Università degli Studi di Pisa

FACOLTÀ DI INGEGNERIA
DIPARTIMENTO DI INGEGNERIA DELL'ENERGIA E DEI SISTEMI

PH.D. IN
AUTOMATION, ROBOTICS AND BIOENGENEERING

SSD: ING-INF/04

PH.D. DISSERTATION ON

BEHAVIOR CLASSIFICATION, SECURITY, AND CONSENSUS IN SOCIETIES OF ROBOTS

Candidate Student: **Simone Martini**

Supervisors: **Prof. Antonio Bicchi**

Ing. Adriano Fagiolini

Course Cycle XXIV — Year of Enrolling 2009

To my family and all those who have always believed in me

Abstract

This thesis addresses some fundamental issues toward the realization of “societies” of robots. This objective requires dealing with large numbers of heterogenous autonomous systems, differing in their bodies, sensing and intelligence, that are made to coexist, communicate, learn and classify, and compete fairly, while achieving their individual goals.

First, as in human or animal societies, robots must be able to perform cooperative *behaviors* that involve coordination of their actions, based on their own goals, proprioceptive sensing, and information they can receive from other neighboring robots. An effective way to successfully achieve cooperation is obtained by requiring that robots share a set of decentralized motion “rules” involving only locally available data. A first contribution of the thesis consists in showing how these behaviors can be nicely described by a suitable hybrid formalism, including the heterogenous dynamics of every robots and the above mentioned rules that are based on events.

A second contribution deals with the problem of classifying a set of robotic agents, based on their dynamics or the interaction protocols they obeys, as belonging to different “species". Various procedures are proposed allowing the construction of a distributed classification system, based on a decentralized identification mechanism, by which every agent classifies its neighbors using only locally available information. By using this mechanism, members of the

society can reach a consensus on the environment and on the integrity of the other neighboring robots, so as to improve the overall security of the society. This objective involves the study of convergence of information that is not represented by real numbers, as often in the literature, rather by sets. The dynamics of the evolution of information across a number of robots is described by set–valued iterative maps. While the study of convergence of set–valued iterative maps is highly complex in general, this thesis focuses on Boolean maps, which are comprised of arbitrary combinations of unions, intersections, and complements of sets.

Through the development of an industrial robotic society, it is finally shown how the proposed technique applies to a real and commercially relevant case–study. This society sets the basis for a full–fledged factory of the future, where the different and heterogeneous agents operate and interact using a blend of autonomous skills, social rules, and central coordination.

Acknowledgements

First and foremost I deeply wish to thank Prof. Antonio Bicchi for his guidance and education. I truly believe that his encouragement and precious suggestions have helped me a lot in reaching the results of this thesis.

I also wish to express my sincere gratitude to Adriano Fagiolini. His knowledge and understanding, his encouragement and guidance have provided an essential improvement for the present thesis. Last but not least I wish to thank Davide Di Baccio for his support and assistance during the last period of my Ph.D. I am glad to thank Lucia Pallottino, Paolo Salaris, Luca Greco, and Felipe A. W. Belo for their willingness to talk with me on various concerns regarding my research during these last years.

I owe my most sincere gratitude to Prof. Magnus Egerstedt who gave me the opportunity to work with him at Georgia Tech, in the School of Electrical and Computer Engineering, during my abroad experience.

I wish to thank my best friends, colleagues, and the administrative staff of the Interdepartmental Research Center "E. Piaggio" who made the working environment a stimulating, efficient and fun place in which to learn, work, and grow.

Lastly, and most importantly, I wish to thank my father and my mother, who taught me to give value to learning, and gave me their unconditional support as I traveled the road of education.

Contents

Acronyms

ANF	Algebraic Normal Form
BA	Boolean Algebra
BDS	Boolean Dynamic System
CHAT	Control of Heterogeneous Automation Systems
DFT	Discrete Fourier Transform
EU	European Union
FT	Fourier Transform
FFT	Fast Fourier Transform
HMM	Hidden Markov Model
LGV	Laser Guided Vehicles
LH	Left Hand Traffic Rules
NTP	Network Time Protocol
RH	Right Hand Traffic Rules

SBM	Set–Boolean Map
SRNP	Self Routing Network Protocol
SR^{2}NP	Self–Routing Robust Network Protocol
SVBDS	Set–Valued Boolean Dynamic System
VNN	Von-Neumann Neighborhood
WSN	Wireless Sensor Networks

Chapter 1

Toward a Society of Robots

Since its birth 50 years ago, Robotics has witnessed an increasing need for integration of robots in human environments which has given a boost to the research for new control techniques that involve forms of cooperation among robots and even with humans. However, the trend shown by available market data (Fig. 1.1) seems to anticipate an even more surprising future, when personal robots will be so many and so ubiquitous that the core scientific and technical issues might become those of Robot–Robot Interaction. Applications where two or more robots will coexist and collaborate to perform some set of tasks or to satisfy some set of goals are becoming a reality. These systems may be comprised of homogeneous or heterogeneous robotic agents, i.e. differing in size, perception, computation, and actuation capacities. An agent in the system is considered a locus of problem-solving activity, which operates asynchronously with respect to other agents, and it has a certain level of autonomy. Agent autonomy relates to an agent's ability to make its own decisions about what activities to do, when to do them, what type of information should be communicated and to whom, and how to assimilate the information received. Autonomy can be limited by policies built into the agent by its designer, or as a result of an agent organization dynamically coming to an agreement that specific agents should take on certain roles or adopt certain policies for some specified period. Closely associated with agent autonomy is agent adaptabil-

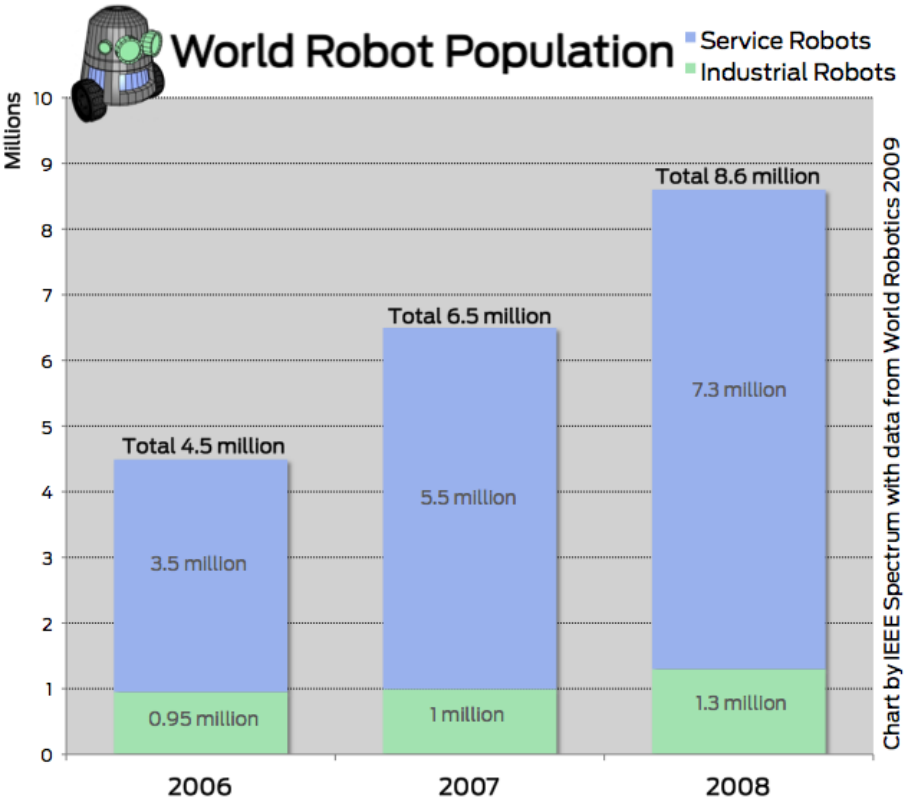


Fig. 1.1 The world robot population. Source: IEEE Spectrum with data from World Robotics 2009.

ity - the more autonomy an agent possesses the more adaptable it is to the emerging problem solving and network context. The degree of autonomy and the range of adaptability are usually associated with the level of intelligence / sophistication that an agent possesses.

Due to this flexibility, multi-agent systems over the last few years have come to be perceived as crucial technology not only for effectively exploiting the increasing availability of diverse, heterogeneous, and distributed on-line information sources, but also as a framework for building large, complex, and

robust distributed information processing systems which exploit the efficiencies of organized behavior. Moreover, multi-agent systems provide a powerful model for computing in the twenty-first century, in which networks of interacting, real-time, intelligent agents seamlessly integrate the work of people and machines, and dynamically adapt their problem solving to effectively deal with changing usage patterns, resource configurations and available sources of expertise and information. By structuring such applications as a multi-agent system rather than as a single agent, the system will have the following advantages: speed-up due to concurrent processing; less communication bandwidth requirements because processing is located nearer the source of information; more reliability because of the lack of a single point of failure; improved responsiveness due to processing, sensing and effecting being co-located; and finally, easier system development due to modularity coming from the decomposition into semi-autonomous agents. In this sense, such systems are ubiquitous in diverse areas of science and engineering such as physiological systems and gene networks [8], large scale energy systems, and multiple space, air, and land vehicles [9, 10, 11].

1.1 From Natural Societies to Robotic Societies

The functions and the structures of these multi-robot systems may indeed be different in terms of organization as it happens in a *society*. Human and animal societies are based on the establishment of "rules of interaction" among the individuals of their societies themselves. These rules may consist of either collaborative or non–collaborative *behaviors* that each (possibly different) individual may display towards the others [12]. The adoption of similar notions of "sociality" and heterogeneity in Robotics is advantageous in many tasks, where a coordination among agents with analogous or complementary capabilities is necessary to achieve a shared goal. More specifically, "robotic societies" are

distributed multi–agent systems where each agent is assigned with a possibly different local goal, but needs to coordinate its actions with other neighboring agents. The flexibility and robustness of such distributed systems, and indeed their ability to solve complex problems, have motivated many works that have been presented in literature (see e.g., [13, 14, 15, 16, 17, 18]). Although in most cases agents are modeled as identical *copies* of the same prototype, this assumption is often restrictive as the different agents that forms a society may be implemented by different makers, and with different technologies etc."Sociality" and heterogeneity in these artificial systems are advantageous when a coordination among agents with analogous or complementary capabilities is necessary when they are supposed to coexist and behave so that the accomplishment of their mission does not jeopardize the chances of others, see e.g. automatic management systems in highways for the distance keeping in queues which are already commercially available. One possibility is that robots could be organized in teams, flocks or swarms, to more effectively and robustly pursue a goal which is shared by all members. In this case, the paradigm of "emergent behaviors" is often used to describe the coordination of large numbers of robots with limited individual capabilities which achieve complex tasks (see e. g. [16, 19, 18, 20]). When more complex robots put their specific capabilities at the disposal of a common goal, the paradigm of "intentional cooperation" is evoked [21].

Looking a bit further ahead, one can easily imagine personal robots going to shop for the family (Fig. 1.2). The user might take the robot to the local mall, where the robot could obtain information on goods and their locations, fill the cart (checking the list prepared by smart appliances in the house), wait in the queues etc. - while the owner makes time for more amusing activities. It is interesting to speculate on what it is that makes this scenario impossible today, and what breakthroughs could enable it tomorrow. The major obstacle in such a scenario would be that tens or hundreds of robots going about their individual programmed missions would compete for resources (e.g. room, power, goods),

Fig. 1.2 A futuristic mall scenario.

possibly creating conflicts and ending up in traffic deadlocks, collisions, or even safety hazards. To negotiate the potential conflicts, communication and interaction among robots will have to be codified in a set of rules to which different robot producers and infrastructure authority should adhere.

1.2 What is Missing

Comparing the state of the art with the analysis of the mall scenario, several challenging problems are still outstanding. First, a *general model* to uniformly describe the behavior of large, open groups of cooperative robots is needed. Through this model it will be possible to specify motion cooperation protocols guaranteeing e.g. safety (collision avoidance with robots and humans) and per-

formance (ensuring that each robot eventually gets a chance to accomplish its mission). In the society new robots may get in, or leave, at any time, irrespective of their model, type, size or weight of the robots – provided only that they abide to the society's rules. A very effective way of achieving features such as "scalability", "heterogeneity", and "reconfigurability" is *decentralization*, i.e. decisions should be made by each agent autonomously and should be based on information limited to a local neighborhood of each robot, reducing the role of a central authority to the minimum necessary.

A system that relies on social behaviors to mitigate the excess of individualism is intrinsically very sensitive to the possibility that misbehaviors occur, due to either faults in some robots or malicious programming of agents. Thus, security requirements are crucial for a society of robots, which imply the capability to detect, isolate, and neutralize the threat posed by misbehaving robots. In a society of autonomous robots, *behavior classification and detection* must also rely on information available locally and on limited knowledge of a model for the behavior of other robots. A common problem with over–cautious security policies is that they can make the system too stiff and ineffective. In a heterogeneous robot society, a robot should not deem another robot to be a malevolent intruder just because it behaves differently, as far as that behavior does not pose a threat. Hence, a problem of detecting which type of behavior other robots in the neighborhood are following, or which "species" they belong to, is also in order.

Finally, at the base of interaction, there is the ability for the agents to *consent* on a global view of the system's state by exchanging locally estimated views, i.e. to reach an agreement on complete system awareness.

1.3 Contributions

This thesis discusses the above challenges focusing on the problem of how very large numbers of heterogenous robots, differing in their bodies, sensing and computational capabilities, may be made to coexist, communicate, and compete fairly towards achieving their individual goals, i.e. to build a "society of robots". This is achieved through various contribution that are detailed below.

Distributed Classification in Linear Consensus Networks

In most of the literature, it is common to represent large systems of many autonomous but networked units, capable of acting in and on the environment by using tools of graph theory, where agents are identified with nodes and encoding the existence of an interaction between nodes with edges [22, 23, 24, 25, 26, 16, 27, 28]. In this thesis, systems of heterogeneous autonomous agents involved in establishing a consensus on an unknown variable through communication over a network are firstly considered. The work focus on the problem for an agent to identity the type of neighboring agents by using only locally available information. To this aim, for linear consensus networks and a model of heterogeneity described by individual weights, it will be proposed a method that allows an observer node to accurately identify the weight of all its neighbors from the analysis of the received signals only, without any knowledge of the network structure (Chapter 2). The method relies on the generation by the observer node of an excitation signal, and on a frequency-domain analysis of the network response. Any node can be an observer, and simultaneous observation by an arbitrary number of nodes is possible, provided only that excitation signals are orthogonal.

Hybrid Automata as a Model for Social Behaviors

Whereas this model of heterogeneity based on scalar weights applied to the standard agreement algorithm represents a viable solution for systems where the difference between agents is *simple* (e.g. different performances and rates in mobility or communication, different clock rates, etc.), this thesis aims at defining a formalism to represent complex heterogenous systems such as the human and animal societies. A "Behavior–based" society of robots can be built by giving a set of rules that each robot has to follow that can be represented through the *hybrid automata* formalism and verification techniques that can be effectively used to that purpose (Chapter 3). The protocol allows the description of many societies of interacting robots at a suitable abstraction level, including artificial systems (e.g. society of vehicles traveling along a highway and following different sets of driving rules) as well as approximations of systems inspired from Biology (e.g. society of Daceton Armigerum tree–dwelling workers ants).

Global Awareness and Consensus on Set–Valued Information

As it will be introduced in Chapter 2, many of the problems attacked so far in the literature on distributed robotics require that agents consents on information represented by data consisting of scalars (such as a temperature or the concentration of a chemical) or vectors (e.g., positions or velocities) [16, 17, 29]. Available approaches differ mainly in the type of rule each agent uses to combine its own information with the one received from its neighbors in the communication graph. However, the complexity of the problems emerging in society of robots entails defining consensus algorithms on different representations of the state of information. This work focuses on the convergence of information in distributed systems which is not represented by real numbers, as often in the literature, rather by sets. The dynamics of the evolution of informa-

tion across the network is accordingly described by set–valued iterative maps. While the study of convergence of set–valued iterative maps is highly complex in general, this thesis focuses on Boolean maps, which are comprised of arbitrary combinations of unions, intersections, and complements of sets. For these important class of systems this work provides tools to study both global and local convergence (Chapter 4) .

