\ 0, & \text{if } j \notin N_i(l), \end{cases} \tag{2.45}$$

where $N_i(l) = \{j \mid (j,i) \in E(l)\}$ is the set of nodes j (including i) whose value is taken into account by i at time l, and $d_i(l)$ is its cardinality.

Under the assumptions above, the agreement algorithm guarantees asymptotic consensus, that is, there exists some c (depending on $x(0)$ and on the sequence of graphs $G(.)$) such $\lim_{l\to\infty} x_i(l) = c$, for all i.

Average consensus is an important problem in algorithm design for distributed computing. Let $G = (V, E)$ be an undirected connected graph with node set $V = \{v_1, \ldots, v_n\}$ and edge set E, where each edge $(i, j) \in E$ is an ordered pair of distinct nodes. Let $w_i(0)$ be a real scalar assigned to node i at time $l = 0$. The (distributed) average consensus problem is to compute the average $(1/n)\sum_{i=1}^{n} w_i(0)$ at every node, via local communication and computation on the graph. Thus, node i carries out its update, at each step, based on its local state and communication with its neighbors $N_i = \{j|j, i \in E\}$. Distributed average consensus has been extensively studied in computer science, for example in distributed agreement and synchronization problems [22]. It is a central topic for load balancing (with divisible tasks) in parallel computers [10]. More recently, it has also found applications in distributed coordination of mobile autonomous agents [28], and distributed data fusion in sensor networks [33]. There are several simple methods for distributed average consensus. For example, each node can store a table of all initial node values known at that time. At each step each pair of neighbors exchange tables of initial values (or just the entries the other node does not have), and update their tables. In this simple flooding algorithm, all nodes know all initial values in a number of steps equal to the diameter of the graph, at which point each can compute the average (or any other function of the initial values). In this work we focus on averaging algorithms that come directly from the agreement algorithms described above, i.e., we modify the iteration matrix F in order to compute the average of the initial values and not a simple consensus. In Section 6.3 we present three methods to solve the average consensus problem.

Many variation of the consensus algorithm have been studied. These include for example problems where the weights are not symmetric and problems where the final node value have a specified non-uniform distribution. Convergence conditions have also been established for distributed consensus on dynamically changing graphs with asynchronous communication and computation. The problem of the fastest distributed linear averaging has been studied, in which the weights are chosen to obtain the fastest convergence rate [32].

2.3.1 Convergence Rate and Convergence Time

An important aspect of agreement algorithms is the convergence time, i.e., the amount of time the algorithm needs to get arbitrarily close to the asymptotic distribution. Recall

that F is an irreducible stochastic matrix, i.e., $1 > \lambda_2 \geq \lambda_3, \ldots, \geq \lambda_n$ are its eigenvalues sorted in order of decreasing magnitude. Let X be the set of vectors of the form $c\mathbf{1}$, i.e., with equal components. The convergence rate is defined as

$$\rho = \sup_{x(0)\notin X} \lim_{l\to\infty} \left(\frac{\|x(l) - x^*\|_2}{\|x(0) - x^*\|_2} \right)^{1/t}, \tag{2.46}$$

where x^* stands for $\lim_{l\to\infty} x(l)$. As is well known, we have $\rho = \max\{|\lambda_2|, |\lambda_n|\}$, and, if $f_{ii} > 0$, we have $\rho = |\lambda_2|$. The convergence time is consequently defined by

$$T_n(\varepsilon) = min\left\{ \tau : \frac{\|x(l) - x^*\|_\infty}{\|x(0) - x^*\|_\infty} \leq \varepsilon, \ \ \forall t \geq \tau, \ \ \forall x(0) \notin X \right\}. \tag{2.47}$$

For the equal neighbor, time invariant, bidirectional model, tight bounds on the convergence rate were derived in [20].

Theorem 2.3.1. *[20] Consider the equal-neighbor, time invariant, bidirectional model, on a connected graph with n nodes. The convergence rate satisfies*

$$\rho \leq 1 - \gamma_1 n^{-3}, \tag{2.48}$$

where γ_1 is a constant independent oh n. Moreover there exists some $\gamma_2 > 0$ such that for every positive integer n, there exists a $n-$ node connected graph for which

$$\rho \leq 1 - \gamma_2 n^{-3}. \tag{2.49}$$

Theorem 2.3.1 is provided in [20] for the case of symmetric graphs without self-arcs. It's not hard to check that essentially the same proof goes trough when self-arcs are present, the only difference being in the value of the constant γ_1 and γ_2. This is intuitive because the effect of the self-arcs is essentially a slowing down of the Markov chain by a factor of at most 2, and therefore the convergence rateshould stay the same.

Using some additional results, Theorem 2.3.1 leads to a tight bound on the convergence time [2].

Corollary 2.3.1. *The convergence time for the equal-neighbor, time invariant, symmetric model on a connected graph on n nodes, satisfies*

$$T_n(\varepsilon) = O(n^3 log(n/\varepsilon)). \tag{2.50}$$

Furthermore, for every positive integer n, there exists a $n-$node connected graph for which

$$T_n(\varepsilon) = \Omega(n^3 log(1/\varepsilon)). \tag{2.51}$$

The $\Omega(n^3)$ convergence time is not particularly attractive. Authors in [2] explores possible improvements in the convergence time by using different choices for the weight matrix F. It is shown that the convergence rate can be brought arbitrarily close to zero, with the drawback of numerical instability.

Chapter 3

Unknown Input Observability: The Geometric Approach

The essence of the geometric approach consists of developing most of the mathematical support in coordinate-free form, to take advantage of simpler and more elegant results, which facilitate insight into the actual meaning of statements and procedures; the computational aspects are considered independently of the theory and handled by means of the standard methods of matrix algebra, once a suitable coordinate system is defined. The cornerstone of the approach is the concept of invariance of a subspace with respect to a linear transformation. In this chapter the properties and geometric meaning of invariants are presented and investigated in order to solve the Unknown Input Observation problem [23].

3.1 Geometric Tools

Consider a linear trasformation $A : \mathbf{X} \to \mathbf{X}$ with $\mathbf{X} = \mathbf{R}^n$. Recall that an A-invariant is a subspace $\mathbf{J} \subseteq \mathbf{X}$ such that

$$A\mathbf{J} \subseteq \mathbf{J}. \tag{3.1}$$

Invariant subspaces define the structure of linear transformations, thus playing an important role in linear dynamic system analysis. Consider the discrete time system described by the equations

$$\begin{array}{rcl} x(t+1) & = & Ax(t) + Bu(t) \\ y(t) & = & Cx(t) \end{array} . \tag{3.2}$$

It may be proved that in the absence of control action (i.e., when function $u()$ is identically zero) a subspace of the state space $\mathbf{X}$ is a locus of trajectories if and only if it is an A-invariant. The extension of this property to the case in which the control is present and suitably used to steer the state along a convenient trajectory leads to the concept of $(A, \mathbf{B}) -$ controlled invariant: a subspace $\mathbf{V} \subseteq \mathbf{X}$ is said to be an $(A, \mathbf{B}) -$ controlled invariant if

$$A\mathbf{V} \subseteq \mathbf{V} \oplus \mathbf{B} \quad \text{with} \quad \mathbf{B} = \mathrm{Im}(B). \tag{3.3}$$

The dual of the controlled invariant is the conditioned invariant, which is defined as follows: a subspace $\mathbf{S} \subseteq \mathbf{X}$ is said to be an $(A, \mathbf{C}) -$ conditioned invariant if

$$A(\mathbf{S} \cap \mathbf{C}) \subseteq \mathbf{S} \quad \text{with} \quad \mathbf{C} = \text{Ker}(C). \tag{3.4}$$

Note that any A-invariant is also an $(A, \mathbf{B}) -$ controlled invariant for any $\mathbf{B}$ and an $(A, \mathbf{C}) -$ conditioned invariant for any $\mathbf{C}$: in particular, the origin $\{\emptyset\}$ and the whole space $\mathbf{X}$ are so. Furthermore, the $(A, \{\emptyset\}) -$ controlled invariants and $(A, \mathbf{X}) -$ conditioned invariants are, in particular, A-invariants.

Recall that the reachability subspace of (3.2) is the set of all the states that can be reached from the origin in any finite time by means of control actions, and let $\Re$ denote that subspace, then it can be proven that

$$\Re = \text{min}\mathbf{J}(A, \mathbf{B}) \quad \mathbf{B} = \text{Im}(B), \tag{3.5}$$

where $\text{min}\mathbf{J}(A, \mathbf{B})$ denotes the minimal A-invariant containing $\mathbf{B}$. The dual result concerning observability is geometrically approached in a similar way. Let $\mathbf{Q}$ be the unobservability subspace of (3.2), i.e., the set of all the initial states that cannot be recognized from the output function, then

$$\mathbf{Q} = \text{max}\mathbf{J}(A, \mathbf{C}) \quad \mathbf{C} = \text{Ker}(C), \tag{3.6}$$

where $\text{max}\mathbf{J}(A, \mathbf{C})$ denotes the maximal A-invariant contained in $\mathbf{C}$.

Controlled and conditioned invariants are very important in connection with synthesis problems because of their feedback properties: in fact a controlled invariant can be transformed into a simple invariant by means of a suitable state feedback, just as a conditioned invariant can be transformed into a simple invariant by means of a suitable output injection. Formally we say that a subspace $\mathbf{S} \subseteq \mathbf{X}$ is an $(A, \mathbf{C}) -$ conditioned invariant if and only if there exists at least one matrix G such that $(A + GC)\mathbf{S} \subseteq \mathbf{S}$.

Still referring to the system (3.2) when the control input is identically zero, we shall now introduce the concept of stability of an invariant. We recall that system (3.2) is (asymptotically) stable if and only if all the eigenvalues of matrix A belong to the unit circle. By extension, in this case A is said to be a stable matrix. Since an A-invariant $\mathbf{J} \subseteq \mathbf{X}$ is a locus of trajectories, stability can be split with respect to $\mathbf{J}$. Consider the similarity transformation T such that

$$\tilde{A} = T^{-1}AT = \begin{bmatrix} \tilde{A}_{11} & \tilde{A}_{12} \\ 0 & \tilde{A}_{22} \end{bmatrix}, \tag{3.7}$$

where $\tilde{A}_{11}$ is an $h \times h$ matrix with $h = dim(\mathbf{J})$. Let $x = Tz$, in the new coordinate we obtain the system

$$\begin{bmatrix} z_1(t+1) \\ z_2(t+1) \end{bmatrix} = \begin{bmatrix} \tilde{A}_{11} & \tilde{A}_{12} \\ 0 & \tilde{A}_{22} \end{bmatrix} \begin{bmatrix} z_1(t) \\ z_2(t) \end{bmatrix}, \tag{3.8}$$

which is equivalent to the original system. Consider an initial state $x_0 \in \mathbf{J}$, the corresponding transformed state $z_0 = T^{-1}x_0$ decomposes into $[z_{01}, 0]^T$. The motion on $\mathbf{J}$ is described by

$$z_1(t+1) = \tilde{A}_{11}z_1(t), \quad z_1(0) = z_{01}, \tag{3.9}$$

while $z_2(t)$ remains identically zero. Therefore, the motion on $\mathbf{J}$ is stable if and only if submatrix $\tilde{A}_{11}$ is stable. On the other hand, consider an initial state $x_0 \notin \mathbf{J}$, so that

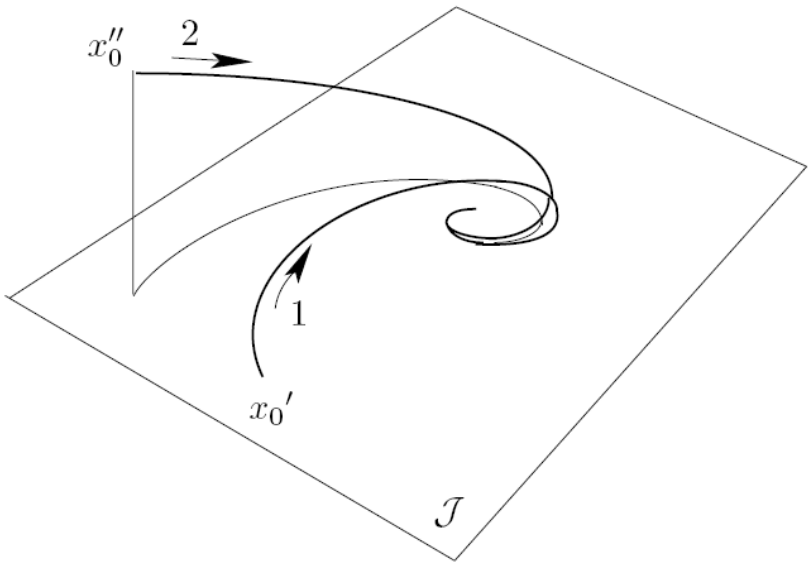

Figure 3.1: Internal and external stability of an invariant subspace.

$z_{02} \neq 0$; the time evolution of the second component of the transformed state is described by

$$z_2(t+1) = \tilde{A}_{22}z_2(t), \quad z_2(0) = z_{02}. \tag{3.10}$$

This means that the projection of the state along $\mathbf{J}$ on any complement of $\mathbf{J}$ has a stable behavior if and only if $\tilde{A}_{22}$ is a stable matrix. In other words, in this case the canonical projection of the state on the quotient space $\mathbf{X}/\mathbf{J}$ tends to the origin as t approaches infinity: this means that the linear variety parallel to $\mathbf{J}$, which contains the state, tends to coincide with $\mathbf{J}$ for t approaching infinity. By definition, an A-invariant $\mathbf{J} \subseteq \mathbf{X}$ is said to be internally stable if $A|_{\mathbf{J}}$ is stable, externally stable if $A|_{\mathbf{X/J}}$ is stable.

The concepts of internal and external stabilizability of A-invariants, introduced referring to the asymptotic behavior of trajectories of linear free dynamic systems, can be extended to controlled and conditioned invariants. Clearly, an $(A, \mathbf{B})$ – controlled invariant subspace $\mathbf{V}$ is internally stabilizable if, for any $x(0) \in \mathbf{V}$, there exists at least one

admissible trajectory of the pair (A, B) belonging to $\mathbf{V}$ and converging to the origin. The external stabilizability of $\mathbf{V}$ is guaranteed if for any $x(0) \in \mathbf{X}$ there exists at least one trajectory of the pair (A, B) converging to the subspace $\mathbf{V}$. If we find a matrix F which turns the $(A, \mathbf{B})-$ controlled invariant into an $A + BF$ invariant, then the internal and external stabilizability are given by the property of the matrix $A + BF$. In particular, if and only if there exists a real matrix F such that $(A + BF)\mathbf{V} \subseteq \mathbf{V}$, with $A + BF$ stable, then the controlled invariant V is both internally and externally stabilizable. By duality, an $(A, \mathbf{C})-$ conditioned invariant $\mathbf{S}$ is both internally and externally stabilizable if and only if there exists at least one real matrix G such that $(A + GC)\mathbf{S} \subseteq \mathbf{S}$, with $A + GC$ stable.

3.2 Unknown Input Observation of a Linear Function of the State

We consider in this section the problem of asymptotically estimate a linear function of the state when the system is affected by some unknown inputs. Consider the linear discrete time system

$$x(l+1) = Hx(l) + Bu(l), \tag{3.11}$$
$$y(l) = Cx(l), \tag{3.12}$$

and recall the structure of the Luenberger observer

$$z(t+1) = (A + GC)z(t) + Bu(t) - Gy(t). \tag{3.13}$$

Defining the estimation error as

$$e(t+1) = z(t+1) - x(t+1), \tag{3.14}$$

we obtain

$$e(t+1) = (A + GC)z(t) + Bu(t) - Gy(t) - Ax(t) - Bu(t) \tag{3.15}$$
$$= (A + GC)z(t) - GCx(t) - Ax(t) \tag{3.16}$$
$$= (A + GC)e(t). \tag{3.17}$$

It is clear that, if the matrix $(A + GC)$ is stable, the estimation error converges to zero so that $z(t) \to x(t)$. In the case the input $u(t)$ is unknown, the equation of the observer is

$$z(t+1) = (A + GC)z(t) - Gy(t). \tag{3.18}$$

and the estimation error becomes

$$e(t+1) = (A + GC)e(t) - Bu(t), \tag{3.19}$$

which does not converges to zero even if the matrix $(A+GC)$ is stable. Since the estimation error converges asymptotically to minimal $(A+GC)$-invariant containing $\mathrm{Im}(B)$, i.e., to the reachable set of (3.19), it is convenient to choose G to make this subspace of minimal dimension. The best choice corresponds to transforming into an $(A+GC)$-invariant the minimal externally stabilizable $(A,C)-$conditioned invariant containing $\mathrm{Im}(B)$. Let $\mathbf{S}$ be the minimal $(A+GC)$-invariant containing $\mathrm{Im}(B)$ and assume that $(A+GC)$ is stable. The observer provides an asymptotic estimate of the state modulo $\mathbf{S}$ or, in more precise terms, an asymptotic estimate of the state canonical projection on $\mathbf{X}/\mathbf{S}$ (similarly, the direct use of output without any dynamic observer would provide knowledge of state modulo $\mathbf{C}$, or its canonical projection on $\mathbf{X}/\mathbf{C}$). This incomplete estimate may be fully satisfactory if, for instance, it is not necessary to know the whole state, but only a given linear function of it: in this case the asymptotic estimate of this function is complete if and only if $\mathbf{S}$ is contained in its kernel (in the kernel of the matrix describing the function to be estimated). If the information about the system that is not reconstruct by the observer is retrieved through the direct output of the system, i.e., the estimation is obtained combining the output of the system and those of the observer and the unknown input is directly observable through the output of the system, then the whole state can be estimated. The following theorem summarize these considerations.

Theorem 3.2.1 (Unknown Input Observability)**.** *[23] Assume that the pair (A,C) is detectable, the problem of asymptotically estimating the whole state, in the presence of the unknown input u, has a solution if and only if*

- $\boldsymbol{S}^* \cap \mathrm{Ker}(C) = \emptyset$*;*
- $\boldsymbol{S}^*$ *is externally stabilizable.*

where where $\boldsymbol{S}^*$ is the minimal $(A, \mathrm{Ker}(C)) -$ conditioned invariant containing $\mathrm{Im}(B)$*.*

Chapter 4

Single Intruder Detection and Isolation

In order to implement an agreement algorithm, it should be ensured that the failure of one agent to perform its designated duties, or the presence of an intruder among the nodes, does not block the task completely. Some failure modes have been defined in [14] as follows:

1. stopping failure: the agent fails simply ceasing to communicate with other agents;
2. constant failure: the agent fails by setting its value to a constant value;
3. function failure: the agents alters the control input generating function setting its state to an arbitrary value at every time step. If the sequence is chosen maliciously, then this failure is referred as Byzantine failure.

Let $\{e_1, \dots, e_n\}$ be the canonical base of $\mathbb{R}^n$, the failure or the misbehavior of a node can be modeled as an external input, so that the system becomes

$$w(l+1) = Fw(l) + e_i\bar{u}(l), \quad l \in \mathbb{Z}_{\geq 0}. \tag{4.1}$$

The index i and the input $\bar{u}$ are unknown to every node, if the model describes a fault in the system, and to every node except i, if agent i is an intruder. Because of the presence of an exogenous input $\bar{u}$, the network is not able to achieve agreement.

The objective is for each node to detect and isolate the misbehaving nodes using only the information flow adopted by standard averaging protocols.

4.1 Solvability Conditions

Given a linear averaging algorithm with weight matrix F, we associate the linear discrete time system Σ

$$w(l+1) = Fw(l) + e_i\bar{u}(l) \tag{4.2}$$

$$y_j = C_j w(l), \tag{4.3}$$

and we denote it with $\Sigma_{ij} = (F, e_i, C_j)$. The matrix C_j denotes the information about the status of the network available to the agent j, where $C_j = [e_{k_1} \cdots e_{k_p}]^T, \quad k_1, \ldots, k_p \in N_j, \quad p \in \mathbb{Z}_{\geq 0}$.

Recall that a discrete system described by matrices (A, B, C) is *unknown input observable* (UIO), if it is possible to estimate the whole state in the presence of the unknown input $\bar{u}$, then

Theorem 4.1.1. (Unknown input observability for averaging systems): *Given a linear averaging algorithm, the system Σ_{ij} is UIO, for all $(i, j) \in \{1, \ldots, n\}$, if and only if the associated digraph is strongly connected and $i \in N_j$.*

Proof. Since $i \in N_j$, the minimal $(F, \mathbf{C}_j) -$ conditioned invariant containing $\mathbf{e}_i$ coincides with $\mathbf{e}_i$. In fact $F(\mathbf{e}_i \cap \mathbf{C}_j) = F(\emptyset) = \mathbf{0}$, and hence the first condition of Theorem 3.2.1 is verified. The external stabilizability of $\mathbf{S}^*$ is provided by finding a matrix G such that $(F + GC_j)\mathbf{e}_i \subseteq \mathbf{e}_i$, with $(F + GC_j)$ stable. Being $i \in N_j$, we can choose G in order to nullify the i_{th} column of F, so that

$$(F + GC_j)\mathbf{e}_i = \mathbf{0}. \tag{4.4}$$

Because of the connectivity of the associated digraph, F is irreducible, and since $F \geq| F+GC_j |$, $F \neq| F+GC_j |$, we have $\rho(F+GC_j) < \rho(F) = 1$, so that the matrix $(F+GC_j)$ is stable [15]. Because of the stability of $(F + GC_j)$, the pair (F, C_j) is detectable [3].

Suppose now that Σ_{ij} is UIO, then $\mathbf{S}^* \cap \mathbf{C}_j = \emptyset$, and $\mathbf{e}_i \subseteq \mathbf{S}^*$. We deduce $\mathbf{e}_i \subseteq \mathrm{Im}(C_j)$, and thus $i \in N_j$. Furthermore, if the digraph associated with F is not strongly connected, then there is at least a node j that does not receive information from a partition of the network, and that node obviously can not estimate the whole state of the system, even if the unknown input is constantly zero. We conclude that the digraph associated with F is strongly connected. □

We shall consider now the problem of isolating the misbehaving node, in order to guarantee that the task is accomplished at least by all working agents of the network. Once a node has detected an intruder, it simply ceases keeping into consideration the information coming from that agent, and adjusts the weights of the remaining incoming messages. The resulting matrix is row stochastic, and thus describes an averaging algorithm that converges to an agreement configuration, if the associated digraph is connected. Note that only the neighbors of the faulty node modify the topology of the network, so that no communication among the agents is needed to detect, identify and isolate the misbehaving node. The following Theorem formalizes these considerations.

Theorem 4.1.2. (Convergence in 2-connected faulty networks): *Let Σ_{ij} be an UIO system. If the associated digraph is 2-connected, then there exists $M \in \mathbb{R}^{n \times n}$, with entries $m_{rk} = 0$ if $k \notin N_i$, $r \in \{1, \ldots, n\}$, such that the algorithm*

$$w(l + 1) = (F + M)w(l) + e_i\bar{u}(l) \tag{4.5}$$

achieves agreement for all w_r, $r \neq i$, and for all possible trajectories $\bar{u}$.

Proof. Choose M such that $(F+M)_{r,i} = 0$ and $(F+M)_{r,r} = F_{r,r} + F_{r,i}$, if $r \in N_i \setminus \{i\}$. Since $(F+M)_{r,i} = 0$ for all $r \neq i$, the unknown input does not affect the variables w_r, $r \neq i$. Moreover the submatrix obtained deleting the i_{th} row and column from $(F+M)$ is, by construction, row stochastic and its associated digraph is strongly connected, since the one associated with F is 2-connected. These conditions are sufficient for achieving agreement among the variables w_r, $r \neq i$. □

The filter can be designed in many ways. We could be interested in minimizing the convergence rate of the estimation process, by placing the eigenvalues of the filter as close as possible to the origin. However, given an UIO system Σ_{ij}, the dimension of the observability subspace depends on the topology of the network, so that it is not always possible to place all the eigenvalues of the filter. For this reason, we describe a design that is applicable to every topology of network, and then we investigate its convergence rate.

Theorem 4.1.3 (Filter design)**.** *Let Σ_{ij} be an UIO system. If a filter is designed as*

$$\begin{aligned} z(l+1) &= (F + GC_j)z(l) - Gy_j(l), \\ \tilde{w}(l) &= Lz(l) + Ky_j(l), \end{aligned}$$

with

$$G = -F_{N_j}, \quad K = C_j^T, \quad L = I_n - KC_j,$$

being F_{N_j} the columns of F with indexes N_j, then

$$\tilde{w}(l) \to w(l), \tag{4.6}$$

as $l \to +\infty$, for all possible input trajectories $\bar{u}$.