Distributed Synthesis of Robust Logical Consensus Systems

At a suitable abstraction level, every robotic society requires that agents consent on a centralized logical decision e.g. one can imagine that the behaviors of agents described according to the protocol defined in Chapter 3 depends on a set of input events on which agents have partial visibility and therefore need to be estimated by the agents themselves. In this view, Chapter 5 considers the synthesis of consensus systems where agents' interaction can be described by rules that are *decentralized* and *event–based*, i.e. they specify what cooperation action every agent can perform based only on locally measured events. The approach proposed is based on the construction of an iterative map, whose computation is fully distributed and guaranteed to "converge" to a unique decision under suitable conditions on the input visibility and graph connectivity. Moreover, in order to cope with possible faults of some agents, eventually leading the system to incorrect or disconnected decisions, the above solution is extended. The proposed solution is based on a fully distributed synthesis procedure which generates a nonlinear consensus map that is able to tolerate misbehaving agents that may send incorrect information, due to spontaneous failure or even tampering. The procedure is formalized as a distributed protocol and is based on the well–known result of Lamport [30] ensuring that redundant minimum–length paths from a generic input to every agent of the network can be found if the number of faults is bounded.

Behavior Classification, Detection, and Security

As introduced before, at the base of interaction between agents in a society it stands the ability of each individual (agent) to distinguish or *classify* the other neighboring members (agents) of the society that it gets in contact with, as belonging to the same group or species. In particular some robots may experience sensor or actuator failure, and thus be unable to follow the protocol rules. Moreover, since robot communication is based on a wireless network, an adversary could easily drop on communication as well as inject/modify packets. In such cases, safety requires that the society must be able to recognize such failures or intrusions in order to activate countermeasures to preserve the overall system and its individuals. All these cases can be recast as behavior classification problems, i.e. the problem to classify heterogeneous agents that "behave" in a different way, due to their own physical dynamics or to the rules of interaction they are obeying, as belonging to a different species (Chapter 6).

Firstly, this work presents a technique allowing a decentralized classification system to be built in a systematic way, once the models describing the behavior of the different species are given. It is based on a decentralized identification mechanism, by which every agent classifies its neighbors using only locally available information (Section 6.1.2). However, an agent with partial visibility on the neighborhood of another agent may be unable to determine the correctness of the estimate on the neighborhood. To overcome this limitation, this work present a consensus algorithm allowing agents to reach an agreement on the species to which other neighboring agents belong to by combining the information extracted from local classifiers (Section 6.1.3).

The limitation of the above solution is related to the necessity of the a priori knowledge of the models describing each possible behavior and interaction. In this sense, every agent should be endowed with the ability to autonomously and iteratively learn the different possible behaviors and interactions by locally observing its neighbors. The problem of learning behaviors through demon-

strated sequences of actions from a human performer, a.k.a. apprenticeship learning [31, 32, 33], is largely studied in the literature.

This thesis provide a solution to the classification problem without any priori knowledge, that is partially built upon the above mentioned results on model learning, and that consists of a procedure that each agent can run to iteratively grow a knowledge base of all possible species that are present in the "society". More precisely, each "estimated" species is represented by a Hidden Markov Model (HMM), whose structure and parameters are iteratively learnt based on the actually measured trajectory of the target agent (Section 6.2).

Even though the HMM represent a valuable tool for classifying agents belonging to different species based on observed behavior, due to the probabilistic nature of the HMM, the reconstruction of the hybrid model of the agents' behavior according to the protocol defined in Chapter 3, i.e. to relate the behavior of an agent to its neighborhood starting from the HMM parameters may be very complicated. For this reason, aiming at reconstructing the hybrid model from the observed behavior, the problem need to be tackled with a different approach. By assuming that the continuos states of the hybrid model are given, a technique allowing agents to investigate and to reconstruct the rules of interaction describing the behavior of an agent against its neighbors is proposed (Section 6.3).

The approach is based on a canonical form of representation for Boolean functions [34], the so–called *Algebraic Normal Form* (ANF), and consists on a procedure by which an unknown logical map can be learnt as its truth table is dynamically observed. The estimated map represents a lower approximation of the real one, which coincides with it as soon as the truth table is entirely observed.

An Industrial Society for Fast and Reliable Factory Automation

One of the mean asset of this work is to prove that the developed tools can be applied to very concrete industrial scenarios which provide detailed requirements for methods and technologies in a very representative application domain for industrial automation. By extending the above concept to the factory structured environment, it is possible to set the basis for a full-fledged factory of the future, where the different and heterogeneous agents operate and interact using a blend of autonomous skills, social rules and central coordination, i.e. it is possible to integrate the society of robots in the industrial field.

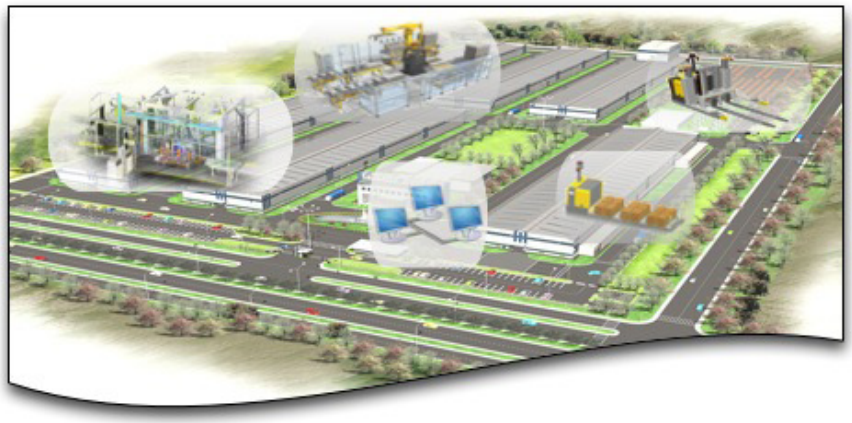


Fig. 1.3 An autonomous production plant where industrial robotic systems (such as Laser Guided Vehicles) are largely used and highly integrated with the information systems of the enterprise and of the suppliers.

Multi–vehicle robotic systems are largely used in industrial transportation and logistics systems as they provide competitive advantage versus the single-agent solutions in terms of task speedup, robustness and scalability. For instance, a typical function of a multi Laser Guided Vehicle (LGV) system con-

sists in transporting raw or semi–finished material from a factory's warehouse to its production lines. However, their adoption raises management and coordination problems, such as collision avoidance, conflict resolution requiring fast and reliable negotiation of shared resources.

Conflict resolution in the use of these resources is critical for safety and robustness of the operations of material handling. Avoidance/resolution of stall situations as well as fluent navigation of the robotic agents must be guaranteed for system efficiency. Stall situations occur when agents are unable to move from the particular configuration (i.e. deadlock) or constrained to move along a finite number of paths without reaching the final destination (i.e. livelock) [14, 35, 36, 37]. Many works focus also on the important aspects of collision avoidance and deadlock avoidance (see e.g. [38, 39, 40, 41]).

For the final demonstration of the EU project CHAT [42], the tools and theory described in this thesis have been used to realize a real industrial robotic society allowing the improvement of the factory's decision skills by enhancing its awareness on the current situation of the factory floor (Chapter 7). This has been done through an effective distributed sensing scheme, where every "member" of the society (the autonomous mobile vehicles that are moving in the factory's logistic area) acquires local information of the system state for its own use to share it with the neighbors and with the central supervisor. Members of the factory are able to reconstruct and estimate the status of every other member and even that of the global system, through cooperative interaction schemes and communication protocols that are scalable, reliable, and fast. In this respect, detection and classification of faults in the factory – including deviance from the correct assigned rules of some of its "peers" that can spontaneously occur or that are maliciously introduced – represent built–in functionalities of the factory (made possible by cooperation of the robots) enabling a quick and effective reaction and reconfiguration of all the factory components involved.

Chapter 2

Distributed Classification in Linear Consensus Networks

In most of the literature, it is common to refer large systems of many autonomous but networked units, capable of acting in and on the environment as *Multi–agent systems*. By abstracting away the complex interaction geometries associated with the sensing and communication footprints of the individual agents, it is possible to use tools of graph theory to model these systems where agents are identified with nodes and the existence of an interaction between nodes is encoded by edges. In this context, the problem of controlling such kind of systems has attracted substantial attention, and has lead to the development of several distributed algorithms to solve tasks as varied as parameter estimation, average consensus, rendez–vous, sensor coverage and simultaneous localization and mapping [43, 44]. Therein, it is common to model agents as identical copies of the same prototype. However, such assumption is often restrictive, because these system may be comprised of homogeneous or heterogeneous agents which may be implemented by different technologies, makers, etc.

Referring to the structure of the linear consensus network it is possible to introduce a model of heterogeneity described by individual weights applied to the standard agreement algorithm where different weights may represent e.g. different velocities of the robot agents in a rendez–vous task, or different response rates of agents in their update functions, etc. For such networks, a

distributed strategy based on local interactions to make decisions concerning the class of the other agents in the network is proposed. The method allows an observer node to accurately identify the weight of all its neighbors from the analysis of the received signals only, without any knowledge of the network structure. The basic idea of the method is that the observer node probes its neighborhood, for a limited time interval, with an excitation input [45, A7]. The observer then applies standard frequency-domain analysis tools to the signals received from its neighbors, from which the sought information is elicited. Any node can be an observer, and simultaneous observation by an arbitrary number of nodes is possible, provided only that excitation signals are orthogonal.

2.1 Linear Consensus Networks

As standard in multi-agent systems, the nodes of a graph represent the agents, and the arcs in the graph represent the communication links. A two-way communication channels is assumed.

Let the undirected graph G be given by the pair $(V, \mathcal{E})$, where $V = \{1, \ldots, n\}$ is a set of n vertices, and $\mathcal{E}$ is a set of edges. G can be associated with an *adjacency matrix* $\mathcal{A} \in \mathbb{R}^{n \times n}$ whose entries satisfy

$$[\mathcal{A}]_{ij} = \begin{cases} 1 \text{ if } (i,j) \in \mathcal{E} \\ 0 \quad \text{otherwise.} \end{cases}$$

The standard linear consensus algorithm is described in continuous–time by the update law

$$\dot{x}_i(t) = \sum_{j \in N_i} (x_j(t) - x_i(t)), \tag{2.1}$$

where N_i is the set of all agents in the neighborhood of agent i and it is defined as $N_i = \{k : (i,k) \in \mathcal{E}\}$.

Since it is assumed that $x_i \in \mathbb{R}$, Eq. (2.1) is rewritten as

$$\dot{x}(t) = (\mathcal{A} - \mathcal{D})x(t) = -\mathcal{L}x(t) \tag{2.2}$$

where $x(t) = [x_1\, x_2\, \cdots\, x_n]^T$ is the vector with the states of all nodes at time t, and $\mathcal{D}$ is G's the *degree* matrix and $\mathcal{L}$ is the *Graph Laplacian*.

Under some connectivity conditions, the consensus algorithm is guaranteed to converge [22, 46, 47], i.e.

$$\lim_{t \to +\infty} x_i(t) = g \;\; i \in \{1, \ldots, n\}$$

where g is the time-invariant centroid depending on $\mathcal{L}$, and on the initial conditions $x_0 = x(0)$ and it is given by

$$g = \frac{1}{n}\mathbf{1}^T x.$$

To introduce heterogeneity in a consensus system, it is possible to consider a simple but interesting model of diversity between agents by assuming that agents differ by a scalar gain or weight parameter, which may model e.g. different performances and rates in mobility or communication, different computation capabilities, different clock rates, etc. Denoting γ_i as the constant weight associated to the $i - th$ agent

Eq. (2.2) is replaced by the following interaction rule

$$\dot{x_i} = \gamma_i \sum_{j \in N_i} (x_j - x_i), \tag{2.3}$$

or equivalently

$$\dot{x} = -\Gamma\, L\, x, \tag{2.4}$$

where $\Gamma = diag(\gamma_1, \ldots, \gamma_n)$is the matrix of the class of the agents in network. It is proved in [23] that the system (2.4) converges to the weighted centroid

$$c_\Gamma = \frac{1}{tr(\Gamma^{-1})} 1^T \Gamma^{-1} x$$

which is also invariant in time.

2.2 Beyond the standard Linear Consensus

This works considers the problem of the classification of different kind of agents in networked systems, i.e the estimation of the diagonal entries of Γ in the system defined by (2.4), given the measurement signal $\theta = [\theta_1, \ldots, \theta_n]$ with not known-static (or piecewise static), connected network topology.

As such, the classification problem involves solving a partial system identification problem where the overall interaction topology and the node weights are not known.

Problem 2.1. Let G be a multi agent network with dynamics described by (2.4) and let A_i and A_j be two agents in the network such that $i, j \in \{1, 2, \cdots, n\}, i \neq j, i \in N_i$. Given the knowledge of the vector $\theta_j = [x_j, \dot{x}_j, \ddot{x}_j]$, design a a fully decentralized procedure allowing A_i to correctly estimate the value of γ_j.

Since the the overall interaction topology is not known, let z_j be the term in Eq. (2.4) representing the consensus algorithm involving neighbors of the j-th agent excluding agent i, i.e.

$$z_j = \sum_{k \in N_j - \{i\}} (x_j - x_k). \tag{2.5}$$

Equation (2.4) can be rewritten as

$$\dot{x}_j = -\gamma_j(x_j - x_i) - \gamma_j z_j. \tag{2.6}$$

Remark 2.1. The estimation of the value of γ_j from A_i as in Problem 2.1 based on the knowledge of $\alpha_j = [x_j, \dot{x}_j]$ is not possible because the topology of the network involving neighbors of agent j is unknown to A_i. It results that

$$\forall\ \gamma_j\ \exists\ z_j\ s.t.\ \ \gamma_j(x_i - x_j - z_j) = \dot{x}_j.$$

Therefore, as a solution of the Problem 2.1, in this work it is proposed to modify interactions between nodes by envisioning a scenario where agents are executing a different control strategy that is different from the pure consensus dynamics in order to excite the system (see e.g. [45, A7]). In this sense, the main contribution of this work is to propose a possible characterization of what signal the agents should exert in order to achieve the correct estimation of the γ values in 2.4, i.e. to solve the Problem 2.1.

To allow agents to exert a control action different from the consensus-based algorithm, assume that the network behavior can be described by

$$\dot{x} = -\Gamma\,\mathcal{L}x + Bu. \tag{2.7}$$

Here B is an $n \times n$ matrix (typically the identity matrix), u is a vector of inputs whose i-th component u_i is the scalar input exerted by agent i. In the following it is proposed a possible choice for the input in Eq. (2.7) allowing agent A_i to solve Problem 2.1 for its neighboring agents in the frequency domain.

The goal of this frequency domain identification is to apply inputs in the form

$$u_i(t) = W_i \sin(\omega_i\, t) \qquad t \in [0,\ T_f]; \tag{2.8}$$

for some W_i, ω_i, and T_f, so as to determine the estimation of $\hat{\gamma}$ values based on the known inputs of Eq. (2.8), and the sampled measured outputs $\ddot{x}(k\,T_s)$ so that

$$\hat{\gamma}_j \to \gamma_j \quad j \in N_i.$$

Following this approach, the agent classification problem can be now successfully mapped into a problem of signal processing that each agent can efficiently and independently solve by computing the Discrete Fourier Transform (DFT). In order to better understand the proposed solution, in the next section some theory about the *Fourier series representation* is recalled.

2.3 Fourier Series representation

It is well known that by using the Fourier series representation, a complex-valued function can be transformed from the time domain to the frequency domain representation of the original function. In particular through the Fourier Transform (FT) a periodic signal can be represented as a sum of complex exponential. Therefore, the result is a decomposition of the original function into oscillatory functions which describe the frequencies that are present in the analyzed function.