Proof. Consider the equation of the estimation error,

$$\begin{aligned} r(l+1) &= z(l+1) - w(l+1) && (4.7)\\ &= (F + GC_j)z(l) - Gy_j(l) - Fw(l) - e_i\bar{u}(l) && (4.8)\\ &= (F + GC_j)z(l) - GC_jw(l) - Fw(l) - e_i\bar{u}(l) && (4.9)\\ &= (F + GC_j)r(l) - e_i\bar{u}(l). && (4.10) \end{aligned}$$

We choose $G = -F_{N_j}$, in order to nullify the N_j columns of F. Using the same procedure as in Theorem 4.1.1, we note that the matrix $(F + GC_j)$ is stable, and the reachable set of r is the minimum $(F + GC_j)$ invariant containing $\mathbf{e}_i$. Since

$$(F + GC_j)\mathbf{e}_i = \mathbf{0}, \tag{4.11}$$

the reachable set of the error r is $\mathbf{e}_i$. Let $K = C_j^T$, $L = I_n - KC_j$, and consider the estimated function $\tilde{w}$:

$$\tilde{w}(l) = Lz(l) + KC_j w(l) \tag{4.12}$$
$$= L(w(l) + r(l)) + KC_j w(l) \tag{4.13}$$
$$= (KC_j + L)w(l) + Lr(l) = w(l) + Lr(l). \tag{4.14}$$

Being $\mathbf{e}_i \subset \mathrm{Ker}(L)$, the term $Lr(l)$ will converge to zero, so that, as $l \to +\infty$,

$$\tilde{w}(l) \to w(l). \tag{4.15}$$

□

4.2 Convergence Rate Analysis

Given an $n \times n$ matrix F describing an agreement algorithm, construct the $n \times n$ matrix $\tilde{F}$ such that $\tilde{F}_{r,r} = 1$, $\tilde{F}_{r,k} = 0$, and $\tilde{F} = F$ in all the other entries, being $r \in N_j$, and $k \in \{1, \ldots, n\}$. Compute now the matrix $F + GC_j$ as described in Theorem 4.1.3, and recall that its largest eigenvalue is positive and smaller than 1. Let $\lambda_{\max}$ be the largest eigenvalue of $F + GC_j$, and $\lambda_2 = \max \lambda(\tilde{F})$, where the maximum is taken over all eigenvalues λ of $\tilde{F}$ different than 1. We note that $\tilde{F}$ and $F + GC_j$ are block diagonal matrices, so that their eigenvalues are the union of the eigenvalues of the two blocks. The first block is respectively the identity and the zero matrix of dimension n_0, being n_0 the cardinality of the set N_j, and the other block is in common in the two matrices. We conclude that $\lambda_{\max} = \lambda_2$. Let

$$\tilde{F}x_2 = \lambda_2 x_2. \tag{4.16}$$

Since $\lambda_{\max} = \lambda_2 < 1$, the N_j components of x_2 must be equal to 0, while the other entries are positive (Perron Frobenius Theorem). The eigenvalue λ_2 can be computed as

$$\lambda_2 = \max(\langle \tilde{F}x, x \rangle), \tag{4.17}$$

subject to $\langle x, x \rangle = \sum_{k=1}^{n} x_k^2 = 1$, and $x_{N_j} = 0$. We first analyze the convergence rate of the filter applied to the equal - neighbor consensus problem, and then we present a more general result valid for any kind of averaging algorithm.

4.2.1 Equal - Neighbor Case

We denote with equal - neighbor consensus algorithm the agreement procedure, when the graph is un-weighted and un-directed, so that the entry (i, j) of the adjacency matrix is equal to 1, if there is a directed link from j to i, 0 otherwise. In this case the iteration matrix F of the consensus algorithm can be expressed as

$$F = (I_n + D)^{-1}(I_n + A), \tag{4.18}$$

where D, A are, respectively, the out - degree and the adjacency matrices of the graph. Consequently, all the entries of $F_{k,j}$ are equal to $1/(d_{\text{out}}(k)+1)$. The iteration matrix of the filter is

$$F + GC_j = F - [\mathbf{0} \;\; F_{N_j} \;\; \mathbf{0}], \tag{4.19}$$

being F_{N_j} the N_j columns of F. The graph associated with $(F+GC_j)$ is obtained from the one associated with F, after deleting the edges from the node N_j to every other node of the network. A trivial lower bound for $\lambda_{\max}$ is obtained from Perron Frobenius Theorem:

$$\lambda_{\max} \geq 1 - \frac{1}{\Delta}, \tag{4.20}$$

where Δ is the maximum vertex degree, while an upper bound must be investigated with other techniques, because $\| F \|_\infty = \| F + GC_j \|_\infty = 1$, for all agreement algorithms of dimension $n > 3$.

Theorem 4.2.1 (Convergence rate)**.** *Let G_r be the digraph associated with an UIO system Σ_{ij}. Let n be the number of vertices, Δ the maximum out-degree including self-loops, and d the diameter of the digraph, then*

$$\lambda_{\max} \leq 1 - \frac{1}{nd\Delta}. \tag{4.21}$$

Proof. Construct $\tilde{F}$ as previously described, we use the methods of [20] to find an upper bound for $\lambda_2(\tilde{F})$. The problem

$$\lambda_2 = \max(\langle \tilde{F}x, x \rangle), \tag{4.22}$$

subject to $\sum_{k=1}^{n} x_k^2 = 1$ and $x_{N_j} = 0$, can be rewritten as

$$\lambda_2 = \max(\langle Ax, x \rangle), \tag{4.23}$$

subject to $\sum_{k=1}^{n} d_k x_k^2 = 1$ and $x_{N_j} = 0$, being d_k the out-degree of vertex k, and A the adjacency matrix, both including self-loops. Our next step is to rewrite $\langle Ax, x \rangle$ in a more revealing form. By expanding $\langle Ax, x \rangle$, we find

$$\langle Ax, x \rangle = \sum_{t=1}^{n} x_t \left(\sum_{k \in N_j} x_k \right) = \sum_{(t,k) \in E} x_t x_k, \tag{4.24}$$

where E denotes the set of all the edges of the graph. Moreover, we have

$$\sum_{(t,k)\in E} x_t x_k = \frac{1}{2}\sum_{(t,k)\in E} 2x_t x_k \tag{4.25}$$

$$= \frac{1}{2}\sum_{(t,k)\in E} x_t^2 + x_k^2 - (x_t - x_k)^2 \tag{4.26}$$

$$= \frac{1}{2}\left(\sum_{t=1}^{n} 2d_t x_t^2 - \sum_{(t,k)\in E}(x_t - x_k)^2\right) \tag{4.27}$$

$$= 1 - \frac{1}{2}\sum_{(t,k)\in E}(x_t - x_k)^2. \tag{4.28}$$

On combining this with (4.23), we find

$$\lambda_2 = 1 - \frac{1}{2}\min \sum_{(t,k)\in E}(x_t - x_k)^2, \tag{4.29}$$

subject to $\sum_{k=1}^{n} d_k x_k^2 = 1$ and $x_{N_j} = 0$. Let x_m denote the component of x which is largest in magnitude, and let $\Delta = \max_k d_k$. Since $d_k \leq \Delta$, we find

$$1 = \sum_{k=1}^{n} d_k x_k^2 \leq n\Delta x_m^2, \tag{4.30}$$

so that

$$x_m \geq (n\Delta)^{-\frac{1}{2}}. \tag{4.31}$$

Since $x_{N_j} = 0$, we have $x_m = x_m - x_s \geq (n\Delta)^{-\frac{1}{2}}$, being $s \in N_j$. Given the connectivity of the graph, there is a sequence of vertices of length $r \leq d$, which joins vertex m to s. Letting $\{x_{k_1} \dots x_{k_r}\}$ denote the set of vertices traversed by this chain, we have

$$(n\Delta)^{-\frac{1}{2}} \leq (x_m - x_s) = (x_m - x_{k_1}) + \cdots + (x_{k_r} - x_s). \tag{4.32}$$

The inequality can be rewritten as

$$(n\Delta)^{-\frac{1}{2}} \leq (x_m - x_{k_1})1 + \cdots + (x_{k_r} - x_s)1, \tag{4.33}$$

and by Cauchy's inequality

$$(n\Delta)^{-1} \leq \frac{r}{2}\sum_{(t,k)\in E}(x_t - x_k)^2 \leq \frac{d}{2}\sum_{(t,k)\in E}(x_t - x_k)^2. \tag{4.34}$$

Combining (4.34) and (4.29) shows that

$$\lambda_2(\tilde{F}) \leq 1 - \frac{1}{nd\Delta}. \tag{4.35}$$

□

4.2.2 Upper Bound for Weighted Digraph

The matrix $\tilde{F}$ is an absorbing Markov chain, with absorbing nodes N_j. Given an arbitrary absorbing Markov chain, renumber the states so that the transient states come first. If there are r absorbing states and t transient states, then the transition matrix will have the following canonical form

$$\tilde{F}_c = \begin{bmatrix} Q & R \\ 0 & I \end{bmatrix}, \tag{4.36}$$

where I is an $r \times r$ identity matrix, $\mathbf{0}$ is an $r \times t$ zero matrix, R is a nonzero $t \times r$ matrix, and Q is an $t \times t$ matrix. The first t states are transient and the last r states are absorbing. A standard matrix algebra argument shows that $\tilde{F}_c^l$ is of the form

$$\tilde{F}_c^l = \begin{bmatrix} Q^l & \left(\sum_{j=0}^{l-1} Q^j\right) R \\ 0 & I \end{bmatrix}. \tag{4.37}$$

The form of $\tilde{F}_c^l$ shows that the entries of Q^l give the probabilities for being in each of the transient states after l steps, for each possible transient starting state. It has been proven that $Q^l \to \mathbf{0}$ as $l \to +\infty$, and we say that the process is absorbed with probability 1.

Theorem 4.2.2. (Convergence rate of an absorbing Markov chain): *Let $\tilde{F} \in \mathbb{R}^{n \times n}$ be the transition matrix of an absorbing Markov chain, and let x_{N_j} be the absorbing states of the chain. Let G_r be the associated weighted digraph, such that the entry $\tilde{F}_{k,j}$ represents the weight of the edge from k to j. Let d be the diameter of G_r, and Δ the maximum vertex degree including self-loops, then*

$$\lambda_2(\tilde{F}) < \left(1 - \frac{\bar{n}}{\Delta^d}\right)^{1/d}, \tag{4.38}$$

being $\bar{n}$ the cardinality of the set $N_j \setminus \{j\}$.

Proof. Let $\tilde{F}_c$ be the canonical form of $\tilde{F}$. Recall that the largest eigenvalue of $\tilde{F}_c$ is 1, with multiplicity n_0. Consequently $\lambda_2(\tilde{F}_c) = \lambda_{\max}(Q)$, where Q is the transient matrix of $\tilde{F}_c$. Recall that

$$\lambda(Q) \leq \rho(Q) \leq \|Q^k\|^{1/k}, \quad k = 1, 2, \ldots, \tag{4.39}$$

being ρ the spectral radius of the matrix.

The entry (j, i) of $\tilde{F}_c^k$ gives the probability to be in the node i after k steps, being started from node j. Suppose we start a random walk on G_r from the node k more distant to one of the nodes belonging to N_j: it takes at most d steps before reaching a node belonging to N_j, with probability greater or equal to $1/\Delta^d$. This means that

$$(\tilde{F}_c^d)_{k,s} \geq \frac{1}{\Delta^d}, \quad s \in N_j \setminus \{j\} \tag{4.40}$$

that leads to

$$\rho(Q) \leq \|\tilde{Q}^d\|_\infty^{1/d} \leq \left(1 - \frac{\bar{n}}{\Delta^d}\right)^{1/d}, \tag{4.41}$$

being $\bar{n} = n_0 - 1$. The difference between n_0 and $\bar{n}$, is due to the fact that we need to reach a node of $N_j \setminus \{j\}$ before reaching j. Since nodes of $N_j \setminus \{j\}$ are absorbing states, it is not possible to reach j, so that the j_{th} column of $\tilde{F}_c^k$ is $\mathbf{0}$ for every k. □

Remark 3. *Given an UIO system Σ_{ij}, if the pair (F, C_j) is observable, then all the eigenvalues of the matrix $(F + GC_j)$ can be assigned, i.e., the estimate of the current state can be arbitrarily fast. The complete assignability of the eigenvalues of the observer is feasible for instance for a regular directed cycle graph of n nodes, where $N_j = \{j, j-1\}$. Consider in fact a random walk on the graph starting from node j, it takes $n-1$ steps to reach node $j-1$. The walk can be represented by the Krilov sequence generated by the transpose of the agreement algorithm iteration matrix F and e_j, i.e.,*

$$S = [e_j \;\; F^T e_j \;\; (F^T)^2 e_j \ldots (F^T)^{n-1} e_j]. \tag{4.42}$$

Since the entry $s_{kl} > 0$ only if the node k can be reached in l steps, and since we only visit a new node at each time step, the matrix S has full rank. The rank of the matrix S^T coincides with the dimension of the observability subspace of the pair (F, e_j), so that we conclude that the system is observable.

4.3 Intrusion Detection Procedure

By analyzing the property of the iteration error, we describe a methodology to detect and identify a misbehavior for an UIO system. The working conditions of the network are then restored through the exclusion of the faulty agent, i.e., by modifying the topology of the network, such that the information he provides is not considered.

4.3.1 Analysis of the Iteration Error

Consider the linear discrete time UIO system described by the algorithm

$$w(l+1) = Fw(l) + B_i\bar{u}(l), \tag{4.43}$$
$$y_j(l) = C_j w(l), \tag{4.44}$$

and the filter built by the node j

$$z(l+1) = (F + GC_j)z(l) - Gy_j(l), \tag{4.45}$$
$$\tilde{w}(l) = Lz(l) + Ky_j(l). \tag{4.46}$$

Reorder the states variables, such that index set N_j comes first. The filter can be rewritten as

$$z(l+1) = F \begin{bmatrix} w_{N_j}(l) \\ z_p(l) \end{bmatrix}, \tag{4.47}$$

$$\tilde{w}(l) = \begin{bmatrix} w_{N_j}(l) \\ z_p(l) \end{bmatrix}, \tag{4.48}$$

with $p \notin N_j$. Consider the iteration error

$$\varepsilon(l) = \mid \tilde{w}(l+1) - F\tilde{w}(l) \mid \tag{4.49}$$

$$= \left| \begin{bmatrix} w_{N_j}(l+1) \\ z_p(l+1) \end{bmatrix} - F \begin{bmatrix} w_{N_j}(l) \\ z_p(l) \end{bmatrix} \right|. \tag{4.50}$$

Recall that $L = I_n - K$, and $K = C_j^T C_j$; we can rewrite ε as

$$\varepsilon(l) = \left| LF \begin{bmatrix} w_{N_j}(l) \\ z_p(l) \end{bmatrix} + Kw(l+1) - F \begin{bmatrix} w_{N_j}(l) \\ z_p(l) \end{bmatrix} \right| \tag{4.51}$$

$$= \left| \begin{bmatrix} [w(l+1) - F\tilde{w}(l)]_{N_j} \\ \mathbf{0} \end{bmatrix} \right| \tag{4.52}$$

$$= \left| \begin{bmatrix} \bar{\varepsilon} \\ \mathbf{0} \end{bmatrix} \right|. \tag{4.53}$$

Since $\tilde{w}(l) \to w(l)$ as $l \to +\infty$, $(\tilde{w}(l+1) - F\tilde{w}(l)) \to B_i\bar{u}(l)$, so that we can detect and identify the intruder. The following three cases are possible:

1. there is no unknown input in the system. In this case $\| \bar{\varepsilon}(l) \| \to 0$, as $l \to +\infty$;
2. there is a faulty node i and $i \in N_j$. In this case the i_{th} component of $\bar{\varepsilon}(l) \to \bar{u}(l)$, while the other components converge to zero as fast as the convergence rate of the filter;
3. there is a faulty node i and $i \notin N_j$. In this case $\bar{\varepsilon}$ do not converge to zero in more than 1 component. In fact, the estimation error is $r(l+1) = (F + GC_j)r(l) - B_i\bar{u}(l)$, and its reachable set is not included into $\mathrm{Ker}(L)$. We deduce $\tilde{w}(l) = w(l) + Lr(l)$ does not converge to $w(l)$.

4.3.2 Intrusion Detection Algorithm

Consider the iteration error $\bar{\varepsilon}$, we have

$$\bar{\varepsilon}(l) \leq \lambda_{\max}^l \varepsilon_0, \quad l \in \mathbb{Z}_{\geq 0}, \tag{4.54}$$

where $\varepsilon_0 = \max(\mid \tilde{w}(1) - F\tilde{w}(0) \mid)$. The iteration error can be written as

$$\varepsilon(l) = e(l) + e_u(l), \tag{4.55}$$

where e denotes the error due to the estimation process, while e_u is due to the unknown input. If

$$e(l) + e_u(l) \geq \lambda^l_{\max}, \tag{4.56}$$

then we can detect a fault in the system. Since $\lambda^l_{\max}$ vanishes as $l \to +\infty$, the recognizable misbehavior becomes smaller with l. The amount of time $\bar{l}$ we need to wait to ensure a correct estimation of the misbehavior, depends on the minimal unknown input we want to recognize and on $\lambda^l_{\max}$. We do not specify this parameter, and we consider that after $\bar{l}$ steps the estimation error is small enough to identify the intruder. Note that this problem does not arise, if we consider that the initial conditions of the nodes are known to all the network, or if the unknown input enters the system when the estimation error is small enough. A possible procedure is:

1. each node j builds an observer as described in Theorem 4.1.3;
2. each agent computes $\bar{\varepsilon} =\mid \tilde{w}(l+1) - F\tilde{w}(l) \mid, l \geq \bar{l}$; calling $\bar{\varepsilon}_k$ the k_{th} variable of the vector $\bar{\varepsilon}$, and letting $\delta = \lambda^l_{\max}$,
   (a) if $\bar{\varepsilon}_k < \delta$ for all $k \in N_j$, then no fault is detected;
   (b) if $\bar{\varepsilon}_k < \delta$ for all $k \in N_j \setminus \{i\}$, then the i_{th} agent is an intruder;
   (c) if $\bar{\varepsilon}_k \geq \delta$ for more than one $k \in N_j$, then there is a misbehavior in the network, but it is does not belong to N_j (see **??**).
3. while no intruder is found, step 2 is repeated. Otherwise, as soon as agent j detects that agent i is an intruder, agent j changes the topology of the network according to Theorem 4.1.2.

If the original network was k-connected, with $k > 2$, then all the neighbors of the faulty node can propagate the new topology over the remaining network, so that each agent (after that the new topology has been validated by all the neighbors of the intruder) starts the procedure from step 1.

Chapter 5

Multi Intruders Detection and Isolation

In this section we consider the problem of intrusion detection for a sensor network introducing some reputation algorithms and distributed procedures. We address in particular two problems: the consensus problem in the presence of many intruders, and the maximum likelihood distributed estimation of a parameter, which requires a consensus on the exact average of the initial values, still in the presence of many intruders. Reputation mechanisms are adopted to avoid dealing with filters and matrices whose dimensions grow with the number of nodes. In Section 4 we describe a filter which gives an exact estimation of the network state under certain conditions, but the dimensions of the filter depend on the number of nodes, that is not desiderate. A better scenario is the one in which intruders are identified and isolated by their neighbors, while distant nodes not even feel the presence of those faulty processors. The described Unknown Input Filter is not suitable for such task, since it requires a long time to converge, so that the error propagates in the network and every node feels the presence of the intruders. On the other side, the iteration error of the Unknown Input Filter is a good parameter to judge the behavior of a node. Consider the consensus network shown in Fig. 5.1, in which nodes are disposed and connected as a two dimensional grid, and suppose that every node runs an Unknown Input Filter and computes the iteration error, whose components can be interpreted as the reputation of the corresponding node. Collect the iteration errors of all the nodes, and plot the largest value for every node. We obtain the result of Fig. 5.2, where the picks correspond to estimation error referring to the faulty processors. It is clear that, if a central unit had the estimation error of every node, then it could be easy for it to detect the intruders. Since we do not consider central unit, we will try to reconstruct that surface in a distributed way, in order to identify and isolate the intruders.

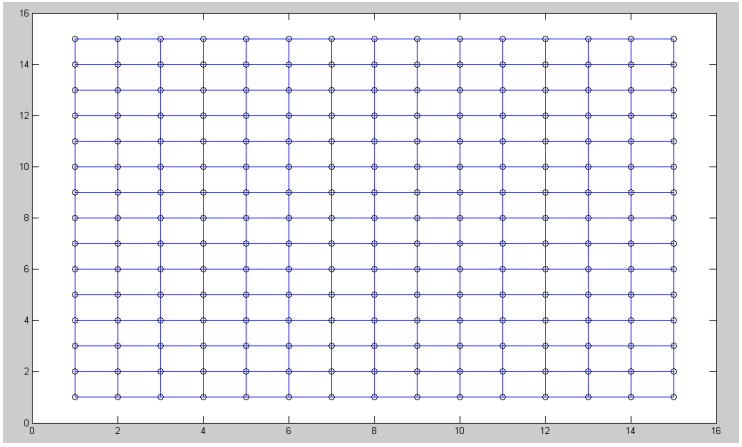

Figure 5.1: Consensus network.

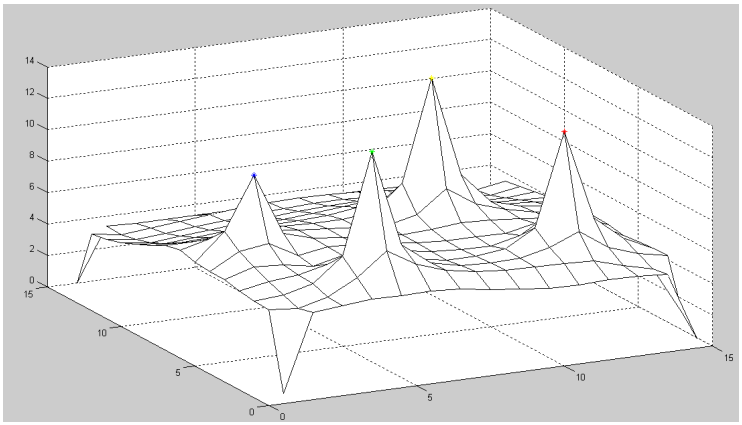

Figure 5.2: Iteration error.

5.1 An Exact Solution

We describe in this section some properties of the iteration error in a consensus network. Some of these results may be used to develop algorithms able to solve the consensus problem for a connected sensor network in the presence of several intruders, under a set of assumptions on the possible input function $\bar{u}$.

Consider the system described in Section 4, and suppose there are $p > 1$ intruders. The system is described by

$$w(l+1) = Fw(l) + B\bar{u}(l), \quad B = [e_{k_1} \cdots e_{k_p}], \quad k_1, \ldots, k_p \in \{1, \ldots, n\}. \tag{5.1}$$

The UIO property can be generalized to the case of many faulty nodes, as follows

Corollary 5.1.1. *Consider the system described by*

$$w(l+1) = Fw(l) + B\bar{u}(l), \quad l \in \mathbb{Z}_{\geq 0}, \tag{5.2}$$

with $B \subseteq \mathbb{R}^{n\times p}$ defined as

$$B = [e_{k_1} \cdots e_{k_p}], \quad k_1, \ldots, k_p \in \{1, \ldots, n\}. \tag{5.3}$$

the system (F, B, C_j) is UIO, if and only if $\boldsymbol{B} \subseteq \mathrm{Im}(C_j)$, and the digraph associated with F is strongly connected.

Proof. We use the same strategy as in Theorem 4.1.1. If $\mathbf{B} \subseteq \mathrm{Im}(C_j)$, then we have $\mathbf{B} \cap \mathbf{C}_j = \emptyset$, that is, $\mathbf{S}^* = \mathbf{B}$. Hence, $\mathbf{S}^* \cap \mathbf{C}_j = \emptyset$. We compute G in order to nullify the k_{th} columns of F, such that

$$(F + GC_j)\mathbf{B} = \mathbf{0}. \tag{5.4}$$

The matrix $(F + GC_j)$ is stable, because of the connectivity of the graph, and because $F \geq| F + GC_j |$, $F \neq| F + GC_j |$, and therefore the system is detectable. Now, if the system is UIO, then $\mathbf{S}^* \cap \mathbf{C}_j = \emptyset$, so that $\mathbf{B} \subseteq \mathrm{Im}(C_j)$. Moreover, being the system UIO, it is also observable, so that the associated digraph is strongly connected. □

Theorem 5.1.1 asserts that, for the UIO property, all the intruders must be neighbors of all their neighbors, i.e., $\mathrm{Im}(B) \subseteq \mathrm{Im}(C_j)$, $B = [e_{k_1}, \ldots, e_{k_p}]$, $\forall j \in \{N_{k_1} \cup \cdots \cup N_{k_p}\}$. This condition is not desiderate because it forces the network to have an huge number of edges. If this condition is not satisfied, then the iteration error does not converges to zero, and the estimation of the network status in not correct.

However the iteration error can be used to retrieve information about the position of the intruders in the network, as it will be explained in the next sections.

5.1.1 Absorbtion Probability and Iteration Error

Consider a consensus network formed by $n-p$ nodes and p intruders $I_k = \{i_{k_1}, \ldots, i_{k_p}\}$. Let $j \notin I_k$, and consider the system Σ_{ij} related to the consensus algorithm

$$w(l+1) = Fw(l) + B\bar{u}(l), \tag{5.5}$$

$$y_j(l) = C_j w(l), \tag{5.6}$$

where $B = [e_{k_1} \cdots e_{k_p}]$, $k_1, \ldots, k_p \in \{1, \ldots, n\}$. Suppose that $\mathrm{Im}(B) \nsubseteq \mathrm{Im}(C_j)$, so that Corollary 5.1.1 can not be applied. Recall that the iteration error gives information only about the N_j components, then the term $B\bar{u}(l)$ is not visible in $\varepsilon(l)$. Hence we have

$$\varepsilon(l) = \left| \sum_{\tau=0}^{l-1} (F + GC_j)^{l-\tau} B\bar{u}(\tau) \right|. \tag{5.7}$$

Suppose now that $\bar{u}$ is a constant signal, we can rewrite $\varepsilon(l)$ as

$$\varepsilon(l) = \left| \sum_{\tau=0}^{l-1} (F + GC_j)^{l-\tau} \right| |B\bar{u}|. \tag{5.8}$$

Reorder the state variables so that the index set N_j comes first, the structure of $F + GC_j$ becomes the following:

$$F + GC_j = \begin{bmatrix} \mathbf{0} & R \\ \mathbf{0} & Q \end{bmatrix}, \tag{5.9}$$

and

$$\varepsilon(l) = \left| \sum_{\tau=0}^{l-1} (F + GC_j)^{l-\tau} \right| |B\bar{u}| = \left| \sum_{\tau=0}^{l-1} \begin{bmatrix} \mathbf{0} & R \\ \mathbf{0} & Q \end{bmatrix}^{l-\tau} \right| |B\bar{u}| \tag{5.10}$$

$$= \left| \sum_{\tau=0}^{l-1} \begin{bmatrix} \mathbf{0}^{l-\tau} & \sum_{j=0}^{l-\tau-1} \mathbf{0}^j R Q^{l-\tau-1-j} \\ \mathbf{0} & Q^{l-\tau} \end{bmatrix} \right| |B\bar{u}| \tag{5.11}$$

$$= \left| \sum_{\tau=0}^{l-1} \begin{bmatrix} \mathbf{0} & RQ^{l-\tau-1} \\ \mathbf{0} & Q^{l-\tau} \end{bmatrix} \right| |B\bar{u}|, \tag{5.12}$$

$$\tag{5.13}$$

so that

$$\varepsilon(l)_{N_j} = \left| \sum_{\tau=0}^{l-1} RQ^{\tau-1} \right| |B_p \bar{u}|, \quad p \notin N_j. \tag{5.14}$$

Consider now the reducible absorbing Markov chain defined as

$$\tilde{F} = \begin{bmatrix} I & R \\ \mathbf{0} & Q \end{bmatrix}, \tag{5.15}$$

where I is the identity matrix of dimension equal to the cardinality of the set N_j, and R, Q are of appropriate dimensions. The probability of being after l steps in the state j being started in the state i is given by the entry (j, i) of $\tilde{F}^l$. We have

$$\tilde{F}^l = \begin{bmatrix} I & \sum_{\tau=0}^{l-1} I^\tau R Q^{l-\tau-1} \\ \mathbf{0} & Q^l \end{bmatrix} \tag{5.16}$$

$$= \begin{bmatrix} I & R \sum_{\tau=0}^{l-1} Q^{l-\tau-1} \\ \mathbf{0} & Q^l \end{bmatrix}. \tag{5.17}$$

By comparing this expression with $\varepsilon(l)$, we note that, if $\bar{u} = 1$, $\varepsilon(l)_k = P_{k,i}$, $k \in N_j$, being $P_{k,i}$ the probability of being absorbed by the state k being started from node i, where i denotes an intruder in the system, i.e., the entry equal to 1 in B. If $\bar{u} = \text{const} \neq 1$, then $\varepsilon(l)_k = P_{k,i}\bar{u}$. Moreover, if $B = [e_{k_1} \cdots e_{k_p}]$, $\quad k_1, \ldots, k_p \in \{1, \ldots, n\}$, and $\bar{u} = [K_1 \cdots K_p]$ is a constant matrix, then $\varepsilon(l)_k = \sum_{j=1}^{p} P_{k,j} K_j$.