Consider a discrete time periodic signal $g_{T_0}(k)$ of period T_0, that is $g_{T_0}(k + T_0) = g_{T_0}(k)$. Hence $g_{T_0}(k)$ can be obtained as to samples from a continuous time signal $\tilde{g}_{T_0}(t)$ sampled at instants $t = kT_0$, i.e. $g_{T_0}(k) = \tilde{g}_{T_0}(kT_0)$ Then $g_{T_0}(k)$ can be represented in terms of the *Discrete Fourier Transform* (DFT), i.e.

$$g_{T_0}(k) = \sum_{k=0}^{T_0-1} \bar{G}_h e^{j\frac{2\pi hk}{T0}}.$$

The sequence $\mathcal{F}[g_{T_0}(t)] = \bar{G}_k$ of the *complex Fourier coefficient* is defined by

$$\bar{G}_h = \frac{1}{T_0} \sum_{k=0}^{T_0-1} g_{T_0}(k) e^{-j\frac{2\pi hk}{T_0}}. \tag{2.9}$$

Although, DFT is a very useful tool for the frequency domain function analysis, it is clear that computing it directly from the definition (2.9) is often too slow to be practical. A key enabling factor for these applications has been the introduction of Fast Fourier transform (FFT) algorithm by which the DFT can be computed efficiently and quickly. There are many distinct FFT algorithms involving a wide range of mathematics, from simple complex-number arithmetic to group theory and number theory. Therefore, the difference in speed can be substantial, especially for long data sets where N, number of samples, may be in the thousands or millions, and make the DFT have a great importance to a wide variety of applications, from digital signal processing and solving partial differential equations to algorithms for quick multiplication of large integers.

2.4 Frequency Domain Agents Classification

In this section a method to solve Problem 2.1 is proposed. In this perspective the following theorem can be stated.

Theorem 2.1. *Let $u_i(t) = W_i \sin(\omega_i t)$ be the i-th scalar input exerted by agent i in Eq.* (2.7)*, and let A_j be an agent s.t. $j \in N_i$. Let T_s be the sampling time, and suppose that A_i is able to measure $\ddot{x}_j(kT_s),\ kT_s \in [0, T_f]$. The estimation $\hat{\gamma}_j$ executed by A_i defined as*

$$\hat{\gamma}_j = 2 \max \left(\mid \mathcal{F}[\ddot{x}_j(kT_s)] \mid_{f \approx \frac{\omega_i}{2\pi}} \right) \tag{2.10}$$

is such that

$$|\hat{\gamma}_j - \gamma_j| < \epsilon$$

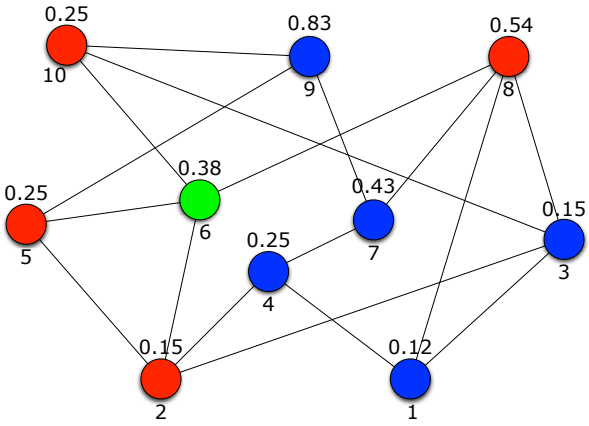


Fig. 2.1 A network of 10 agents where the Agent 6 (green) is investigating about the class (γ values) of the Agents 2,5,8,10 (red).

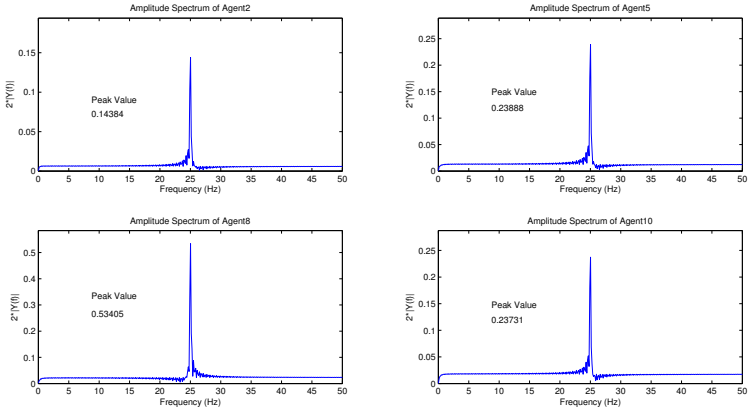


Fig. 2.2 The result of the frequency domain agent classification. The Amplitude spectrums of the DFT report peak values at the frequency of 25 Hz close to the real γ values.

where $\epsilon > 0$ *depends on* T_s *and* T_f.

Proof. As previously discussed, A_i is able to measure $[\ddot{x}_j, \dot{x}_j]$. According to Eq. (2.7), $\dot{x}_i$ can be written as

$$\dot{x}_i(t) = -\mathcal{L}_i x(t) + W_i \sin(\omega_i t) \tag{2.11}$$

where $\mathcal{L}_i$ is the i-th row of the matrix $\Gamma\mathcal{L}$. The value of $\ddot{x}_j$ results

$$\ddot{x}_j = \gamma_j(\dot{x}_i - \dot{x}_j - \dot{z}_j). \tag{2.12}$$

By substituting (2.11) in (2.12) it results

$$\ddot{x}_j = \gamma_j(-\mathcal{L}_i x + W_i \sin(\omega_i t) - \dot{x}_j - \dot{z}_j)$$

which can be rewritten as

$$\ddot{x}_j = \gamma_j(-\mathcal{L}_i x - \dot{x}_j - \dot{z}_j) + \gamma_j W_i \sin(\omega_i t). \tag{2.13}$$

According to the linearity property of the DFT

$$\mathcal{F}\left[\frac{\ddot{x}_j(kT_s)}{W_i}\right] = \mathcal{F}\left[\frac{\gamma_j(-\mathcal{L}_i x - \dot{x}_j - \dot{z}_j)}{W_i}\right] + \mathcal{F}[\gamma_j \sin(\omega_i\, kT_s)]. \tag{2.14}$$

For large values of W_i, the contribution of the first term of (2.14) is not relevant. Thus, the Fourier Transform in (2.14) can be approximated as

$$\mathcal{F}\left[\frac{\ddot{x}_j(kT_s)}{W_i}\right] \approx \mathcal{F}[\gamma_j \sin(\omega_i\, kT_s)]. \tag{2.15}$$

It is well known from the DFT theory that the *Fourier representation* in the frequency domain of a general sinusoidal oscillation signal of the form $a\ \sin(\omega_0 t)$ has a peak of amplitude $\frac{a}{2}$ at the frequency $\frac{\omega_0}{2\pi}$. It follows that the DFT in (2.15) has a peak in the amplitude spectrum of $\frac{\gamma_j}{2}$ at a frequency $\frac{\omega_i}{2\pi}$. It follows straightforward that the value

$$\hat{\gamma}_j = 2\max\left(\mid \mathcal{F}[\frac{\ddot{x}_j(kT_s)}{W_i}] \mid_{f\approx\frac{\omega_i}{2\pi}} \right) \tag{2.16}$$

represents an approximation for the real value of γ_j depending on T_s and T_f, and this proves the theorem.

Remark 2.2. Since the frequency of the exerted signal may be arbitrary chosen by each agent, any node can be an observer, and simultaneous observation is possible, provided only that excitation signals are orthogonal. It follows straightforwardly from the Fourier Transform definition which is a composition of oscillatory function at the frequencies that are present in the analyzed data. Since the frequencies ω_i are different $\forall i$, each agent can separate the frequency of its own interest and is able to successfully estimate the desired γ values.

Remark 2.3. The accuracy of a generic γ estimated with the proposed method depends on the number of samples that are considered for computing the FFT. In this sense, let ξ be the error on the γ estimation, i.e.

$$\xi = |\hat{\gamma} - \gamma|. \tag{2.17}$$

It is clear that the value of ξ is decreasing with the increasing number of samples considered in the FFT. Therefore, as stated before it results

$$\lim_{N_s, T_f \to \infty} \xi = |\hat{\gamma} - \gamma| = 0.$$

where N_s represents the number of considered samples in $[0, T_f]$ observation interval.

2.5 Application

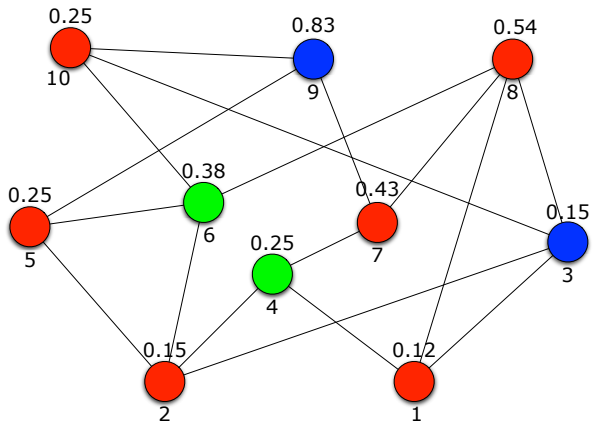


Fig. 2.3 A network of 10 agents where the Agent 6 (green) is investigating about the class (γ values) of the Agents 2,5,8,10 (red), while the agent 4 is investigating on agents 1,2,7 (red).

As an application of the proposed method consider a simple network consisting of 10 agents where agents move using the consensus protocol as in (2.4), with the Γ values specified as in Fig. 2.1. Suppose that Agent 6 (green) wants to classify its neighbors (red). In this respect it exerts a sinusoidal control action as in Eq.(2.7) with at a frequence of 25 Hz. By collecting $N_s = 1000$ sampled values of $\ddot{x}_i(kT_s)$ $i = 2, 5, 8, 10$ at the sampling time $T_s = 10^{-2}s$ ($kT_s \in [0, T_f]$, $T_f = N_s\, T_s$), agent 6 is able to compute the FFT transform $\mathcal{F}[\frac{\ddot{x}_j(kT_s)}{W_i}]$. As reported in Fig. 2.2, it is easy to verify that the peaks of amplitude at the frequency of the sine wave of the exerted control input u_6 represent an accurate estimation of the real γ values in Fig. 2.2.

As a further application, suppose that both agent 4 and agent 6 are investigating about the γ values of their neighbors in the same time (Fig. 2.3). To this aim they exert a control action with a different amplitude and frequency of the sine waves (Fig. 2.4 and Fig. 2.5). It is worth noting that since agent 2 repre-

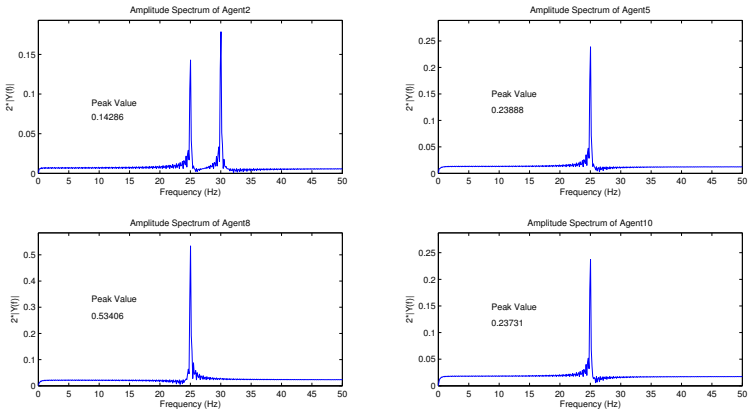


Fig. 2.4 The result of the frequency domain agent classification executed by agent 4. The Amplitude spectrums of the DFT report peak values at the frequency of 25 Hz which are an accurate approximation of the realγ values.

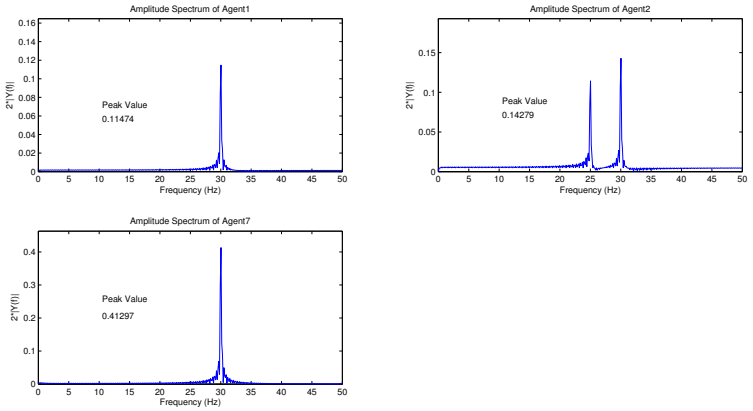


Fig. 2.5 The result of the frequency domain agent classification executed by agent 4. The Amplitude spectrums of the DFT report peak values at the frequency of 30 Hz which are an accurate approximation of the real γ values.

sent a common neighbor, the spectrum analysis for that agent show two peak values, one for each frequency exerted by each classifier agents (Remark 2.2).

Since the frequency of the exerted input is obviously known to each classifier agent, each of them is able to consider only the frequency of its own interest and correctly estimate the γ values.

Chapter 3

Hybrid Automata as a Model for Social Behaviors

In Chapter 2 it is introduced a model of heterogeneity based on scalar weights applied to the standard agreement algorithm allowing e.g. the description of systems composed of agents with e.g. different performances in mobility or communication, different clock rates, etc. Although this simple model may well serve the purpose for a limited number of robots and for simple tasks, a problem may arise when a more complex rule set is necessary to model larger and/or complex heterogenous systems as the human and animal societies. A "Behavior–based" society of robots can be built by giving a set of rules that each robot has to follow. Establishing rules for the physical interaction of robots, i.e. what behaviors are acceptable in the society, is a very challenging problem that has been explored only to a limited extent so far. Pioneering work on methods for describing rules for negotiating traffic and avoiding collisions are reported in [48], [49]. The idea of defining *group behaviors* was formalized and presented in [9], although in a cooperative framework. Behavior-based techniques are used for coordination in multi–agent RoboCup teams (see e.g. [50]). More recently, applications of protocol–based collision avoidance methods in a marine scenario has been presented in [51], where a multi–objective optimization is proposed to deal with situations where multiple rules are simultaneously activated.

One should observe that, in dealing with physically embodied autonomous agents such as robots, traditional automata theory is limited, because of the lack of expressivity power to model continuous dynamics. The *hybrid automata* formalism and verification techniques can be effectively used to that purpose. The complexity need to represent such behaviors can be successfully captured by hybrid models, in which a continuous–time dynamics describes the physical motion of each agent, while an event–based one describes the sequence of interactions with its neighbors. Rules must be based only on information that is locally available for each robot via communication with neighboring robots and proprioceptive sensing. In this chapter the formalization of a cooperation protocol by which societies of interacting robots can be described at a suitable abstraction level is proposed. The protocol allows the description of many distributed systems, including artificial systems (e.g. society of vehicles traveling along a highway and following different sets of driving rules) as well as approximations of systems inspired from Biology (e.g. society of Daceton Armigerum tree–dwelling workers ants). Some of the results reported in this chapter can be found in [A5].