This property can be used to improve the performance of the single intruder detection, and to develop some algorithms for the multiple intruder case. For example, in the linear deployment problem (Section 6.2), we have that every node can say who the intruder is, by only analyzing the structure of the iteration error.

5.1.2 Constant Unknown Inputs

Given a consensus network, build at every node an Unknown Input Filter as described in Section 4, and define the absorbtion probability matrix associated with the filter iteration matrix as

$$P = R(I - Q)^{-1}, \tag{5.18}$$

where P, R, Q are defined above. We know that, if there are no intruders, the iteration error of each filter converges to zero, while converges to $\varepsilon(l)_k = \sum_{j=1}^{p} P_{k,j} K_j$ if there are p intruders that act with a constant disturb function K. To identify the intruders a node j has to find the matrix $\tilde{P}$, formed by p columns of the matrix

$$\tilde{P} = \left[\begin{array}{cc} P & I_n \end{array} \right], \tag{5.19}$$

such that

$$\text{rank}(\left[\begin{array}{ccc} \tilde{P} & \vdots & \varepsilon \end{array} \right]) = rank(\tilde{P}). \tag{5.20}$$

Let $\tilde{P} = [P_{i_1}, \vdots, P_{i_p}]$, where $\{i_1, \ldots, i_p\}$ are the indexes of the columns chosen to determine $\tilde{P}$, then $\{i_1, \ldots, i_p\}$ are the identifiers of the intruders in the networks. Depending on the topology of the network, there are many solutions to the problem, i.e., many matrices $\tilde{P}$ that can locally produce the iteration error ε. The more information we know about the inputs $\bar{u}$, the better our identification is, so that it is possible to assign a reputation (an index of reliability) to every node.

As drawbacks, this method requires strong assumptions, in particular it forces the external input to be constant, and it needs a lot of computation to identify the intruders.

5.2 Distributed Robust Averaging Algorithm

We describe in this section a possible procedure to compute the average in a sensor network in the presence of faulty processors. The method can be also used to reach consensus in the network, since the average consensus problem includes the simple consensus task.

Consider a sensor network with n nodes $V = \{v_1, \ldots, v_n\}$. A sensor in the network is a device with some memory, able to perform some computations and capable of transmitting values to its neighbors, i.e., a set of nodes within a certain distance. Each node in particular has a variable x_j containing the estimated parameter, a variable S_j representing the working state of the node, and an Unknown Input Filter Σ_j to detect possible intruders in the network. The following three cases are possible:

- S_j = TEST, if j is testing a neighbor;
- S_j = WAIT, if S_k = TEST, $k \in N_j$;

- S_j = RUN in the other situations.

When the algorithm starts there are no communication edges among the nodes, or equivalently, all the existing edges are secure, i.e., there are no intruders among them. Each node aims to build the edges towards its neighbors in order to compute the desired average. For security issues, a node can build an edge only if all its neighbors are in RUN state. Every time a link is created, all the nodes in TEST and WAIT mode modify their Unknown Input Filter by considering the new link and the node introduced in the network. If the added node is recognized as intruder, then it is excluded from the network, and it is not considered again, otherwise the WAIT and TEST nodes return to the normal operating state. Let R_j the set of nodes within communication range from the node v_j, N_j the set of actual neighbors of the node v_j, ε the iteration error vector, and δ a decision threshold. The set E indicates as usual the set of edges in the graph, and the operation $E+ = (v_j, v_h)$ means that the undirected edges form v_j to v_h is added to the network, and $R_j- = \{i\}$ means that the node v_i as been recognized as intruder by the node v_j and it is permanently excluded from the neighbors set R_j. We summarize the procedure as follows:

Algorithm 1 Secure network assembly for consensus computation

Each node j
while 1 **do**
  Send x_j, S_j to all its neighbors;
  Execute the consensus step (weighted average);
  if (The intruder i is detected) **then**
    if ($S_j ==$ TEST) **then**
      $N_j- = \{i\}$;
      Update the Unknown Input Filter;
      $S_j =$ RUN
    else
      $N_j- = \{i\}$;
      Update the Unknown Input Filter;
    end if
  end if
  if ($S_j ==$ TEST and $\|\varepsilon\| < \delta$) **then**
    $S_j =$ RUN;
  end if
  if ($S_k ==$ TEST, $k \in N_j$,) **then**
    $S_j =$ WAIT;
    Update the Unknown Input Filter;
  else
    if ($S_k ==$ RUN, for all $k \in N_j$ and $R_j \neq \emptyset$) **then**
      $S_j =$ TEST;
      $E+ = (v_j, v_h)$, $h \in R_j$;
      $N_j+ = \{i\}$;
      $R_j- = \{i\}$
      Update the Unknown Input Filter;
    else
      $S_j =$ RUN;
    end if
  end if
end while

Remark 4. *With the proposed algorithm only a finite error can enter the system. The average estimated by each node is*

$$\frac{1}{n}\left(\sum_{i=1}^{n}(x_i(0)) + \varepsilon\right), \tag{5.21}$$

where ε represents the error introduced in the system. Letting $n \to \infty$ we have

$$\lim_{n\to\infty}\frac{1}{n}\left(\sum_{i=1}^{n}(x_i(0)) + \varepsilon\right) = \frac{1}{n}\sum_{i=1}^{n}(x_i(0)). \tag{5.22}$$

5.2.1 Reduced Unknown Input Filter

The filter can be computed in several ways, depending upon the amount of information we have about the network. We suppose a node knows only the information it can retrieve from its neighbors, i.e., the degree and the state value. Each node j builds the Unknown Input Filter considering a two hops network, in which each node $k \in N_j$ has nodes $\{j, p_k\}$ as neighbors. The weight of the edges $\{k, p_k\}$ are chosen according to the degree of k in the real network. We call this network G_j. Fig. 5.3 explains the procedure when $j = 2$. The iteration matrix related to this network is by construction row stochastic, and, since

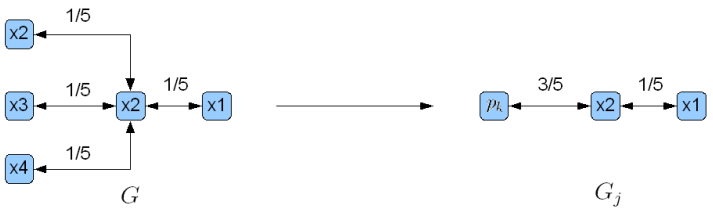

Figure 5.3: Filter weights.

the filter takes the its input from the real network, the iteration error converges to zero only in the absence of external inputs. In particular we define

$$\varepsilon(l) = \tilde{x}(l+1) - F_{filter}\tilde{x}(l) \tag{5.23}$$

where F_{filter} is the iteration matrix of the algorithm defined on G_j. Since ε converges to zero in the absence of faulty processors, the action of an intruder is forced to converges to zero as fast as the estimation error, otherwise the node corresponding to the biggest component in ε is recognized as faulty. The error an intruder can introduce in the system depends on the moment the unknown input enters the system. It is clear that the maximum disturb can be introduced during a transition, i.e., when a new link is created. Anyway, since the amount of error is finite, when the number of nodes n grows, the estimation algorithm converges to the exact average of the initial values.

5.2.2 Intrusion Detection and Filter Initialization

To assign the correct value to the state of the unknown input filter is of crucial importance, because, by reducing the initial estimation error, we reduce the amount of error that can enter the system without being detected. The problem of initializing the filter arises when an edge is added to the network. Consider for simplicity the situation of Fig. 5.4, in which the link $\{4, 5\}$ has been added. Every node has to modify its filter since the change

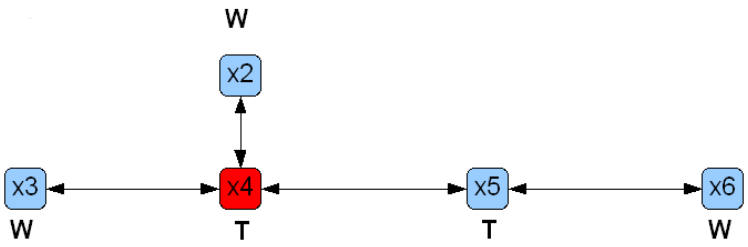

Figure 5.4: Filter initialization.

in the network has occurred within two hops of distance. Node 5 in particular add two variable to its filter, corresponding respectively to node 4 and to its neighbor p_4. Since the network where 4 belongs to is supposed to be at steady state, the value of p_4 in the filter is set as the one received by node 4. Node 6 add just a node corresponding to p_5, and sets its value to $\Delta/2$, where Δ indicates the range where variables x_j take value, since it has no information about the state value of node 4.

Fig 5.5 shows the propagation delay of the error when the intruder is the node number 4. It is clear that node number 5 detects the presence of the faulty node one step before node number 6, so that it can exclude the faulty processor and preserve the functionality of the network. Similar considerations can be done for the other nodes of the network.

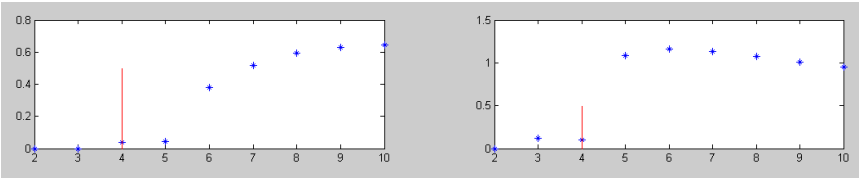

Figure 5.5: Error propagation: error detected by node 5 (right), and by node 6 (left).

5.2.3 Convergence Time

If we use two parallel passes of the agreement algorithm (Section 6.3) to perform the average consensus task, then, every time a new link is created, the weights are adjusted according to the equal neighbors model. It is known [2], that for the equal neighbor, time-invariant, bidirectional model, the convergence rate satisfies

$$\rho \leq 1 - \gamma_1 n^{-3}, \tag{5.24}$$

where γ_1 is a constant independent of n, and the convergence time satisfies

$$T_n(\varepsilon) = O(n^3 log(n/\varepsilon)). \tag{5.25}$$

Using this result we can estimate the convergence time of the proposed procedure when the number of nodes of the network approaches infinity. Since the algorithm builds the network incrementally, and since a new node can be added only when the network has reached the steady state, the averaging algorithm is executed every time a node is introduced in the network. The maximum number of executions of the averaging algorithm is $n-1$, because, starting from a node j and adding a node at a time, we need $n-1$ steps to introduce all the nodes. In this case we say that the procedure convergence time satisfies

$$T_n(\varepsilon) = O(n^4 log(n/\varepsilon)). \tag{5.26}$$

It is possible to execute the averaging algorithm a less than $n-1$ times, because in a large network the assemble process starts from many point far from each other, so that many links are created at each time step. Finding a tighter bound on the convergence time is one of the issue we are studying.

The equal neighbor consensus rule does not have the optimal convergence rate, but has the advantage that can be easily computed. Other simple averaging heuristics, as the Max-Degree or the Metropolis rules, produce in simulation a worse convergence rate [2].

Chapter 6

Applications

We present in this Section some numerical examples to show how the proposed procedures work. We consider in particular three problems: the Agreement Evaluation, where the nodes simply have to agree on a parameter, the Linear Deployment, where a group of mobile agents want to uniformly deploy on a circle, and finally an estimation problem, where a set of sensor want to evaluate the average of their measurements.