3.1 An interaction model for a large variety of cooperative systems

To make the reading easier and illustrate what kind of systems this work deals with, and which are the issues involved, a running example is used. It consists of a society of vehicles traveling along a highway with m lanes and following different sets of driving rules (Fig. 3.1). Standard vehicles must strictly adhere either to the European, right–hand or the left–hand traffic rules (RH and LH species), while emergency vehicles are allowed to overtake both on the left and on the right (E species). LH (RH) species is described by the following rules: $\mathsf{rule}_1 \stackrel{\text{def}}{=}$ "proceed at the maximum speed along the rightmost (leftmost)

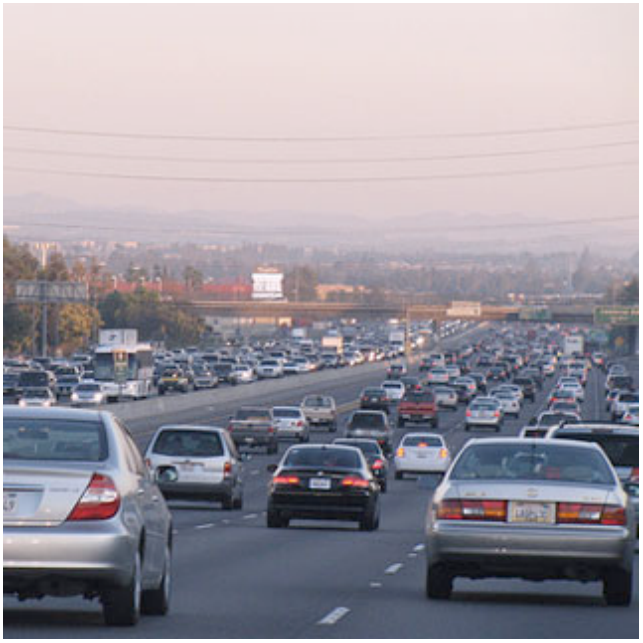


Fig. 3.1 Vehicles traveling along a highway.

free lane when possible"; $\mathsf{rule}_2 \stackrel{\text{def}}{=}$ "if a slower vehicle proceeds in front on the same lane, then overtake the vehicle if the next lane on the left (right) is free, or reduce the speed otherwise"; $\mathsf{rule}_3 \stackrel{\text{def}}{=}$ "as soon as the next lane on the right (left) becomes free, change to that lane"; $\mathsf{rule}_4 \stackrel{\text{def}}{=}$ "overtaking cars on the right (left) is forbidden". Rule_4 is ignored by the emergency traffic rules species, and rule_2 is modified as "if a slower vehicle proceeds in front on the same lane, then either overtake the car on the left if the next left lane is free, or overtake it on the right if the next right lane is free; otherwise reduce the speed". This basically allows an emergency vehicle to overtake everywhere it is possible. The allowed maneuvers are: $\mathsf{FAST} \stackrel{\text{def}}{=}$ "accelerate up to the maximum forward speed, while aligning to the center of the current lane"; $\mathsf{SLOW} \stackrel{\text{def}}{=}$ "decelerate down to null forward speed, while aligning to the center of the current lane";

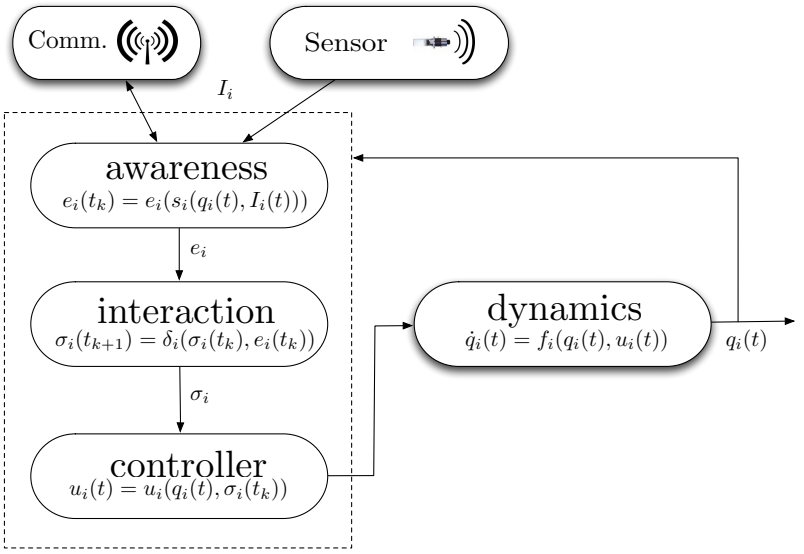


Fig. 3.2 Schema of the components of the Cooperative Robot Model

LEFT $\stackrel{\text{def}}{=}$ "move to the next lane on the left"; RIGHT $\stackrel{\text{def}}{=}$ "move to the next lane on the right".

In this work, this and other examples will be considered as particular instances of a general society model, which is formally described as follows. A "society" of robots is a collection of n robotic agents, $\mathcal{A}_1, \ldots, \mathcal{A}_n$, sharing an environment $\mathcal{Q}$, and each belonging to one species in a set $\{S^1, \cdots, S^p\}$. Each *species* is described by an interaction protocol $\mathcal{P}^r$ that specifies, for each agent $\mathcal{A}_i$, a motion model, a notion of neighborhood, a set of event–based interaction rules, and a local controller. A protocol is more formally described by the following components:

- A *dynamic map* $f_i : \mathcal{Q} \times \mathcal{U}_i \to T_{\mathcal{Q}}$, with $T_{\mathcal{Q}}$ the tangent space to $\mathcal{Q}$, describing how the agent's configuration $q_i \in \mathcal{Q}$ is updated:

$$\begin{cases} \dot{q}_i(t) = f_i(q_i(t), u_i(t)) \\ q_i(0) = q_i^0 \end{cases}, \; t \geq 0\,,$$

where q_i^0 is the initial configuration, and input $u_i \in \mathcal{U}_i$, with $\mathcal{U}_i$ denoting the set of admissible input values for the agent;

- A set of *topologies* $\eta_{i,1}, \cdots, \eta_{i,\kappa_i}$ on $\mathcal{Q}$, where $\eta_{i,j} : \mathcal{Q} \to 2^{\mathcal{Q}}$, defining the agent's neighborhood $N(q_i) = \cup_{j=1}^{\kappa_i} \eta_{i,j}(q_i)$, the neighbor set $M_i = \{\mathcal{A}_k \in \{\mathcal{A}_1 \cdots, \mathcal{A}_n\} \,|\, q_k \in N(q_i)\}$, the neighbor configuration set $I_i = \{q_k \in \mathcal{Q} \,|\, \mathcal{A}_k \in M_i\}$, and its *encoder map* $s_i : \mathcal{Q} \times \mathcal{Q}^{n_i} \to \mathbb{B}^{\kappa_i}$, where $n_i = \text{card}(I_i)$, and $\mathbb{B} \stackrel{\text{def}}{=} \{0, 1\}$, whose j–th component, $s_{i,j}$, is a logical–valued function returning true in the presence of an agent in the j–th topology $\eta_{i,j}(q_i)$, i.e.,

$$\begin{array}{l} s_{i,j} : \mathcal{Q} \times \mathcal{Q}^{n_i} \to \mathbb{B} \\ \qquad (q_i, I_i) \mapsto \sum_{q_k \in I_i} \mathbf{1}_{\eta_{i,j}(q_i)}(q_k) \end{array},$$

where $\sum$ represents the logical sum (*or*), and $\mathbf{1}_A(x)$ is the Indicator function of a set A;

- A finite event alphabet $E_i = \{e^{i,1}, \cdots, e^{i,\nu_i}\}$ and an event *detector map*

$$\begin{array}{l} e_i : \mathbb{B}^{\kappa_i} \to 2^{E_i} \\ \qquad s_i \mapsto \{e^{i,j} \in E_i \,|\, c_{i,j}(s_i) = 1\}\,, \end{array}$$

where each detector condition $c_{i,j}$ is a logical function

$$\begin{array}{l} c_{i,j} : \mathbb{B}^{\kappa_i} \to \mathbb{B} \\ \qquad s_i \mapsto \mathit{\Pi}_{k \in \gamma_{i,j}} s_{i,k}\, \mathit{\Pi}_{k \in \rho_{i,j}} \neg s_{i,k} \cdot \\ \qquad\quad \cdot \mathit{\Pi}_{k \in \mu_{i,j}} \mathbf{1}_{\lambda_{i,k}}(q_i)\, \mathit{\Pi}_{k \in \nu_{i,j}} \neg\, \mathbf{1}_{\lambda_{i,k}}(q_i)\,, \end{array} \tag{3.1}$$

with $\lambda_{i,1}, \cdots, \lambda_{i,h_i}$ constants in $2^{\mathcal{Q}}$, $\gamma_{i,j} \cup \rho_{i,j} = \{1, \cdots, \kappa_i\}$ and $\gamma_{i,j} \cap \rho_{i,j} = \emptyset$, $\mu_{i,j} \cup \nu_{i,j} = \{1, \cdots, h_i\}$ and $\mu_{i,j} \cap \nu_{i,j} = \emptyset$, and $\mathit{\Pi}$ and $\neg$ the logical product (*and*) and negation (*not*), respectively;

- A finite set of discrete states $\Sigma_i = \{\sigma^{i,1}, \ldots, \sigma^{i,p}\}$ and a deterministic *automaton* $\delta_i : \Sigma_i \times 2^{E_i} \to \Sigma_i$, describing how the agent's discrete state σ_i is updated:

$$\begin{cases} \sigma_i(t_{k+1}) = \delta_i(\sigma_i(t_k), e_i(t_{k+1})) \ , \quad t_k > 0 \\ \sigma_i(0) = \sigma_i^0 \end{cases} ,$$

  where $\sigma_i^0 \in \Sigma_i$ is the initial discrete state, and t_k is the k–th instant t at which e_i detects a new event;
- A *decoder map* (or controller) $u_i : \mathcal{Q} \times \Sigma_i \to \mathcal{U}_i$ implementing a feedback–based control law of the type

$$u_i(t) = u_i(q_i(t), \sigma_i(t_k)) \, .$$

Therefore, the state $(q_i, \sigma_i) \in \mathcal{Q} \times \Sigma_i$ of an agent $\mathcal{A}_i$, correctly following the protocol $\mathcal{P}^r$, must evolve according to the dynamics

$$\begin{aligned} \dot{q}_i(t) &= f_i(q_i(t), u_i(q_i(t), \sigma_i(t_k))) = f_i^*(q_i(t), \sigma_i(t_k)) \, , \\ \sigma_i(t_{k+1}) &= \delta_i(\sigma_i(t_k), e_i(s_i(q_i(t), I_i(t)))) = \\ &= \delta_i^*(\sigma_i(t_k), q_i(t), I_i(t)) \, , \end{aligned}$$

that can be written more compactly as

$$\begin{cases} (\dot{q}_i(t), \sigma_i(t_{k+1})) = \mathcal{H}_i^{(r)}(q_i(t), \sigma_i(t_k), I_i(t)) \, , \\ (q_i(0), \sigma_i(0)) = (q_i^0, \sigma_i^0) \, , \end{cases} \tag{3.2}$$

Fig. 3.3 Daceton Armigerum ants during the foraging process.

where $\mathcal{H}_i^{(r)} : \mathcal{Q} \times \Sigma_i \times \mathcal{Q}^{n_i} \to T_{\mathcal{Q}} \times \Sigma_i$ is the agent's *hybrid dynamic map* [52]. The schematization of the above described components is reported in Fig. 3.2

The details of protocols for the three species in the running example are reported in Section 3.3.

Furthermore, agents can be equipped with sensors providing information on other agents laying within a "visibility" region, described by a *visibility map* $\mathcal{V}_i : \mathcal{Q}^n \to 2^{\mathcal{Q}}$.
Local sensors are assumed to be chosen so that $\mathcal{V}_i(q_1, \cdots, q_n) \supseteq N(q_i)$, which ensures that each agent has complete knowledge of its own neighborhood.

3.2 Polymorfic Tree Dwelling Ant *Daceton Armigerum*

Ants of several species are organized in colonies, where "worker members" cooperate during the foraging process whenever a prey cannot be moved by a single ant [53]. In these species, sociality should affect not only the behavior of ants, but also the brains that generate and control the behavior. In partic-

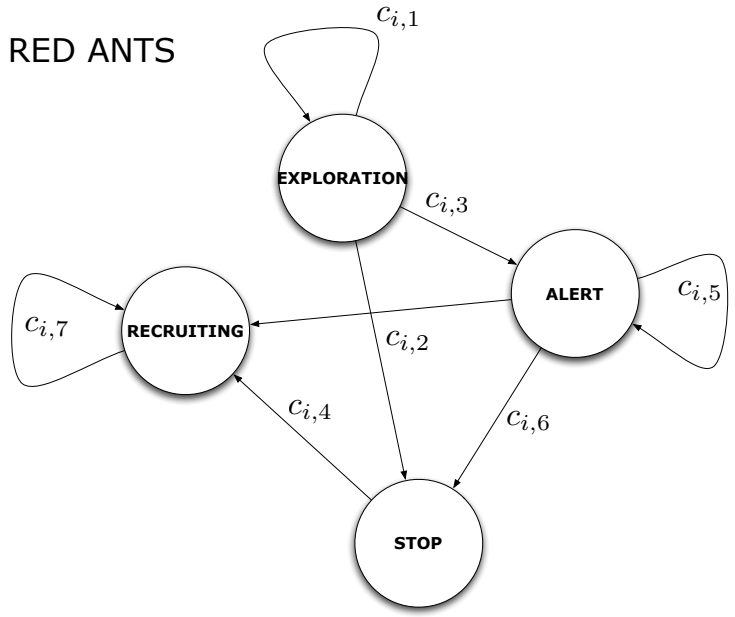


Fig. 3.4 The resulting Automaton for the Red ants species.

ular, the brain composition might reflect the behavioral specialization of the ant colonies. Chemical (pherormones) and mechanical communication (vibration, touch) among nestmates are hallmarks of a social lifestyle, and one might expect that sensory capabilities in ants are well developed [54, 55].

This work focuses on the foraging behavior of the polymorphic tree dwelling ant *Daceton Armigerum* which lives and forages almost completely on trees and forms colonies of polymorphic workers (Fig. 3.3). This process involves the recruitment of nestmates of the same colony, by issuing a *distinct*, colony–dependent visual or chemical marking. Suppose that two ant colonies exist, a *Green* and a *Red* one. Green ants start moving around the prey to inform their neighbors of their impossibility to move it, whereas Red ants stop in front of it. A generic ant $\mathcal{A}_i$'s configuration is $q_i = (x_i, y_i, \theta_i, v_i)$, where (x_i, y_i) is the position of the ant's center, θ_i is its orientation w.r.t. the x–axis, and v_i is its speed of motion, and evolves following the dynamics of Eq. (3.3). Both species

$\tau_i^{k,w}$	$s_{i,j}$[RED]	$s_{i,j}$[GREEN]
EXPLORATION $\rightarrow$ EXPLORATION	$c_{i,1} = \neg s_{i,1}\, \neg s_{i,2}$	$c_{i,1} = \neg s_{i,1}\, \neg s_{i,2}$
EXPLORATION $\rightarrow$ STOP	$c_{i,2} = s_{i,1}$	$c_{i,2} = s_{i,1}$
EXPLORATION $\rightarrow$ ALERT	$c_{i,3} = \neg s_{i,1}\, s_{i,2}\, \neg s_{i,3}$	$c_{i,3} = \neg s_{i,1}\, s_{i,2}\, \neg s_{i,3}$
STOP $\rightarrow$ RECRUITING	$c_{i,4} = s_{i,1}$	$c_{i,4} = s_{i,1}$
ALERT $\rightarrow$ ALERT	$c_{i,5} = \neg s_{i,3}$	$c_{i,5} = \neg s_{i,3}$
ALERT $\rightarrow$ RECRUITED	–	$c_{i,6} = \neg s_{i,1}\, s_{i,3}$
ALERT $\rightarrow$ STOP	$c_{i,6} = s_{i,1}$	–
RECRUITED $\rightarrow$ RECRUITED	–	$c_{i,7} = \neg s_{i,1}$
RECRUITED $\rightarrow$ STOP	–	$c_{i,8} = s_{i,1}$
RECRUITING $\rightarrow$ RECRUITING	$c_{i,7} = s_{i,1}$	$c_{i,9} = s_{i,1}$

Table 3.1 Transitions of the automaton of $\mathcal{A}_i$ in the ant example.

share the following set of rules: $\mathsf{rule}_1 \stackrel{\text{def}}{=}$ "proceed along a casual direction until a prey is found or a visual signal from a nestmate is received", $\mathsf{rule}_2 \stackrel{\text{def}}{=}$ "if a visual signal is received from a nestmate, go toward the nestmate and verify the actual existence of a nearby prey", $\mathsf{rule}_3 \stackrel{\text{def}}{=}$ "if a prey has been found, then issue a visual signal to recruite other nestmates". Their allowed maneuvers are: EXPLORATION $\stackrel{\text{def}}{=}$ "move straight along a casual direction", STOP $\stackrel{\text{def}}{=}$ "remain fixed", ALERT $\stackrel{\text{def}}{=}$ "go toward the nestmate", RECRUITING $\stackrel{\text{def}}{=}$ "issue the visual signal to recruit neighboring nestmates", RECRUITED $\stackrel{\text{def}}{=}$ "come closer to a nestmate and check that there is a prey". Consider, for each ant $\mathcal{A}_i$, the logical variables: $s_{i,1} \stackrel{\text{def}}{=}$ "there is a prey in front of $\mathcal{A}_i$", $s_{i,2} \stackrel{\text{def}}{=}$ "a nestmate of $\mathcal{A}_i$ has issued the recruiting signal", and $s_{i,3} \stackrel{\text{def}}{=}$ "$\mathcal{A}_i$ is in the recruiting point" (Fig. 3.4). The two species only differ from the way these logical variables, representing atoms, are combined together into events (see Table 3.1).

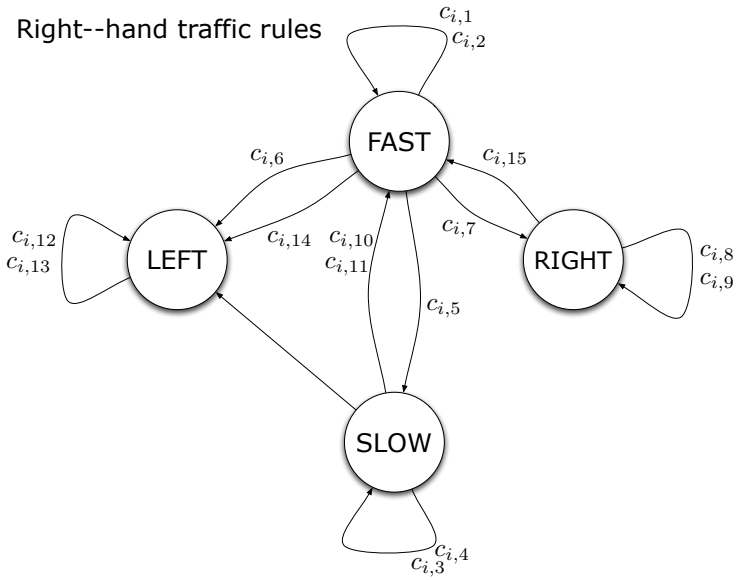


Fig. 3.5 The resulting Automaton for the RH species.

3.3 Vehicle in Highways

As introduced in Section 3.1, the considered robotic society is composed of vehicles belonging to the right–hand, left–hand, or emergency traffic rule species. For space reasons only the complete specification of the right-hand species is reported below. The environment is $\mathcal{Q} = R^2$. The configuration of the generic agent $\mathcal{A}_i$ is $q_i = (x_i, y_i, v_i, \theta_i)$ and is updated according to the dynamic map

$$\begin{aligned} f_i : \mathcal{Q} \times \mathcal{U}_i &\to T_{\mathcal{Q}} \\ q_i &\mapsto (\cos(\theta_i)v_i, \sin(\theta_i)v_i, a_i, \omega_i)^T \end{aligned}, \tag{3.3}$$

where $\mathcal{U}_i = R^2$.

We need to introduce a topology $\eta_{i,1}(q_i)$ representing a region in the immediate front of the agent, a topology $\eta_{i,2}(q_i)$ for a region on its left, a topology

$\eta_{i,3}(q_i)$ for a region on its right, and a topology $\eta_{i,4}(q_i)$ for a region on its back (Fig. 3.7). These are formally described as

$$
\begin{aligned}
\eta_{i,1} &: \mathcal{Q} \to 2^{\mathcal{Q}} \\
&\quad q_i \mapsto \{(x, y, \theta, v) \,|\, x_i \le x \le x_i + d_f, \\
&\qquad\qquad \lfloor \tfrac{y_i}{w} \rfloor w \le y \le \left(\lfloor \tfrac{y_i}{w} \rfloor + 1 \right) w \} \ , \\
\eta_{i,2} &: \mathcal{Q} \to 2^{\mathcal{Q}} \\
&\quad q_i \mapsto \{(x, y, \theta, v) \,|\, x_i - d_b \le x \le x_i + d_f, \\
&\qquad\qquad \left(\lfloor \tfrac{y_i}{w} \rfloor + 1 \right) w \le y \le \left(\lfloor \tfrac{y_i}{w} \rfloor + 2 \right) w \} \ , \\
\eta_{i,3} &: \mathcal{Q} \to 2^{\mathcal{Q}} \\
&\quad q_i \mapsto \{(x, y, \theta, v) \,|\, x_i - d_b \le x \le x_i + d_f, \\
&\qquad\qquad \left(\lfloor \tfrac{y_i}{w} \rfloor - 1 \right) \le y \le \lfloor \tfrac{y_i}{w} \rfloor w \} \ , \\
\eta_{i,4} &: \mathcal{Q} \to 2^{\mathcal{Q}} \\
&\quad q_i \mapsto \{(x, y, \theta, v) \,|\, x_i - d_b \le x \le x_i, \\
&\qquad\qquad \lfloor \tfrac{y_i}{w} \rfloor w \le y \le \left(\lfloor \tfrac{y_i}{w} \rfloor + 1 \right) w \} \ ,
\end{aligned}
$$

where w is the lane width, d_f and d_b are a forward and backward safety distances, and $\lfloor \cdot \rfloor$ returns the nearest lower integer of the argument. Thus, the encoder map is $s_i : \mathcal{Q} \times \mathcal{Q}^{n_i} \to \mathbb{B}^4$, $s_i = (s_{i,1}, \cdots, s_{i,4})$, and the agent's neighborhood is $N(q_i) = \eta_{i,1}(q_i) \cup \cdots \cup \eta_{i,4}(q_i)$.

Moreover, we need to introduce two constants $\lambda_{i,1}, \lambda_{i,2}$ representing the left–most and right–most lanes, respectively, and two constants $\lambda_{i,3}, \lambda_{i,4}$ representing the current target lane's left and right edges, respectively:

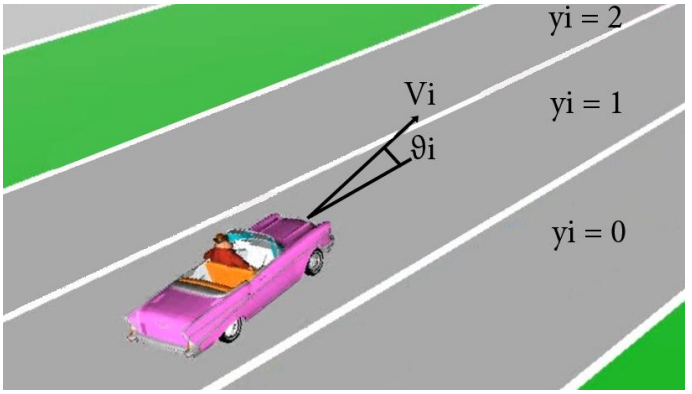


Fig. 3.6 Configuration a generic vehicle $\mathcal{A}_i$.

$$\lambda_{i,1} = \{(x, y, \theta, v) \,|\, (m-1)w \leq y \leq m\, w\} \ ,$$
$$\lambda_{i,2} = \{(x, y, \theta, v) \,|\, 0 \leq y \leq w\} \ ,$$
$$\lambda_{i,3} = \left\{(x, y, \theta, v) \,|\, y = \left(\left\lfloor \frac{y_i(t_k)}{w} \right\rfloor + 1\right) w\right\} \ ,$$
$$\lambda_{i,4} = \left\{(x, y, \theta, v) \,|\, y = \left\lfloor \frac{y_i(t_k)}{w} \right\rfloor w\right\} \ .$$

The event alphabet is $E_i = \{e^{i,1}, \cdots, e^{i,13}\}$ and the event map is

$$e_i : \mathbb{B}^{13} \to 2^{E_i}$$
$$\begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \mapsto \emptyset \,, \quad \begin{pmatrix} 1 \\ \vdots \\ 0 \end{pmatrix} \mapsto \{e^{i,1}\}$$
$$\begin{pmatrix} 0 \\ 1 \\ \vdots \end{pmatrix} \mapsto \{e^{i,2}\} \,, \cdots \,, \quad \begin{pmatrix} 0 \\ \vdots \\ 1 \end{pmatrix} \mapsto \{e^{i,13}\}.$$

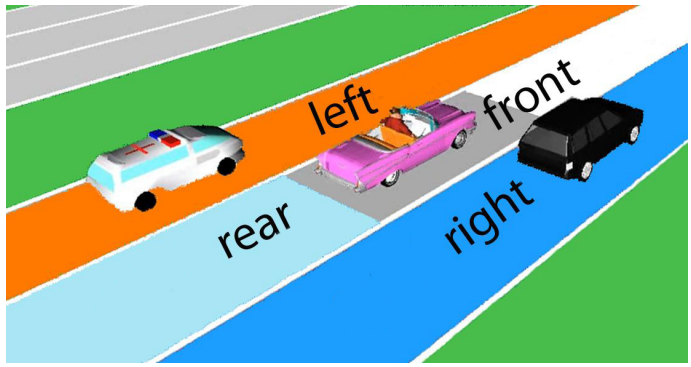


Fig. 3.7 Neighborhood of a generic vehicle $\mathcal{A}_i$.

with event conditions given by

$$c_{i,1} = \neg s_{i,1} s_{i,3}\,, \quad c_{i,2} = \neg s_{i,1} \lambda_{i,2}\,, \quad c_{i,3} = s_{i,1} s_{i,2}\,,$$

$$c_{i,4} = s_{i,1} s_{i,4}\,, \quad c_{i,5} = s_{i,1} \lambda_{i,1}\,, \quad c_{i,6} = s_{i,1} \neg s_{i,2} \neg s_{i,4}\, \neg \lambda_{i,1}\,,$$

$$c_{i,7} = \neg s_{i,1}\, \neg s_{i,3}\, \neg \lambda_{i,2}\,, \quad c_{i,8} = \neg s_{i,1}\,, \quad c_{i,9} = \lambda_{i,3}\,,$$

$$c_{i,10} = s_{i,1}\, \neg \lambda_{i,3}\,, \quad c_{i,11} = s_{i,1}\,, \quad c_{i,12} = \lambda_{i,4}\,,$$

$$c_{i,13} = \neg s_{i,1}\, \neg \lambda_{i,4}\,.$$

The finite set of discrete states is $\Sigma_i = \{\mathsf{FAST}, \mathsf{SLOW}, \mathsf{LEFT}, \mathsf{RIGHT}\}$ $(p = 4)$ and the automaton's dynamics is

$$
\begin{aligned}
\delta_i : \Sigma_i \times 2^{E_i} &\to \Sigma_i \\
(\mathsf{FAST}, e^{i,1}), (\mathsf{FAST}, e^{i,2}) &\mapsto \mathsf{FAST}\,, \\
(\mathsf{FAST}, e^{i,3}), (\mathsf{FAST}, e^{i,4}), (\mathsf{FAST}, e^{i,5}) &\mapsto \mathsf{SLOW}\,, \\
(\mathsf{FAST}, e^{i,6}) &\mapsto \mathsf{LEFT}\,, \\
(\mathsf{FAST}, e^{i,7}) &\mapsto \mathsf{RIGHT}\,, \\
(\mathsf{SLOW}, e^{i,8}) &\mapsto \mathsf{FAST}\,, \\
(\mathsf{SLOW}, e^{i,3}), (\mathsf{SLOW}, e^{i,4}), (\mathsf{SLOW}, e^{i,5}) &\mapsto \mathsf{SLOW}\,, \\
(\mathsf{SLOW}, e^{i,6}) &\mapsto \mathsf{LEFT}\,, \\
(\mathsf{LEFT}, e^{i,8}), (\mathsf{LEFT}, e^{i,9}) &\mapsto \mathsf{FAST}\,, \\
(\mathsf{LEFT}, e^{i,10}) &\mapsto \mathsf{LEFT}\,, \\
(\mathsf{RIGHT}, e^{i,11}), (\mathsf{LEFT}, e^{i,12}) &\mapsto \mathsf{FAST}\,, \\
(\mathsf{RIGHT}, e^{i,13}) &\mapsto \mathsf{RIGHT}\,,
\end{aligned}
$$

with initial state $\sigma_i^0 = \mathsf{FAST}$.

The decoder map is $u_i : \mathcal{Q} \times \Sigma_i \to \mathcal{U}_i$, $u_i = (a_i, \omega_i)$, with

$$
\begin{aligned}
a_i \; &: \mathcal{Q} \times \Sigma_i \to \mathbb{R} \\
&(q_i, \mathsf{FAST}), (q_i, \mathsf{LEFT}), \\
&(q_i, \mathsf{RIGHT}) \qquad\qquad \mapsto \begin{cases} \bar{a} \text{ if } v_i < v_{max}^i \\ 0 \text{ otherwise} \end{cases}, \\
&(q_i, \mathsf{SLOW}) \mapsto \begin{cases} -\bar{a} \text{ if } v_i > 0 \\ 0 \quad \text{otherwise} \end{cases},
\end{aligned}
$$

$$\omega_i : \mathcal{Q} \times \Sigma_i \to \mathbb{R}$$

$$\begin{array}{ll} (q_i, \mathsf{FAST}), \\ (q_i, \mathsf{SLOW}) \end{array} \mapsto \Big((y^*(q_i) - y_i) \, \tfrac{\sin \theta_i}{\theta_i} - \mu \, \theta_i \Big) \, v_i \, ,$$

$$(q_i, \mathsf{LEFT}), \quad \mapsto \begin{cases} \bar{\omega} \text{ if } \theta_i < \theta_{max} \\ 0 \text{ otherwise} \end{cases} ,$$

$$(q_i, \mathsf{RIGHT}), \mapsto \begin{cases} -\bar{\omega} \text{ if } \theta_i > -\theta_{max} \\ 0 \quad \text{otherwise} \end{cases} ,$$

where $y^*(q_i) = \left(\lfloor \frac{y_i}{w} \rfloor + \frac{1}{2} \right) w$ is the current lane center, θ_{max} and v^i_{max} are the agent's maximum curvature angle and allowed speed, and μ, $\bar{a}$ and $\bar{\omega}$ are positive constants. The other species share the same components described above except for the event conditions and the automaton's dynamics (Fig. 3.4). Some of these differences are highlighted in Table 3.2.

Finally, the visibility map returns the set of configurations laying within a distance R_i and that are not hidden by other cars (for its computation see e.g. the known sweeping line algorithm in [56]). A formal description of the map is avoided for space reasons.

$\tau_i^{k,w}$	$s_{i,j}$[Right–hand]	$s_{i,j}$[Left–hand]	$s_{i,j}$[Emergency]
FAST $\rightarrow$ FAST	$c_{i,1} = \neg s_{i,1}\, s_{i,3}$	$c_{i,1} = \neg s_{i,1}\, s_{i,2}$	$c_{i,1} = \neg s_{i,1}$
	$c_{i,2} = \neg s_{i,1} \lambda_{i,2}$	$c_{i,2} = \neg s_{i,1}\, s_{i,4}$	$c_{i,2} = \neg s_{i,1}\, \lambda_{i,1}$
FAST $\rightarrow$ SLOW	$c_{i,3} = s_{i,1}\, s_{i,2}$	$c_{i,3} = s_{i,1}\, s_{i,3}$	$c_{i,3} = s_{i,1}\, s_{i,2}\, s_{i,3}$
	$c_{i,4} = s_{i,1}\, s_{i,4}$	$c_{i,4} = s_{i,1}\, \lambda_{i,2}$	$c_{i,4} = s_{i,1}\, s_{i,3}\, s_{i,4}$
	$c_{i,5} = s_{i,1}\, \lambda_{i,1}$	$c_{i,5} = s_{i,1}\, \lambda_{i,1}$	$c_{i,5} = s_{i,1}\, s_{i,2}\, \lambda_{i,2}$
	–	–	$c_{i,6} = s_{i,1}\, \lambda_{i,1}$
FAST $\rightarrow$ LEFT	$c_{i,6} = s_{i,1}\, \neg s_{i,2}\, \neg s_{i,4}\, \neg \lambda_{i,1}$	$c_{i,6} = \neg s_{i,1}\, \neg s_{i,2}, \neg s_{i,4}\, \neg \lambda_{i,1}$	$c_{i,7} = s_{i,1}\, \neg s_{i,2}\, \neg s_{i,4}\, \neg \lambda_{i,1}$
FAST $\rightarrow$ RIGHT	$c_{i,7} = \neg s_{i,1}\, \neg s_{i,3}\, \neg \lambda_{i,2}$	$c_{i,7} = s_{i,1}\, \neg s_{i,3}\, \neg \lambda_{i,2}$	$c_{i,8} = s_{i,1}\, s_{i,2}\, \neg s_{i,3}\, \neg \lambda_{i,2}$
	–	–	$c_{i,9} = s_{i,1}\, \neg s_{i,3}\, s_{i,4}\, \neg \lambda_{i,1}$

Table 3.2 Transitions starting from the FAST maneuver of the automaton of $\mathcal{A}_i$ in the highway example.