6.1 Agreement Evaluation

The agreement algorithm is an iterative procedure for the solution of the distributed consensus problem. Consider a set $N = \{1, 2, \ldots, n\}$ of nodes. Each node i starts with a scalar value $x_i(0)$; the vector with the values of all nodes at time t is denoted by $x(l) = (x_1(l), \ldots, x_n(l))$. The agreement algorithm updates $x(l)$ according to the equation $x(t + 1) = F(l)x(l)$, or

$$x_i(l + 1) = \sum_{j=1}^{n} f_{ij}(l)x_j(l), \tag{6.1}$$

where $F(l)$ is a nonnegative matrix with entries $f_{ij}(l)$. The row-sums of $F(l)$ are equal to 1, so that $F(l)$ is a stochastic matrix. In particular, $x_i(t + 1)$ is a weighted average of the values $x_j(l)$ held by the nodes at time t.

6.1.1 Numerical Example

Suppose we have 8 nodes disposed on a digraph as in Fig. 6.1 and suppose the node number 6 is the faulty node. Considering the node number 5 as observer, the system is described by

$$w(l + 1) = Fw(l) + B_6\bar{u}(l) \tag{6.2}$$

$$y_5(l) = C_5w(l), \tag{6.3}$$

where

$$F = \begin{bmatrix} 1/3 & 0 & 0 & 1/3 & 1/3 & 0 & 0 & 0 \\ 1/3 & 1/3 & 0 & 0 & 0 & 1/3 & 0 & 0 \\ 0 & 1/3 & 1/3 & 0 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 1/3 & 1/3 & 0 & 0 & 0 & 1/3 \\ 1/3 & 0 & 0 & 0 & 1/3 & 1/3 & 0 & 0 \\ 0 & 1/3 & 0 & 0 & 0 & 1/3 & 1/3 & 0 \\ 0 & 0 & 1/3 & 0 & 0 & 0 & 1/3 & 1/3 \\ 0 & 0 & 0 & 1/3 & 1/3 & 0 & 0 & 1/3 \end{bmatrix}, \tag{6.4}$$

$$B_6 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad C_5 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}. \tag{6.5}$$

Note that the conditions of Theorem 4.1.1 are verified, such that we can build a whole state unknown input observer, whose matrices are

$$G = \begin{bmatrix} -1/3 & -1/3 & 0 \\ -1/3 & 0 & -1/3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -1/3 & -1/3 & -1/3 \\ 0 & 0 & -1/3 \\ 0 & 0 & 0 \\ 0 & -1/3 & 0 \end{bmatrix}, \quad K = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{6.6}$$

$$L = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{6.7}$$

Let $\bar{u}$ = constant = 10, we initialize the state of the nodes with random variables, and we simulate the system for 40 steps. Fig. 6.2 shows the iteration error related to the estimation of the variables 1 and 6. As we see, the iteration error for the variable 6 converges to $\bar{u} = 10$, while the error for the variable 1 goes to zero, so that the observer can detect the fault, and identify the misbehaving node. The same procedure is applied

by the other neighbor of the faulty node, the agent 2, so that the intruder is isolated from the network (Fig. 6.3).

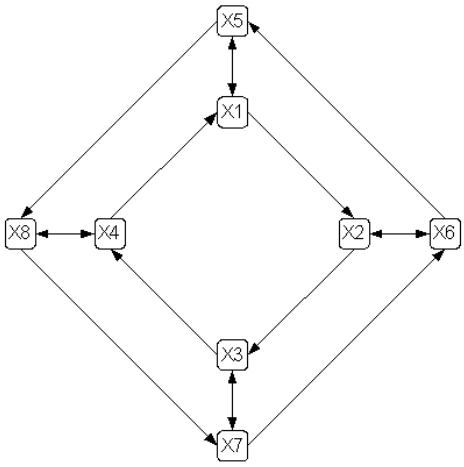

Figure 6.1: Consensus network.

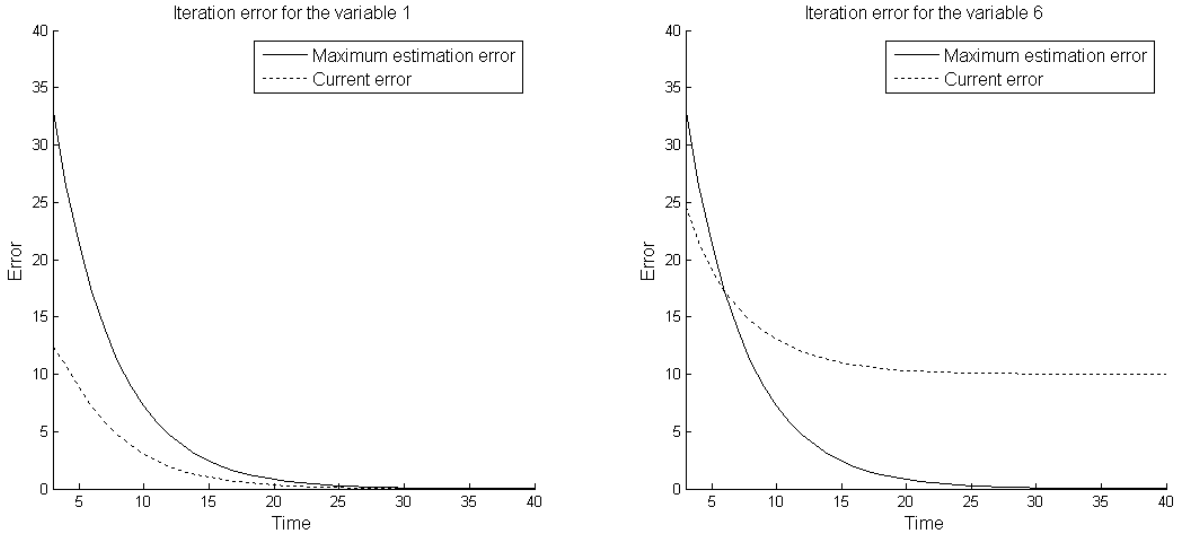

Figure 6.2: Identification of the misbehaving node.

6.2 Linear Deployment

The system consists of n robot that have to reach an uniform distribution over the unit circle. We assume that angles are measured counterclockwise and that the sensors

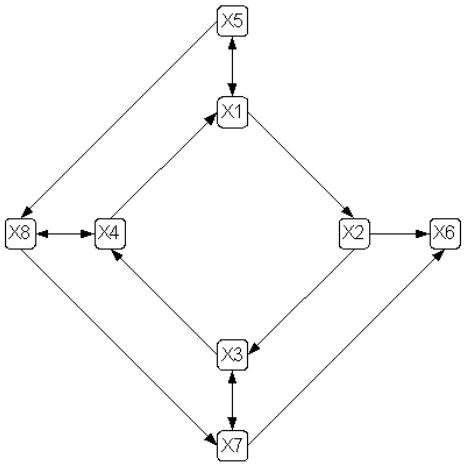

Figure 6.3: Modified consensus network.

are placed in counterclockwise order (we adopt the convention that $\eta_{n+1} = \eta_1$ and that $\eta_0 = \eta_n$). The sensors motion is described by a discrete-time control system:

$$\eta_i(l+1) = \eta_i(l) + u_i \tag{6.8}$$

where u_i is the scalar control magnitude of the i_{th} sensor. We assume that u_i is a function only of the relative angular distances in the counterclockwise direction $d_{counterclock,i} = \eta_{i+1} - \eta i > 0$ and clockwise direction $d_{clock,i} = \eta_i - \eta_{i-1} > 0$. We also assume that each sensor obeys the same motion control law $u : [0, 2\pi][0, 2\pi] \to R$, so that the closed- loop system becomes:

$$\begin{array}{rcl} \eta_i(l+1) & = & \eta_i(l) + u(d_{counterclock,i}(l), d_{clock,i}(l)) \\ d_{counterclock,i}(l) & = & \eta_{i+1}(l) - \eta_i(l) \\ d_{clock,i}(l) & = & \eta_i(l) - \eta_{i-1}(l). \end{array} \tag{6.9}$$

The rule we consider is the "Go Toward the Midpoint of Voronoi Segment":

$$u_{midpointVoronoi}(d_{counterclock}, d_{clock}) = \frac{1}{4}(d_{counterclock} - d_{clock}). \tag{6.10}$$

So the system becomes:

$$\eta(l+1) = A\eta(l) + Ku(l) \tag{6.11}$$

where

$$A = I_n, \tag{6.12}$$

$$K = \frac{1}{4}\begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 & -1 \\ -1 & 1 & \ddots & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & -1 & 1 \end{bmatrix} \tag{6.13}$$

and

$$u = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix}, \quad d_i = \eta_{i+1} - \eta_i. \tag{6.14}$$

The situation is shown in figure 6.4.

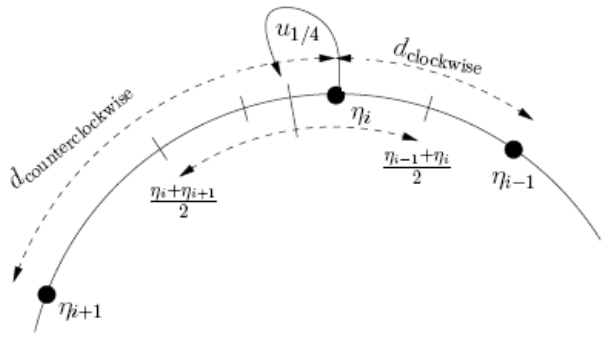

Figure 6.4: Deployment initialization

To perform a convergence analysis, it is convenient to define the relative angular distances $d_i = \eta_{i+1} - \eta_i$, for $i \in 1, ..., n$ (and adopt the usual convention that $d_{n+1} = d_1$ and that $d_0 = d_n$). So long as the counterclockwise order of the sensors is not violated, we have $(d_1, ..., d_n) \in S_{2\pi} = x \in \Re_n \mid x_i \geq 0,\ \Sigma_{i=1}^n x_i = 2\pi$. The change of coordinates from $(\eta_1, ..., \eta_n)$ to $(d_1, ..., d_n)$ and the control law u_k jointly lead to the closed-loop system

$$d_i(k+1) = \frac{1}{4}d_{i+1}(k) + \frac{1}{2}d_i(k) + \frac{1}{4}d_{i-1}(k). \tag{6.15}$$

This is a linear time-invariant dynamical system with state $d = (d_1, ..., d_n)$, transition matrix $A_{\frac{1}{4}}$ given by

$$\begin{bmatrix} \frac{1}{2} & \frac{1}{4} & 0 & \cdots & 0 & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & \ddots & \ddots & 0 \\ 0 & \frac{1}{4} & \frac{1}{2} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{4} & 0 & \cdots & 0 & \frac{1}{4} & \frac{1}{2} \end{bmatrix} \tag{6.16}$$

and governing equation

$$d(k+1) = A_{\frac{1}{4}}d(k), \quad for\ k \in \mathbb{N} \cup \{0\}. \tag{6.17}$$

The properties of this system has been studied in [30], where they prove that, for $k \in]0, 1/2[$, the solutions to the corresponding closed-loop system 6.17 preserve the counter-clockwise order of the sensors and converge exponentially fast to $(2\pi/n, \ldots, 2\pi/n)$.

6.2.1 Numerical Example

We consider 6 agents that are moving on the unit circle in order to reach an uniform distribution over it, and we refer to the algorithm described in the Section 6.2. Suppose the agent number 4 is the intruder, and consider the estimation made by its neighbor number 3. Matrices describing the system are:

$$F = \begin{bmatrix} 1/2 & 1/4 & 0 & 0 & 0 & 1/4 \\ 1/4 & 1/2 & 1/4 & 0 & 0 & 0 \\ 0 & 1/4 & 1/2 & 1/4 & 0 & 0 \\ 0 & 0 & 1/4 & 1/2 & 1/4 & 0 \\ 0 & 0 & 0 & 1/4 & 1/2 & 1/4 \\ 1/4 & 0 & 0 & 0 & 1/4 & 1/2 \end{bmatrix}, \tag{6.18}$$

$$B_4 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad C_3 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}. \tag{6.19}$$

Theorem 4.1.1 is verified, and the related filter is characterized by

$$G = \begin{bmatrix} -1/4 & 0 & 0 \\ -1/2 & -1/4 & 0 \\ -1/4 & -1/2 & -1/4 \\ 0 & -1/4 & -1/2 \\ 0 & 0 & -1/4 \\ 0 & 0 & 0 \end{bmatrix}, \quad K = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{6.20}$$

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{6.21}$$

Fig. 6.5 shows the iteration error, which leads to the identification of the malicious agent, when the unknown input is a sinusoidal signal with amplitude 0.3 and unitary frequency.