Chapter 4

Global Awareness and Consensus on Set–Valued Information

As introduced in Chapter 2, most of the problems attacked so far in the literature, can be formulated as consensus problems over continuous domains, where local agents exchange data that consist of scalars (such as a temperature or the concentration of a chemical) or vectors (e.g., positions or velocities). Models used differ mainly in the type of rule each agent uses to combine its own information with the one received from its neighbors in the communication graph. In the simplest case, the evolution of the network of agents can be described by a linear iterative rule

$$x(t+1) = A\,x(t) + B\,u(t)\,,$$

where $x \in \mathbb{R}^n$ is the system's state, u is an input vector, and A is a weight matrix that needs to comply with the available communication topology and is designed so that the network converges to a unique decision, i.e. $x(\infty) \to \alpha 1$, with α possibly depending on the initial system's state. Falling into this linear framework are most of the key papers on consensus [16, 17, 29]. By using more general nonlinear dynamical systems, other important schemes for achieving consensus on more complex functions of state variables can be accommodated for. For instance, the distributed algorithm based on the centroidal Voronoi tes-

sellation proposed by [57] allows a collection of mobile agents to be deployed within a given environment while maximizing the network's sensing ability.

However, new emerging issues in the field of distributed control entail defining consensus algorithms on different representations of the state of information (see e.g. [58]). As a first example, consider the problem of averaging a set of initial measures taken by a collection of distributed sensors with limited communication bandwidth, which can be solved via a consensus system where agents' state information is represented by *symbols* obtained through a logarithmic quantizer [59]. As a second example, consider the problem of estimating the value of a logical decision task depending on binary input events by a set of agents with limited visibility on the events, e.g. the detection of malicious users in a networked computer system by interaction of local observation monitors [60]. A solution to the problem can be obtained through use of the so–called *logical consensus* approach, according to which agents share binary estimates of the events, combine them according to a suitable logical iterative function, and finally reach an agreement on their values. An interesting problem that is related to consensus is that of studying how to disseminate information through a network where nodes can only elaborate and share data over finite fields [61].

Furthermore, other applications involve problems of increasing complexity where the state of information takes value in continuous, possibly infinite domains. For instance, consider the problem of clock synchronization in distributed loosely–coupled systems via message exchange, where each node has a confidence interval on the true value of time, although the true value may be outside this interval for some sources. Marzullo's algorithm [62], on which the ubiquitous Network Time Protocol (NTP) [63] is based, is an agreement algorithm which estimates the smallest interval consistent with the largest number of sources. The problem of simultaneous localization and mapping by a set of mobile robotic agents is another example, where the traditional approach of modeling each agent's uncertainty on the positions of visually extracted

features as additive or multiplicative signals is possible but not natural. As it was shown in [64], the problem can be solved by a consensus approach where agents exchange data represented by confidence *regions* containing the features' real positions. All these problems require that the information state of a network of n agents is a collection $X = (X_1, \cdots, X_n)^T$ of elements X_i, belonging to the power set $\mathcal{P}(\mathcal{X})$ of a discrete or continuous, finite or infinite set $\mathcal{X}$, which is iteratively updated according to a set–valued map $F = (F_1, \cdots, F_n)^T$, with $F_i : \mathcal{P}(\mathcal{X})^n \to \mathcal{P}(\mathcal{X})$, i.e.

$$X(t+1) = F(X(t))\,. \tag{4.1}$$

The evolution of such a set–valued iterative system can be extremely rich and complex in the general case. Consider e.g. the following set–valued iterative system of 6 agents, where each agent's state is $X_i \in Q$ and $Q = [0,1] \times [0,1]$ is the unit square, evolving according to the map

$$\begin{cases} X_1(t+1) = X_6(t) \cap T_{(a_1,b_1)}(X_3(t)) \\ X_2(t+1) = X_3(t) \cap T_{(a_2,b_2)}(X_1(t)) \\ X_3(t+1) = X_1(t) \cup T_{(a_3,b_3)}(X_5(t)) \\ X_4(t+1) = X_1(t) \cap \mathcal{C}(T_{(a_4,b_4)}(X_3(t)) \cup X_5(t)) \\ X_5(t+1) = \mathcal{C}(T_{(a_5,b_5)}(X_1(t))) \\ X_6(t+1) = T_{(a_6,b_6)}(X_2(t)) \end{cases} \tag{4.2}$$

where $T_{(h,k)}(X_i)$ is a translation of X_i by the vector $(h,k)^T$ (mod 1). As it can be shown by standard number theory arguments (see e.g. [65]), the behavior of such a system is chaotic if at least one of the ratios a_i/b_i is irrational (Fig. 4.1). Although the study of set–valued dynamic systems appears to be a formidable task in its full generality, in many applications of practical interest the set of

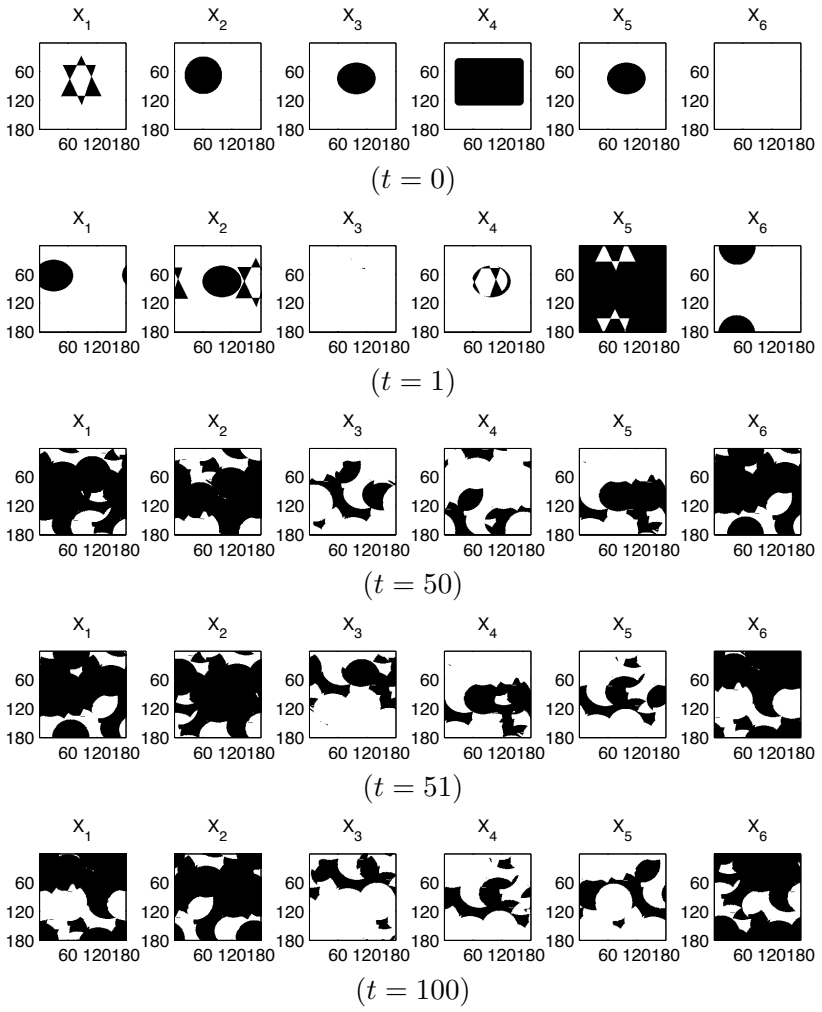


Fig. 4.1 Simulation run of a system with $n = 6$ agents running the set–valued map of Eq. 4.2, with $a_1 = 12/180$, $b_1 = 68/180$, $a_2 = 1/180$, $b_2 = 99/180$, $a_3 = \sqrt{3/10} \simeq 56/180$, $b_3 = 154/180$, $a_4 = 2/180$, $b_4 = 7/180$, $a_5 = 77/180$, $b_5 = 11/180$, $a_6 = 66/180$, $b_6 = 12/180$. The approximation $a_3 = \sqrt{3/10} \simeq 56/180$ implies that the behavior of the system is only approximately chaotic.

rules used in the iterative map are limited to specific classes, which may render analysis more tractable. Of particular relevance are certainly maps involving only Boolean operations, such as the set–theoretic union $\cup$, the intersection $\cap$, and the complement $\mathcal{C}(\cdot)$. Fortunately, it is possible to provide a reasonably simple study and characterization of such systems.

The main intent of the work is to show that information convergence in every instances of a Boolean iterative system can be studied in fundamentally the same way. This is achieved by extending the notions of convergence, local convergence, and contraction, already given in the binary domain [66, 67], to algebras of sets, taken with the union, intersection, and complement operations. The work presented provides results on local convergence in terms of properties of binary matrices for which analysis ([66, 67]) and synthesis (Chapter 5) results are available. Preliminary results reported in this chapter can be found in [A3, A2].

4.1 Boolean Dynamic Systems

This section and the remainder of the chapter focuses on a class of dynamic systems, namely Boolean Dynamic Systems (BDS), to define which the following concepts are needed.

Definition 4.1. A Boolean Algebra (BA) is a sextuple $(\tilde{\mathbb{B}}, \wedge, \vee, \neg, 0, 1)$, consisting of a domain set $\tilde{\mathbb{B}}$, equipped with two binary operations $\wedge$ (called "meet" or "and") and $\vee$ (called "join" or "or"), a unary operation $\neg$ (called "complement" or "not"), and two elements 0 (null) and 1 (unity) belonging to $\tilde{\mathbb{B}}$, s.t. the following axioms hold, for all elements $a, b, c \in \tilde{\mathbb{B}}$:

1. $a \vee (b \vee c) = (a \vee b) \vee c$, $a \wedge (b \wedge c) = (a \wedge b) \wedge c$ (associativity);
2. $a \vee b = b \vee a$, $a \wedge b = b \wedge a$ (commutativity);
3. $a \vee (a \wedge b) = a$, $a \wedge (a \vee b) = a$ (absorption);
4. $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$, $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$ (distributivity);

5. $a \vee \neg\, a = 1$, $a \wedge \neg\, a = 0$ (complementarity). ♦

From the first three pairs of axioms above, it follows that, for any two elements $a, b \in \tilde{\mathbb{B}}$, it holds that $a = a \wedge b$ if, and only if, $a \vee b = b$, which introduces a *partial order relation* $\leq$ among the elements of the domain. In particular, we will say that $a \leq b$, if, and only if, one of the two above equivalent conditions hold. Moreover, 0 and 1 are the least and greatest elements, respectively, of this partial order relation. Then, given any two elements $a, b \in \tilde{\mathbb{B}}$, the meet $a \wedge b$ and the join $a \vee b$ coincide with their infimum or supremum, respectively, w.r.t. $\leq$.

An element $a \in \tilde{\mathbb{B}}$ is referred to as a *scalar*. Consider the set $\tilde{\mathbb{B}}^n$ of Boolean *vectors* and the set $\tilde{\mathbb{B}}^{n \times n}$ of square Boolean *matrices*.

Definition 4.2. Given two *vectors* $v = (v_1, \ldots, v_n)$ and $w = (w_1, \ldots, w_n)$, and two square matrices $A = \{a_{i,j}\}$ and $B = \{b_{i,j}\}$, define the scalar product as

$$w \cdot v \stackrel{\text{def}}{=} \bigvee_{i=1}^{n} v_i \wedge w_i \in \tilde{\mathbb{B}}\,,$$

the product $A\,v$ as the vector whose i–th element is the scalar product between the i–th row of A and the vector v, and the product $A\,B$ as the matrix whose (i, j)–th element is the scalar product between the i–th row of A and the j–th column of B. In other words products between a matrix and a vector and between two matrices are computed in the usual way, substituting $+$ with $\vee$ and $\cdot$ with $\wedge$. ♦

Let denote with 0 the null scalar, vector, or matrix, according to the context. The above described partial order relation $\leq$ between any two elements of $\tilde{\mathbb{B}}$ can be extended to Boolean vectors and matrices by assuming component–wise evaluation.

Definition 4.3 (Boolean Dynamic Systems (BDS)). Given a BA, a Boolean dynamic system is an iterative system of the form in Eq. 4.1, whose state X is a vector in $\tilde{\mathbb{B}}^n$ and where F is a map combining the element of its input

argument to produce a new state vector, by using only the meet $\wedge$, the join $\vee$, and the complement $\neg$ operations of the BA itself.

Definition 4.4 (Linear BDS). A BDS is said to be *linear* if there exists a constant set–valued matrix $A \in \tilde{\mathbb{B}}^{n\times n}$ s.t., for all $X \in \tilde{\mathbb{B}}^n$, $F(X) = A\,X$.

For the following study, some definitions need to be given:

Definition 4.5 (Basis Vectors). The set of the vectors $e_1, e_2, \ldots, e_n$, with $e_j \in \tilde{\mathbb{B}}^n$ contains 1 in the j–th element and zeros elsewhere, is called the canonical basis of $\tilde{\mathbb{B}}^n$.

Definition 4.6 (Eigenvalues and Eigenvectors). A scalar $\lambda \in \tilde{\mathbb{B}}$ is an *eigenvalue* of a Boolean matrix $A \in \tilde{\mathbb{B}}^{n\times n}$ if there exists a vector $x \in \tilde{\mathbb{B}}^n$, called *eigenvector*, s.t. $A\,x = \lambda\,x$.

Definition 4.7 (Incidence Matrix). The incidence matrix of a Boolean map F is a Boolean binary matrix $B(F) = \{b_{i,j}\}$, where $b_{i,j} = 1$ if, and only if, the i–th component of $F(x)$ depends on the j–th component of the input vector x.

4.2 Set–valued Boolean Dynamic Systems - Global Convergence

As it is known from Stone's Representation Theorem [34], every BA is isomorphic (i.e. it possesses the same structural properties) to an algebra of sets. Therefore this work focuses on the following class of systems:

Definition 4.8 (Set–Valued Boolean Dynamic Systems (SVBDS)). A set–valued Boolean dynamic system is a BDS whose BA is defined by the sextuple $(\mathcal{P}(\mathcal{X}), \cup, \cap, \mathcal{C}(\cdot), \emptyset, \mathcal{X})$, where $\mathcal{P}(\mathcal{X})$ is the power set of a continuous, possibly unbounded set in $\mathcal{X}$, the operators $\cup$, $\cap$, $\mathcal{C}(\cdot)$ are the set–theoretic union, intersection, and complement, respectively, and $\emptyset$ is the emptyset. ♦

In the remainder of this section, the conditions under which these systems converge to a unique equilibrium will be studied.

Remark 4.1. It is worth noting that the class of SVBDM includes set–valued maps that also involve set difference $\setminus$ and symmetric difference S between any two inputs sets $X_i, X_j \in \mathcal{P}(\mathcal{X})$ or between an input set X_i and a constant set $A \in \mathcal{P}(\mathcal{X})$.