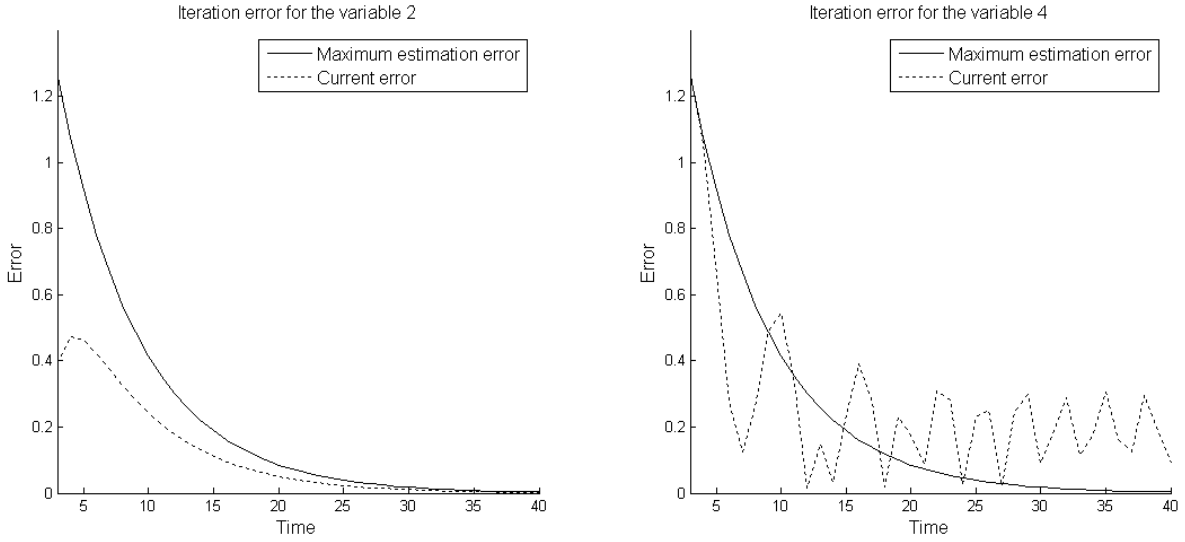

Figure 6.5: Identification of the misbehaving node.

6.3 Maximum Likelihood Estimation

[19] Given a set of n sensor performing measurements of a parameter θ, taking value in a compact set $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$, we aim to compute the average of the taken measurement in the presence of some faulty processors. Let $x_i(l)$ be the measurement of sensor i at time

t, then the desired value is

$$\frac{1}{n}\sum_{i=1}^{n} x_i(0). \tag{6.22}$$

There are at least three methods to choose the weight matrix F in order to compute the average:

1. **Using a doubly stochastic matrix**. With the time-invariant agreement algorithm $x(l+1) = Fx(l)$, we have

$$\lim_{l\to\infty} x_i(l) = \sum_{i=1}^{n} \pi_i x_i(0), \quad \forall i, \tag{6.23}$$

where π is the steady-state probability vector of the Markov chain associated with the stochastic matrix F. It follows that we obtain a solution to the averaging problem if and only if $\pi_i = 1/n$ for every i. Since π is a left eigenvector of F, with eigenvalue equal to 1, this requirement translates to the property $\mathbf{1}^T F = \mathbf{1}^T$, where $\mathbf{1}$ is the vector with all components equal to 1. Equivalently, the matrix F needs to be doubly stochastic.

A possible choice for the weight matrix is given by the Metropolis rule, as follows

$$f_{ij} = \begin{cases} 1/(1+max\{d_i, d_j\}) & \{i,j\} \in E \\ 1 - \sum_{k\in N_i} f_{ik} & i = j \\ 0 & \text{othewise} \end{cases} \tag{6.24}$$

In other words, the weight on each edge is one over one plus the larger degree at its two incident nodes, and the self-weights f_{ii} are chosen so the sum of the weights at each node is 1. The metropolis weights are very simple to compute and are well suited for distributed implementation. In particular, each node only needs to know the degrees of its neighbors to determine the weights on its adjacent edges.

2. **The scaled agreement algorithm**. Suppose that the graph G is fixed a priori and that there is a system designer or other central authority who chooses a stochastic matrix A offline, computes the associated steady-state probability vector (assumed unique and positive), and disseminates the value of $n\pi_i$ to each node i. Suppose next that the nodes execute the agreement algorithm $x = Fx$, using the matrix F, but with the initial value $x_i(0)$ of each node i replaced by

$$\bar{x}_i(0) = \frac{x_i(0)}{n\pi_i}. \tag{6.25}$$

Then, the value $x_i(l)$ of each node i converges to

$$\lim_{l\to\infty} \bar{x}_i(l) = \sum_{i=1}^{n} \pi_i \bar{x}_i(0) = \frac{1}{n}\sum_{i=1}^{n} x_i(0), \tag{6.26}$$

and we therefore have a valid averaging algorithm. This establishes that any (time invariant) agreement algorithm for the consensus problem translates to an algorithm for the averaging problem as well. There are two possible drawbacks of the scheme we have just described:

- If some of the $n\pi_i$ are very small, then some of the initial $\bar{x}_i(0)$ will be very large, which can lead to numerical difficulties.
- The algorithm requires some central coordination, in order to choose F and compute π.

3. **Using two parallel passes of the agreement algorithm**. Given a fixed graph G, let F be the matrix that corresponds to the time-invariant, equal-neighbor, bidirectional model; in particular, if $(i,j) \in E$, then $(j,i) \in E$, and $f_{ij} = 1/d_i$, where d_i is the cardinality of N_i. Assume that the stochastic matrix F is irreducible (the graph G is strongly connected) and aperiodic (because of self loops). Let $D = \sum_{i=1}^{n} d_i$. It is easily verified that the vector π with components $\pi_i = d_i/D$, satisfies $\pi^T = \pi^T F$, and is therefore equal to the vector of steady-state probabilities of the associated Markov chain. The following averaging algorithm employs two parallel runs of the agreement algorithm, with different, but locally determined, initial values:

   - Each node i sets $y_i(0) = 1/d_i$ and $z_i(0) = x_i(0)/d_i$.
   - The nodes run the agreement algorithms $y(l+1) = Fy(l)$ and $z(l+1) = Fz(l)$.
   - Each node sets $x_i(l) = z_i(l)/y_i(l)$.

   We have

$$\lim_{l\to\infty} y_i(l) = \sum_{i=1}^{n} \pi_i y_i(0) = \sum_{i=1}^{n} \frac{d_i}{D}\frac{1}{d_i} = \frac{n}{D}, \tag{6.27}$$

   and

$$\lim_{l\to\infty} z_i(l) = \sum_{i=1}^{n} \pi_i z_i(0) = \sum_{i=1}^{n} \frac{d_i}{D}\frac{x_i(0)}{d_i} = \frac{n}{D}\sum_{i=1}^{n} x_i(0). \tag{6.28}$$

   This implies that

$$\lim_{l\to\infty} x_i(l) = \frac{1}{n}\sum_{i=1}^{n} x_i(0), \tag{6.29}$$

   i.e., we have a valid averaging algorithm. Note that the iteration $y(l+1) = Fy(l)$ need not be repeated if the network remains unchanged and the averaging algorithm is to be executed again with different initial opinions. Finally, if n and D are known by all nodes, the iteration $y(l+1) = Fy(l)$ is unnecessary, and we could just set $y_i(l) = n/E$.

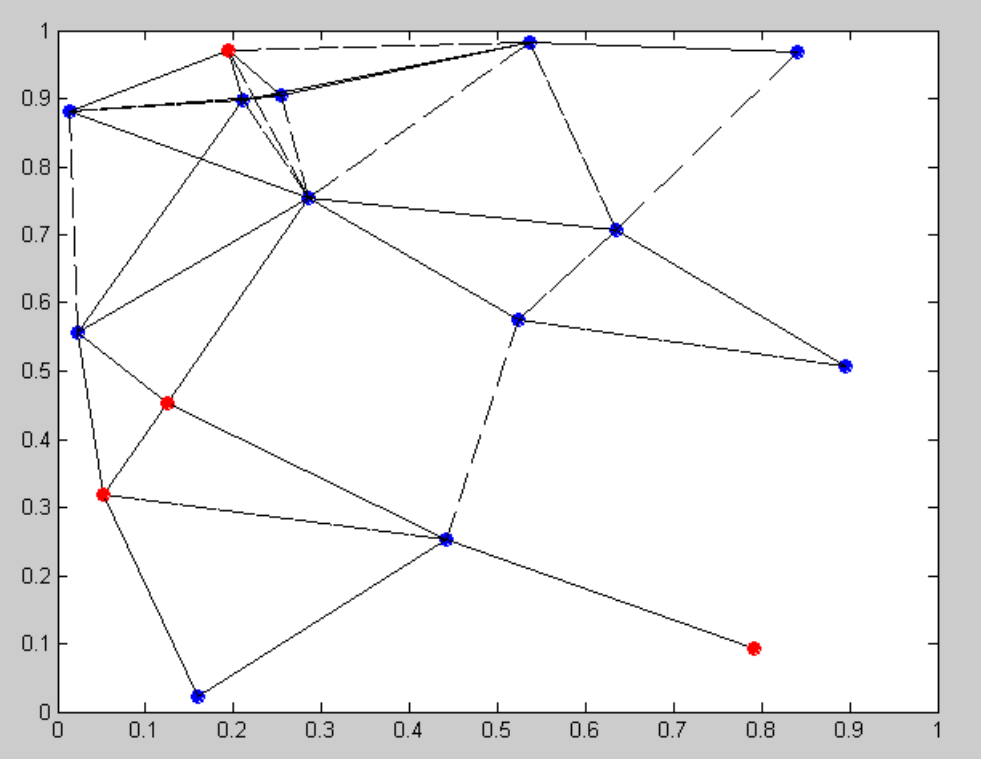

Figure 6.6: A sensor network with 12 nodes (blue) and 4 intruders (red)

The algorithms described above are not robust to the presence of faulty processors, i.e., those algorithms do not converge to the average if some node do not follow the nominal procedure. The faulty processors are modeled as an unknown input in the system, that becomes

$$x(l+1) = F_n(l)x(l) + B\bar{u}(l), \quad B = [e_{k_1}, \ldots, e_{k_p}], \tag{6.30}$$

where B identifies the p faulty processors and $\bar{u}$ denotes the error introduced in the system. If the faulty processors are not excluded from the network, then the average is not computed.

We use Algorithm 1 to solve the faulty processors average computation.

6.3.1 Numerical Example

Consider a sensor network with its communication graph shown in Fig. 6.6. This graph is generated as follows. We first randomly generate $n = 16$ sensor nodes, distributed on the unit square $[0, 1] \times [0, 1]$; then we choose randomly 4 intruders among the nodes of the network. Each node can communicate with all the nodes within a certain distance (0.4 in this simulation), which correspond to the transmission range of the sensor. At the beginning of the simulation there is no communication among the agents, and the

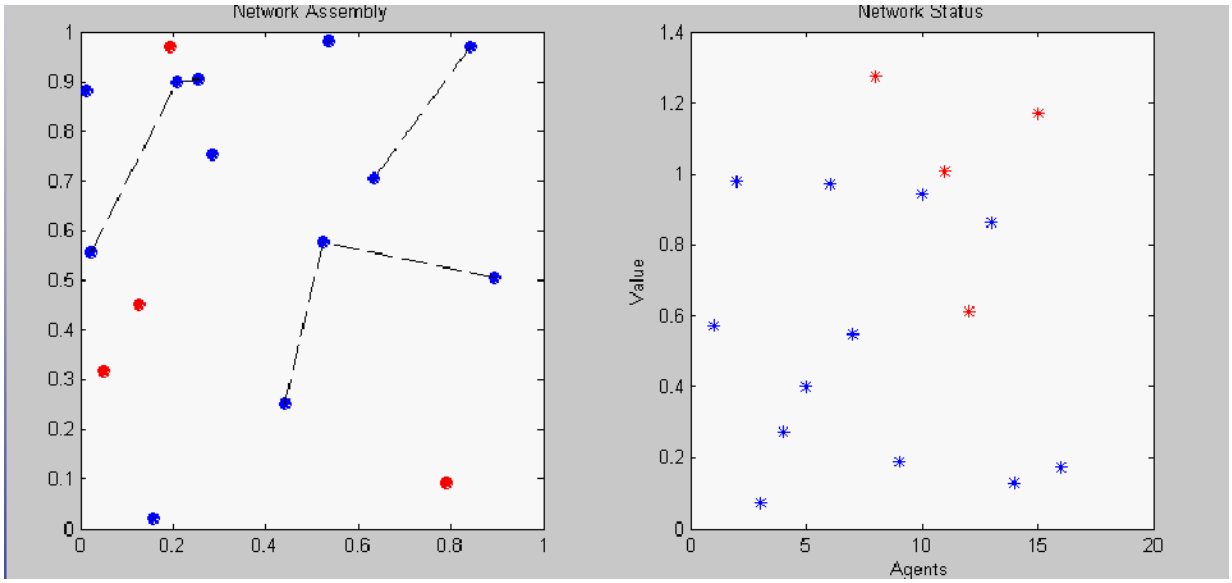

Figure 6.7: Network assembly (1)

procedure aims to assemble a network to evaluate the exact average of the initial values of the agents of the network. Each time an edge is created, nodes updates the weight matrix using the Metropolis rule, so that it is always symmetric and doubly stochastic. The following pictures show the status of the network during the task execution.