To show this, first recall that that the two operations can be expressed in terms of the basic operations of the BA. Set different can be expressed as

$$X \setminus Y = \{x \in X \text{ s.t. } x \notin Y\} = \{X \cap \mathcal{C}(Y)\},$$

and the the symmetric difference as

$$\begin{array}{c} S : \mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{X}) \to \mathcal{P}(\mathcal{X}) \\ (x, y) \mapsto (\mathcal{C}(x) \cap y) \cup (x \cap \mathcal{C}(y)) \end{array} . \tag{4.3}$$

Moreover, if F involves operations with k constant sets, $A_1, \cdots, A_k$, it is possible to consider a system with an augmented state vector $\tilde{X} = (X_1, \cdots, X_n, A_1, \cdots, A_k)^T$ and with the dynamic map

$$\tilde{F}(\tilde{X}) = \begin{pmatrix} F_1(X_1, \cdots, X_n, A_1, \cdots, A_k) \\ \vdots \\ F_n(X_1, \cdots, X_n, A_1, \cdots, A_k) \\ A_1 \\ \vdots \\ A_k \end{pmatrix},$$

which only involves Boolean operations on $\tilde{X}$. ♦

First, we introduce a metric over the Boolean vector space $\mathcal{P}(\mathcal{X})^n$. To this aim, consider the *Boolean vector distance*:

$$\begin{array}{c} \mathrm{d} : \mathcal{P}(\mathcal{X})^n \times \mathcal{P}(\mathcal{X})^n \to \mathcal{P}(\mathcal{X})^n \\ (X, Y) \mapsto (S(X_1, Y_1), \cdots, S(X_n, Y_n)) \end{array},$$

where X_i, Y_i are the i–th components of the vectors X and Y, respectively, and S is the symmetric difference defined in Eq. 4.3. Note that the Boolean vector distance d satisfies the following axioms

$$\begin{aligned} &\mathrm{d}(X, Y) = \mathrm{d}(Y, X), \ \forall\, X, Y\,, \\ &\mathrm{d}(X, Y) = \emptyset \ \text{ iff } \ X = Y\,, \\ &\mathrm{d}(X, Y) \ \subseteq \ \mathrm{d}(X, Z) \cup \mathrm{d}(Z, Y). \end{aligned}$$

Definition 4.9 (Incidence matrix). The incidence matrix of a set–valued map $F = (F_1, \cdots, F_n)^T$, with $F_i : \mathcal{P}(\mathcal{X})^n \to \mathcal{P}(\mathcal{X})$, denoted with $B(F)$, is a binary Boolean matrix whose elements are

$$b_{i,j} = \begin{cases} \mathcal{X} \ \text{ if } F_i \text{ depends on } X_j\,, \\ \emptyset \ \text{ otherwise}\,. \end{cases}$$

Proposition 4.1. *Given any two generic vectors $X, Y \in \mathcal{P}(\mathcal{X})^n$ and a set–valued map F, the following Boolean inequality holds*

$$\mathrm{d}(F(X), F(Y)) \subseteq B(F)\, \mathrm{d}(X, Y)\,. \tag{4.4}$$

Proof. Consider the i–th component of the inequality

$$\mathcal{D}_i(F_i(X_1,\cdots,X_n),F_i(Y_1,\cdots,Y_n)) \subseteq$$

$$\subseteq \mathcal{D}_i(F_i(X_1,\cdots,X_n),F_i(Y_1,X_2,\cdots,X_n)) \cup$$
$$\cup \mathcal{D}_i(F_i(Y_1,X_2,\cdots,X_n),F_i(Y_1,Y_2,X_3,\cdots,X_n)) \cup$$
$$\cdots$$
$$\cup \mathcal{D}_i(F_i(Y_1,\cdots,Y_{n-1},X_n),F_i(Y_1,\cdots,Y_{n-1},Y_n)) \subseteq$$

$$\subseteq b_{i,1}\,\mathcal{D}_1(X_1,Y_1) \cup b_{i,2}\,\mathcal{D}_2(X_2,Y_2) \cup \ldots$$
$$\ldots \cup b_{i,n}\,\mathcal{D}_n(X_n,Y_n)\,,$$

where $b_{i,j}$ are the elements of $B(F)$. The thesis immediately follows by repeating the computation for all rows i.

Proposition 4.2. *A Boolean matrix M satisfies the Boolean inequality*

$$\mathrm{d}(F(X),F(Y)) \subseteq M\,\mathrm{d}(X,Y)\,, \tag{4.5}$$

for all vectors $X,Y \in \mathcal{P}(\mathcal{X})^n$, if, and only if, $B(F) \subseteq M$.

Proof. The proof of sufficiency is trivial. Focus on the necessity and suppose, by absurd, that there exists a Boolean matrix $M = (m_{i,j})$ satisfying Eq. 4.2, but also admitting an element $m_{i,j} \subset b_{i,j}$, where $b_{i,j}$ is the corresponding element in the incidence matrix $B(F)$. This necessarily means that $b_{i,j} = \mathcal{X}$, and $m_{i,j} \subset \mathcal{X}$. Then, as $b_{i,j} = \mathcal{X}$, F_i depends on X_j and there must exist two vectors $X = (X_1,\cdots,X_j,\cdots,X_n)^T$, and $X' = (X_1,\cdots,Y_j,\cdots,X_n)^T$, with $X_j \neq Y_j$, s.t.

$$\mathcal{D}_i(F_i(X),F_i(X')) = \mathcal{X}\,.$$

The computation of X' from X is always possible, but it is omitted here. Basically, given X, and $F_i(X)$, we look for a vector Y s.t. $F_i(Y)$ is complementary to $F_i(X)$. This last quantity is upper bounded by

$$\left(\bigcup_{s=1}^{j-1} m_{i,s}\ \underbrace{\mathcal{D}_s(X_s, X_s)}_{=\emptyset}\right) \cup \underbrace{m_{i,j}}_{\subset\mathcal{X}}\ \mathcal{D}_j(X_j, Y_j) \cup$$
$$\cup \left(\bigcup_{s=j+1}^{n} m_{i,s}\ \underbrace{\mathcal{D}_s(X_s, X_s)}_{=\emptyset}\right) \subset \mathcal{X}\,,$$

that is a contradiction. Therefore, it must hold $b_{i,j} \subseteq m_{i,j}$, for all i and j.

Corollary 4.1. *For any two set–valued maps $F, G : \mathcal{P}(\mathcal{X})^n \to \mathcal{P}(\mathcal{X})^n$, the incidence matrix of the function composition $F(G(X))$ satisfies the Boolean inequality*

$$B(F(G)) \subseteq B(F)\, B(G)\,. \tag{4.6}$$

Proof. The proof trivially follows from above. Indeed, if $(F \circ G)_i$ depends on X_j, then there exists k s.t. F_i depends on X_k and G_k depends on X_j. Hence, $B(F)_{i,k} \cap B(G)_{k,j} = \mathcal{X}$ which in turn implies that $(B(F)\, B(G))_{i,j} = \mathcal{X}$.

Moreover, consider the following notion:

Definition 4.10 (Boolean spectrum). The Boolean spectrum of a Boolean matrix $A \in \mathcal{P}(\mathcal{X})^{n\times n}$ is set of the eigenvalues of A.

A first result about the spectrum of a Boolean map is the following:

Proposition 4.3. *A Boolean matrix $A \in \mathcal{P}(\mathcal{X})^{n\times n}$, $A = \{a_{i,j}\}$, admits the Boolean eigenvalue $\lambda = \emptyset$ if, and only if, it has at least one column for which the union of all its elements is less than $\mathcal{X}$, i.e. there exists $j \in \{1, \cdots, n\}$ s.t.*

$$\bigcup_{i=1}^{n} a_{i,j} \subset \mathcal{X}\,. \blacklozenge \tag{4.7}$$

Proof. (Sufficiency)
Suppose that j satisfies Eq. 4.7. We want to prove that $\lambda = \emptyset$ is a Boolean eigenvalue of A, i.e. there exists $X \neq \emptyset$ s.t. $A\,X = \emptyset\,X = \emptyset$. Consider a vector whose components are emptysets except for the j–th one. Then, we have $A\,X = A_j\,X_j$, where A_i is the i–th column of A, which we want to be the vector of emptysets. This last equation can be explicitly written as

$$\begin{aligned} a_{i,1} \cap X_j &= \emptyset\,, \\ a_{i,2} \cap X_j &= \emptyset\,, \\ &\vdots \\ a_{i,n} \cap X_j &= \emptyset\,, \end{aligned}$$

which holds if, and only if, it also happens that

$$(a_{1,j} \cap X_j) \cup (a_{2,j} \cap X_j) \cup \cdots \cup (a_{n,j} \cap X_j) = \emptyset\,,$$

and, by the distributivity property, that

$$\begin{aligned} (a_{1,j} \cup a_{2,j} \cup \cdots \cup a_{n,j}) \cap X_j &= \emptyset\,, \\ \textstyle\bigcup_{i=1}^{n} a_{i,j} \cap X_j &= \emptyset\,, \end{aligned}$$

which requires that the two sets are disjoint. Moreover, the value $\bar{X}_j = \mathcal{X} \setminus (\bigcup_{i=1}^{n} a_{i,j}) \neq \emptyset$ satisfies this condition and, due to the hypothesis in Eq. 4.7, is different from $\emptyset$, which implies that $X = (\emptyset, \cdots, \emptyset, \bar{X}_j, \emptyset, \cdots, \emptyset)^T$ is an eigenvector of A.

(Necessity)
Suppose that $\lambda = \emptyset$ is an eigenvalue of A. This implies that there exists $X \neq \emptyset$ s.t. $A\,X = \emptyset$. This means $\bigcup_{i=1}^{n} a_{i,j} \cap X_j = \emptyset$, for all j. This condition is trivially satisfied for every null component of X. For every other component

of X that is different than $\emptyset$, the component itself must be disjoint to the union of the sets composing the corresponding column of A. This implies that their union can not cover the entire set $\mathcal{X}$, which finally gives the thesis.

Remark 4.2. It is worth noting that, if A has a Boolean eigenvalue λ, with assigned eigenvector X, then, for every permutation P, the matrix $A' = P^T\, A\, P$ has the same eigenvalue, assigned with eigenvector $\mathcal{V} = P^T X$. Note that P is a permutation matrix in the classical sense, but where every 0 and 1 are replaced with $\emptyset$ and $\mathcal{X}$, respectively.

To prove this, observe that $A\, X = \lambda X$ for hypothesis. Left–multiplying by P^T, we have $P^T A\, X = \lambda\, P^T X$, and, from the identity $P^T\, P = I$ (I being the matrix with $\mathcal{X}$ on its diagonal and $\emptyset$ elsewhere), we have $(P^T\, A\, P)\, (P^T\, X) = \lambda\, (P^T\, X)$, which proves the statement. ♦

A complete characterization of the Boolean spectrum of a generic map is complex (see e.g. the work in [68]), whereas the following result is already available for a subclass of these maps:

Proposition 4.4. *A matrix $A \in \{\emptyset, \mathcal{X}\}^{n \times n}$ admits the Boolean eigenvalue $\lambda = \mathcal{X}$ if, and only if, there exist no permutation bringing A in strictly lower or upper triangular form.*

Proof. (Sufficiency) Suppose the existence of a permutation matrix P s.t. $A' \stackrel{def}{=} P^T\, A\, P$ is strictly lower triangular. Then, we need to prove that there exists no vector $X \neq \emptyset$ s.t. $A'\, X = \mathcal{X}\, X = X$. This trivially holds due to the form of matrix A'. Direct computation of the previous equation gives

$$\begin{aligned}
\emptyset &= X_1\,,\\
a'_{2,1} \cap X_1 &= X_2\,,\\
a'_{3,1} \cap X_1 \cup a'_{3,2} \cap X_2 &= X_3\,,\\
&\vdots\\
a'_{n,1} \cap X_1 \cup \cdots \cup a'_{n,n-1} \cap X_{n-1} &= X_n\,.
\end{aligned}$$

The only vector that solves the system of equations is $X = \emptyset$, which means that $\lambda = \mathcal{X}$ cannot be a Boolean eigenvector A.

(Necessity) We need to prove that, if $\lambda = \mathcal{X}$ is not an eigenvalue of A, then there exists a permutation that brings A in strictly lower triangular form.

Note that $\mathcal{X}$ is an eigenvalue of A if, and only if, A has a fixed point. So, let us start imposing that the vector $w = (\mathcal{X}, \ldots, \mathcal{X})^T$ is not a fixed point. Then, if A has not an empty row, the scalar product between every row of A and w is $\mathcal{X}$, and therefore w would be a fixed point. Then suppose that the i–th row of A is made of emptysets. We can now apply to A a permutation that exchanges the i–th row with the first one, and then exchanges the i–th with the first column. In this way we obtain a matrix where the first row is empty.

By induction, suppose that there exists a permutation matrix P s.t. $P^T\,A\,P$ has the form

$$\begin{pmatrix} \emptyset & \cdots & \emptyset & \emptyset \cdots & \emptyset \\ a'_{2,1} & \cdots & \emptyset & \emptyset \cdots & \emptyset \\ \vdots & \ddots & \vdots & \vdots \ddots & \vdots \\ a'_{i,1} & \cdots & a'_{i,i-1} & \emptyset \cdots & \emptyset \\ a'_{i+1,1} & & \cdots & & a'_{i+1,n} \\ \vdots & & \ddots & & \vdots \\ a'_{n,1} & & \cdots & & a'_{n,n} \end{pmatrix},$$

and consider the vector $v = (\emptyset, \ldots, \emptyset, \mathcal{X}, \ldots, \mathcal{X})^T$*, where the first* i *rows are null.* v *is not a fixed point of* $P^T A P$ *only if there exists* $j > i$ *s.t. the* j*–th row of* $P^T A P$ *has the form* $(a'_{j,1}, \ldots, a'_{j,i}, \emptyset, \ldots, \emptyset)$*. We can now apply to* $P^T A P$ *a permutation that exchanges the* j*–th row with the* i*–th one, and then exchanges the* j*–th with the* i*–th column. The inductive step is complete because we obtain a matrix of the form*

$$\begin{pmatrix} \emptyset & \cdots & \emptyset & \emptyset \cdots & \emptyset \\ a'_{2,1} & \cdots & \emptyset & \emptyset \cdots & \emptyset \\ \vdots & \ddots & \vdots & \vdots \ddots & \vdots \\ a'_{i+1,1} & \cdots & a'_{i+1,i} & \emptyset \cdots & \emptyset \\ a'_{i+2,1} & & \cdots & & a'_{i+2,n} \\ \vdots & & \ddots & & \vdots \\ a'_{n,1} & & \cdots & & a'_{n,n} \end{pmatrix},$$

which concludes the proof.

Proposition 4.5. *Let $A = \{a_{i,j}\}$ be a $n \times n$ matrix s.t. $a_{i,j} \in \{\emptyset, \mathcal{X}\}$. If $\mathcal{X} \notin \sigma(A)$, then*

$$\sigma(A) = \mathcal{P}(\mathcal{X}) \setminus \mathcal{X} .$$

Proof. By Remark 4.2, we can assume that A is strictly triangular. The eigenvector $v = (\emptyset, \cdots, \emptyset, \mathcal{C}(\lambda))^T$ is associated with eigenvalue λ, being $\emptyset = A\,v = \lambda\,v = \emptyset$, with $\lambda \neq \emptyset$, $v \neq \emptyset$.

With the following example, we show that the spectrum of a Boolean matrix may possess a structure that is impossible to find in $\mathbb{R}^n$, e.g. different eigenvalues can be associated with the same eigenvector, or the spectrum may be the entire set $\mathcal{P}(\mathcal{X})$.

Example 4.1. Consider the entire real interval $\mathcal{X} = (-\infty, \infty)$ and the two following matrices

$$A_1 = \begin{pmatrix} \emptyset & \{13\} \\ (17, 28] & \mathcal{X} \end{pmatrix} \;\; ; \;\; A_2 = \begin{pmatrix} [3, 5) & \mathcal{X} \\ \mathcal{X} & \{4\} \end{pmatrix} .$$

A_1 admits the eigenvalue $\lambda = \emptyset$ by Prop. 4.3, being the union of its first column's elements is less than $\mathcal{X}$, with associated eigenvectors $V_\lambda = (X, \emptyset)^T$, where X is any set in $(-\infty, 17] \cup (28, \infty)$. Moreover, A_2 does not admit the eigenvalue $\lambda = \emptyset$ by Prop. 4.3, while any scalar $\lambda \subseteq \mathcal{X} \setminus \emptyset$ is an eigenvalue of A_2, with associated eigenvector $V_\lambda = (X, X)^T$, with $X \subseteq \lambda$. ♦

Definition 4.11 (Contractive SBM). A SBM $F : \mathcal{P}(\mathcal{X})^n \to \mathcal{P}(\mathcal{X})^n$ is said to be *contractive* w.r.t. the vector distance if

$$\mathcal{X} \notin \sigma(B(F)) .$$

Remark 4.3. From Prop. 4.4, F is contractive if there exists a permutation matrix $P \in \{\emptyset, \mathcal{X}\}^{n \times n}$ s.t. $P^T B(F)\, P$ is strictly lower or upper triangular. ♦

Finally, a result characterizing the global convergence of a SBM F is the following:

Theorem 4.1. *F is contractive w.r.t. the vector distance d if, and only if, there exists a positive integer q s.t. F^q is a constant application.*

Proof. (Sufficiency) Being F contractive, $\mathcal{X} \notin \sigma(B(F))$ and $B(F)$ up to transformation $P^T B(F) P$, where P is a permutation matrix, is strictly triangular. Therefore, it is ensured the existence of a positive integer $q \leq n$ s.t. $(B(F))^q = \emptyset$. It also holds that

$$\emptyset \subseteq B(F^q) = B(F \cdots F) \subseteq B(F) \cdots B(F) = (B(F))^q .$$

Then, it must hold $B(F^q) = \emptyset$, which implies that the application F^q is independent of X and guarantees the existence of a point $\xi \in \mathcal{P}(\mathcal{X})^n$ s.t. $F^q(X) = \xi$, for every $X \in \mathcal{P}(\mathcal{X})$. Moreover, the rest of the iteration of F gives $F^{q+1}(\xi) = F^q(F(\xi)) = \xi$, as F^q is constant, but also $F^{q+1}(\xi) = F(F^q(\xi)) = F(\xi)$, which implies that $F(\xi) = \xi$. Being ξ a fixed point for F, we can also that ξ is unique. Suppose by absurd the existence of a second fixed point $\eta \in \mathcal{P}(\mathcal{X})^n$, $\eta \neq \xi$, for the application F. We have

$$\emptyset \subseteq \mathcal{D}(\xi, \eta) = \mathcal{D}(F(\xi), F(\eta)) \subseteq B(F)\, \mathcal{D}(\xi, \eta) \subseteq$$
$$\subseteq \cdots \subseteq (B(F))^q\, \mathcal{D}(\xi, \eta) = \emptyset ,$$

as $(B(F))^q = \emptyset$. Then, $\mathcal{D}(\xi, \eta) = \emptyset$, and $\xi = \eta$, which is a contradiction.