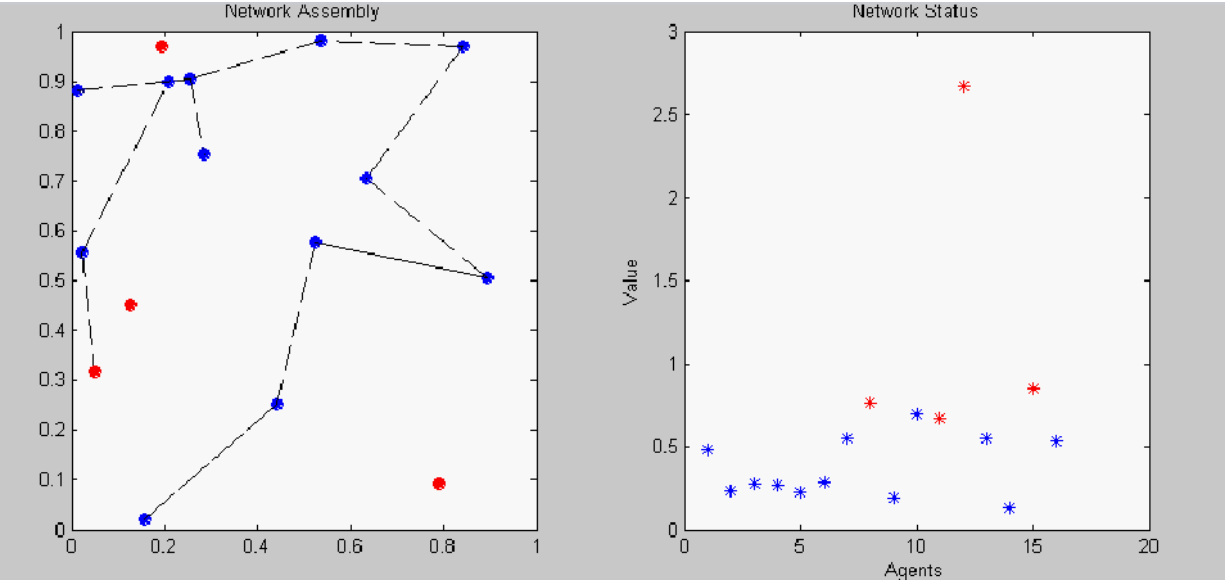

Figure 6.8: Network assembly (2)

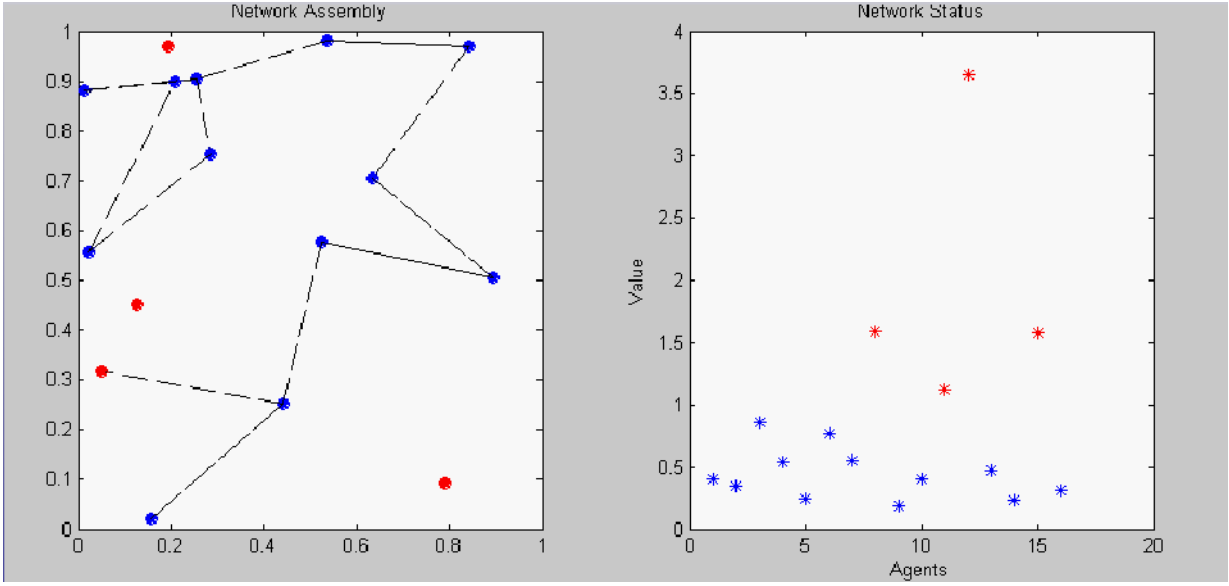

Figure 6.9: Network assembly (3)

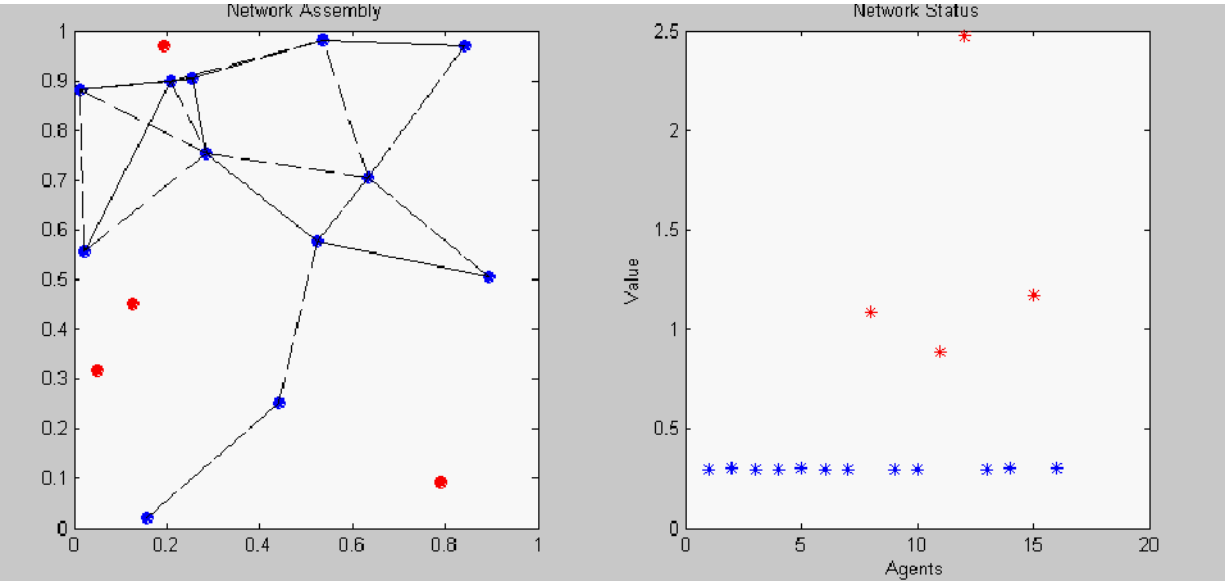

Figure 6.10: Network assembly (4)

Chapter 7

Conclusions

We considered consensus networks in the presence of a misbehaving node, and we proposed a technique based on the theory of Unknown Input Observability to detect, identify, and isolate the misbehavior from the network. We designed an embedded filter, which, only considering the information used by the control protocol, estimates the state of the nodes in the network, allowing the identification of the faulty agent. We analyzed the property of the iteration error of the unknown input filter, and we finally proposed a complete procedure to perform the intrusion detection and isolation task.

We also consider the multiple intruders detection and isolation problem, and we propose an algorithm that aims to assemble a secure network while performing the consensus task among the nodes. With such a procedure, the error an intruder introduces is finite, so that it becomes unimportant when the number of the nodes of the network grows, and the consensus is reached under week assumptions.

The work can be extended in several directions to improve the performance of the security procedure and to include more general situations.

Bibliography

[1] S. Martinez F. Bullo A. Ganguli, S. Susca and J. Cortes. On collective motion in sensor networks: sample problems and distributed algorithms. *IEEE Conf. on Decision and Control and European Control Conference*, 2005.

[2] J.N. Tsitsiklis A. Olshevsky. Convergence speed in distributed consensus and averaging. *In Proceedings of the 45th Conference on Decision and Control, San Diego, California*, 2006.

[3] G. Basile and G. Marro. *Controlled and Conditioned Invariants in Linear System Theory*. Prentice Hall, Englewood Cliffs, NJ, 1991.

[4] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, Belmont, MA, 1997.

[5] S. Buchegger and J.-Y. L. Boudec. Cooperative routing in mobile ad-hoc networks: Current efforts against malice and selfishness. *Proceedings of Mobile Internet Workshop. Informatik*, pages 513–517, September 2002.

[6] S. Buchegger and J.-Y. L. Boudec. Nodes bearing grudges: Towards routing security, fairness, and robustness in mobile ad hoc networks. In *10th Euromicro Workshop on Parallel, Distributed and Network-based Processing*, pages 403–410, Canary Islands, Spain, January 2002.

[7] S. Buchegger and J.-Y. L. Boudec. Performance analysis of the CONFIDANT protocol. In *IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing (MobiHOC)*, pages 226–236, Lausanne, Switzerland, June 2002.

[8] L. Buttyan and J. P. Hubaux. Enforcing service availability in mobile ad-hoc WANs. In *IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing (MobiHOC)*, pages 87–96, Boston, MA, August 2000.

[9] L. Buttyan and J. P. Hubaux. Stimulating cooperation in self-organizing mobile ad hoc networks. *ACM Journal for Mobile Networks (MONET)*, 8(5):579–592, 2003.

[10] G. Ctbenko. Load balancing for distributed memory multiprocessors. *Journal of Parallel Computing*, 7:279–301, 1989.

[11] S. Viscido D. Grunbaum and J. K. Parrish. Extracting interactive control algorithms from group dynamics of schooling fish. *Lecture Notes in Control and Information Sciences*, 2004.

[12] E. Franco, R. Olfati-Saber, T. Parisini, and M. M. Polycarpou. Distributed fault diagnosis using sensor networks and consensus-based filters. In *IEEE Conf. on Decision and Control*, pages 386–391, San Diego, CA, December 2006.

[13] P. M. Frank and X. Ding. Survey of robust residual generation and evaluation methods in observer-based fault detection systems. *Journal of Process Control*, 7(6):403–424, 1997.

[14] V. Gupta, C. Langbort, and R. M. Murray. When are distributed algorithms robust? In *IEEE Conf. on Decision and Control*, San Diego, CA, December 2006. To appear.

[15] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, UK, 1985.

[16] A. Jadbabaie, J. Lin, and A. S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6):988–1001, 2003.

[17] S.-J. Kim L. Xiao, S. Boyd. Distributed average consensus with least-mean-square deviation. *Journal of Parallel and Distributed Computing*, 67(1):33–46, 2007.

[18] S. Lall L. Xiao, S. Boyd. Distributed average consensus with time-varying metropolis weights. *Automatica*, 2006. submitted.

[19] S. Lall L. Xiao, S. Boyd. A space-time diffusion scheme for peer-to-peer least-square estimation. *Proceedings of the Fifth International Symposium on Information Proessing in Sensor Networks (IPSN), Nashville, TN*, April 2006.

[20] H. J. Landau and A. M. Odlyzko. Bounds for eigenvalues of certain stochastic matrices. *Linear Algebra and its Applications*, 38:5–15, 1981.

[21] Y. Liu and Y. R. Yang. Reputation propagation and agreement in mobile ad-hoc networks. In *IEEE Conf. on Wireless Communications and Networking*, pages 1510–1515, New Orleans, LA, March 2003.

[22] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, San Mateo, CA, 1997.

[23] G. Marro. *The Geometric Approach to Fault Detection and Isolation: Motivation, Setting, and Historical Perspective*. DEIS, University of Bologna, Italy, July 2004. CIRA Summer School "Antonio Ruberti".

[24] S. Martínez, F. Bullo, J. Cortés, and E. Frazzoli. On synchronous robotic networks – Part I: Models, tasks and complexity notions. & Part II: Time complexity of rendezvous and deployment algorithms. *IEEE Transactions on Automatic Control*, April 2005. To appear. Short versions were presented at the 2005 CDC/ECC in Seville, Spain.

[25] C. D. Meyer. *Matrix Analysis and Applied Linear Algebra*. SIAM, Philadelphia, PA, 2001.

[26] P. Michiardi and R. Molva. CORE: A collaborative reputation mechanism to enforce node cooperation in mobile ad hoc network. In *Communications and Multimedia Security Conference (CMS)*, Portoroz, Slovenia, September 2002.

[27] R. Olfati-Saber, J. A. Fax, and R. M. Murray. Consensus and cooperation in multi-agent networked systems. *Proceedings of the IEEE*, January 2007. Special Issue on Networked Control Systems. To appear.

[28] R. Olfati-Saber and R. M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49(9):1520–1533, 2004.

[29] B. Prabhakar S. Boyd, A. Ghosh. Gossip algorithms: Design, analysis, and application. *IEEE INFOCOM*, 2005.

[30] F. Bullo S. Martinez. Optimal sensor placement and motion coordination for target tracking. *Automatica*, 42(4):661–668, April 2006.

[31] J. N. Tsitsiklis. *Problems in Decentralized Decision Making and Computation*. PhD thesis, MIT, November 1984. Technical Report LIDS-TH-1424, Laboratory for Information and Decision Systems.

[32] L. Xiao and S. Boyd. Fast linear iterations for distributed averaging. *Systems & Control Letters*, 53:65–78, 2004.

[33] L. Xiao, S. Boyd, and S. Lall. A scheme for asynchronous distributed sensor fusion based on average consensus. In *International Conference on Information Processing in Sensor Networks (IPSN'05)*, pages 63–70, Los Angeles, CA, April 2005.