(Necessity) Suppose that F^q is a constant application. Then $B(F^q) = B(F)^q = \emptyset$. It follows that $B(F)$ does not admit a fixed point. Then, $\mathcal{X} \notin \sigma(B(F))$ and, by Prop. 4.4, $B(F)$ must be strictly triangular up to a transformation $P^T B(F) P$, where P is a permutation matrix. We have that $q \leq n$ and F is contractive.

Corollary 4.2. *If ξ is the unique equilibrium point, iterations of F starting from any initial state $X(0) \in \mathcal{P}(\mathcal{X})^n$ converge to ξ in at most q steps.* ♦

Remark 4.4. Theorem 4.1 practically implies that the dependence of the dynamic map F from its input arguments disappears after a finite number of steps q, i.e. $F^q(\xi)$ is constant for all ξ. For SVBDS this notion of convergence coincides with the typical notion of convergence toward an element of the space (in this case a set). Theorem 4.1 applies not only to SVBDS, but also to set–valued dynamic systems for which, however, it only expresses a sufficient condition for finite–time convergence that is satisfies by only a restricted class of systems.

Example 4.2. Consider a discrete–time dynamic system $X(t+1) = F(X(t))$, where $X = (X_1, X_2, X_3)^T \in \mathcal{P}(\mathcal{X})^3$, F and thus its incidence matrix are

$$F(X) = \begin{pmatrix} X_1 \cup (X_2 \cap X_3) \\ X_1 \cup \mathcal{C}(X_2) \\ \mathcal{C}(X_1) \cap \mathcal{C}(X_2) \cap \mathcal{C}(X_3) \end{pmatrix}, \quad B(F) = \begin{pmatrix} \mathcal{X} & \mathcal{X} & \mathcal{X} \\ \mathcal{X} & \mathcal{X} & \emptyset \\ \mathcal{X} & \mathcal{X} & \mathcal{X} \end{pmatrix}. \quad (4.8)$$

By Theorem 4.1, $B(F)$'s spectrum contains the eigenvalue $\lambda = \mathcal{X}$, and so the map is not contractive.

4.3 Binary Encoding of Set–Valued Boolean Dynamic Systems

While the results in Section 4.2 are very promising, a complete characterization of the spectrum of a general Boolean matrix is still far. Such an analysis is much simpler in the case of binary dynamic systems, which are BDS based on the simplest BA where $\tilde{\mathbb{B}} = \mathbb{B} = \{0, 1\}$ is the binary domain, $\wedge$ is the logical

product ("and") $\cdot$, $\vee$ is the logical sum ("or") $+$, and $\neg$ is the "not" operator. Local convergence for binary dynamic systems has been addressed by introducing the notion of a discrete derivative [66]. A possible generalization of this notion for SVBDS is represented by the so–called Boolean derivative, proposed in [A3]. However, this formulation of "derivative" gives rise to matrices containing not only the emptyset and the unity, for which results characterizing their spectrum cannot be easily obtained.

For this reason, in the remainder of the work, we pursue a different approach, which applies only to SVBDS, but allows their local convergence to fully characterized. In particular, we show how, once initial conditions are given, a SVBDS can be translated into a binary dynamic system

$$x(t+1) = f(x(t))\,, \tag{4.9}$$

where $x \in \mathbb{B}^\kappa$ is binary state vector, and $f : \mathbb{B}^\kappa \to \mathbb{B}^\kappa$ is a binary iterative map, and κ is a suitable dimension. We say that the system in Eq. 4.9 encodes a SVBDS of the form in Eq. 4.1, in the sense that every execution of the original system can be obtained by simulating the binary one and vice–versa. Consider the collection of sets

$$\begin{aligned}
Z_1 &= X_1 \cap X_2 \cap \cdots \cap X_{n-1} \cap X_n\,,\\
Z_2 &= X_1 \cap X_2 \cap \cdots \cap X_{n-1} \cap \mathcal{C}(X_n)\,,\\
Z_3 &= X_1 \cap X_2 \cap \cdots \cap \mathcal{C}(X_{n-1}) \cap X_n\,,\\
&\vdots\\
Z_{\kappa'-1} &= \mathcal{C}(X_1) \cap \mathcal{C}(X_2) \cap \cdots \cap \mathcal{C}(X_{n-1}) \cap X_n\,,\\
Z_{\kappa'} &= \mathcal{C}(X_1) \cap \mathcal{C}(X_2) \cap \cdots \cap \mathcal{C}(X_{n-1}) \cap \mathcal{C}(X_n)\,,
\end{aligned}$$

with $\kappa' = 2^n$. Let us denote with $Z = (Z_1, Z_2, \ldots, Z_\kappa)^T$ the vector composed, up to renaming, of the non–empty sets of the previous collection (note that in general $\kappa \leq \kappa'$). It is straightforward to verify that these sets are a partition of $\mathcal{X}$, i.e. $X_i \cap X_j = \emptyset$, and $X_1 \cup \cdots \cup X_n = \mathcal{X}$. In the remainder of this section, we show that every set $X_i \subseteq \mathcal{X}$, and indeed all unions, intersections, and complements obtained from the X_i can be obtained as the union of some of the above sets in Z, which allows us to find a binary encoding of the SBM F.

Consider an *encoder map* associating a set X_i with a binary vector whose h–th component is 1 if, and only if, X_i has non–null overlapping with the set Z_h, i.e.

$$\mathcal{L} : \mathcal{P}(\mathcal{X}) \to \mathbb{B}^\kappa$$

$$X_i \mapsto x_i = \left(x_1^i, \cdots, x_\kappa^i\right)^T \;,\; x_h^i = \begin{cases} 0 \text{ if } X_i \cap Z_h = \emptyset \,, \\ 1 \text{ otherwise} \,. \end{cases}$$

Thus, given a set X_i, the corresponding associated binary vector is $x_i = \mathcal{L}(X_i)$. The original set X_i can be obtained via the *decoder map*

$$\mathcal{L}^{-1} : \mathbb{B}^\kappa \to \mathcal{P}(\mathcal{X})$$

$$x_i \mapsto X_i = \bigcup_{h=1,\cdots,\kappa \,,\, x_h^i = 1} Z_h \,,$$

which allows us to write $X_i = \mathcal{L}^{-1}(x_i)$. Furthermore, consider any two sets, X_i and X_j, of the given collection, and their logical encoded vectors, $x_i = \mathcal{L}(X_i)$, and $x_j = \mathcal{L}(X_j)$. Consider first the their combination via set intersection:

$$X_i \cap X_j = \mathcal{L}^{-1}(x_i) \cap \mathcal{L}^{-1}(x_j) = \left(\bigcup_{h=1,\, x_h^i = 1}^{\kappa} Z_h\right) \cap \left(\bigcup_{l=1,\, x_l^j = 1}^{\kappa} Z_l\right) \,,$$

which can be expanded, by distributing the set intersection, as the union of the sets given by the intersection of one Z_h with one Z_l. As all Z_i are disjoint, only those Z_i appearing in both the original sets, X_i and X_j, remain in the intersection. Therefore, we can write

$$\begin{aligned} X_i \cap X_j &= \bigcup_{h=1\,,\,(x_h^i=1)\wedge(x_h^j=1)}^{\kappa} Z_h = \\ &= \bigcup_{h=1\,,\,x_h=1}^{\kappa} Z_h = \mathcal{L}^{-1}(x)\,, \text{with } x = x_i\,x_j\,, \end{aligned}$$

which proves the equivalence relation:

$$X_i \cap X_j \underset{\mathcal{L}^{-1}}{\overset{\mathcal{L}}{\rightleftharpoons}} x_i\,x_j\,. \tag{4.10}$$

Moreover, consider the two sets' combination via set union:

$$\begin{aligned} X_i \cup X_j &= \mathcal{L}^{-1}(x_i) \cup \mathcal{L}^{-1}(x_j) = \left(\bigcup_{h=1\,,\,x_h^i=1}^{\kappa} Z_h\right) \cup \left(\bigcup_{l=1\,,\,x_l^j=1}^{\kappa} Z_l\right) = \\ &= \bigcup_{h=1,(x_h^i=1\,\vee\,x_h^j=1)}^{\kappa} Z_h = \bigcup_{h=1,x_h=1}^{\kappa} Z_h = \\ &= \mathcal{L}^{-1}(x)\,,\ \text{with } x = x_i + x_j\,, \end{aligned}$$

which proves the equivalence relation:

$$X_i \cup X_j \underset{\mathcal{L}^{-1}}{\overset{\mathcal{L}}{\rightleftharpoons}} x_i + x_j\,. \tag{4.11}$$

Finally, consider the complementation of one of the two sets:

$$\mathcal{C}(X_i) = \mathcal{C}(\mathcal{L}^{-1}(x_i)) = \mathcal{C}\left(\bigcup_{h=1\,,\,x_h^i=1}^{\kappa} Z_h\right) = \bigcap_{h=1\,,\,x_h^i=1}^{\kappa} \mathcal{C}(Z_h)\,.$$

By definition $\mathcal{C}(Z_h)$ is the set of points not belonging to Z_h, that can be obtained as the union of all the other partition sets:

$$\begin{aligned}\bar{Z}_h = \mathcal{C}(Z_h) = \bigcup_{h'=1,h'\neq h}^{\kappa} Z'_h = Z_1 \cup Z_2 \cup \cdots \cup Z_{h-1} \cup Z_{h+1} \cup \cdots \cup Z_\kappa = \\ = \mathcal{L}^{-1}(z_1) \cup \mathcal{L}^{-1}(z_2) \cup \cdots \cup \mathcal{L}^{-1}(z_{h-1}) \cup \mathcal{L}^{-1}(z_{h+1}) \cup \cdots \cup \mathcal{L}^{-1}(z_\kappa) = \\ = \bigcup_{l=1,\alpha_{l,h}=1}^{\kappa} Z_l\,,\end{aligned}$$

with $\alpha_h = z_1 + z_2 + \cdots + z_{h-1} + z_{h+1} + \cdots + z_\kappa$. Easy computation gives a logical vector $\alpha_h = (1, \ldots, 1, 0, 1, \ldots, 1)^T$ containing all entries to 1 except for the h–th one. Finally, intersection of all $\bar{Z}_h$ yields:

$$\mathcal{C}(X_i) = \bigcap_{h=1\,,\,x_h^i=1}^{\kappa} \bar{Z}_h = \bigcap_{h=1\,,\,\alpha_h=1}^{\kappa} Z_h, \text{ with } \alpha = \alpha_1\,\alpha_2\,\ldots\alpha_r\,,$$

where r is the number of x_i's non–null components. As all these components are assigned with a logical vector α_l containing a null element at position l, and as all these components are considered, the sets that remain in the intersection are those not belonging to X_i, or in other words, for which $x_h^i = 0$. Hence, we have

$$\mathcal{C}(X_i) = \bigcup_{h=1\,,\,x_h^i=0}^{\kappa} Z_h = \bigcup_{h=1\,,\,x_h^i=1}^{\kappa} Z_h = \mathcal{L}^{-1}(y_i), \text{ with } y_i = \neg x_i\,,$$

which proves the equivalence relation:

$$\mathcal{C}(X_i) \underset{\mathcal{L}^{-1}}{\overset{\mathcal{L}}{\rightleftharpoons}} \neg\, x_i\,. \tag{4.12}$$

Remark 4.5. From the above results, it follows that the intersection (union, complement) of any two sets X_i and X_j is equivalent, via the encoder map, to the bitwise logical product (sum, complement) of the corresponding binary vectors $x_i = \mathcal{L}(X_i)$ and $x_j = \mathcal{L}(X_j)$.

We can now state the main result of this section in the following:

Theorem 4.2 (Binary Encoding of SBM). *A dynamic system of the form*

$$X(t+1) = F(X(t))\,,$$

where F is a generic SBM, with initial state $X(0) = X^0$, can be simulated by the binary dynamic system

$$x_{i,j}(t+1) = f_i(x_{1,j}(t), \cdots, x_{\kappa,j}(t))\,,$$

for $i = 1, \cdots, n$, $j = 1, \cdots, \kappa$, with

$$x(0) = \mathcal{L}(X(0)) \in \mathbb{B}^{n\times\kappa}\,,$$

and where $f = (f_1, \cdots, f_n)^T$ is obtained from F by replacing unions, intersections, and set complements with logical sums, products, and binary complements, respectively.

Proof. The proof straightforwardly follows by recursive application of the relations in Eq. 4.10, 4.11, and 4.12.

Example 4.3. Consider again the system of Example 4.2, where the unity is $\mathcal{X} = [0, \infty)$ and system's initial state is $X(0) = ([2, 5], [4, 7], [8, 11])^T$.

The system's state after one iteration step can be obtained according to Eq. 4.8, which yields

$$X(1) = F(X(0)) = \begin{pmatrix} [2,5] \cup ([8,11] \cap [4,7]) \\ [2,5] \cup \mathcal{C}([8,11]) \\ \mathcal{C}([2,5]) \cap \mathcal{C}([8,11]) \cap \mathcal{C}([4,7]) \end{pmatrix} = \\ = \begin{pmatrix} [2,5] \\ [0,5] \cup (7,\infty) \\ [0,2) \cup (7,8) \cup (11,\infty) \end{pmatrix}. \tag{4.13}$$

The same result can be obtained by using the binary encoding of the system. We need to consider the collection of sets

$$Z_1 = X_1 \cap X_2 \cap X_3 = \emptyset\,,$$

$$Z_2 = X_1 \cap X_2 \cap \mathcal{C}(X_3) = [4,5]\,,$$

$$Z_3 = X_1 \cap \mathcal{C}(X_2) \cap X_3 = \emptyset\,,$$

$$Z_4 = X_1 \cap \mathcal{C}(X_2) \cap \mathcal{C}(X_3) = [2,4]\,,$$

$$Z_5 = \mathcal{C}(X_1) \cap X_2 \cap X_3 = \emptyset\,,$$

$$Z_6 = \mathcal{C}(X_1) \cap X_2 \cap \mathcal{C}(X_3) = [5,7]\,,$$

$$Z_7 = \mathcal{C}(X_1) \cap \mathcal{C}(X_2) \cap X_3 = [8,11]\,,$$

$$Z_8 = \mathcal{C}(X_1) \cap \mathcal{C}(X_2) \cap \mathcal{C}(X_3) = [0,2) \cup (7,8) \cup (11,\infty)\,.$$

By excluding the emptysets and reordering the remaining ones, we obtain the partition sets

$$Z_1 = [4,5],\ Z_2 = [2,4),\ Z_3 = [5,7],$$

$$Z_4 = [8,11],\ Z_5 = [0,2) \cup (7,8) \cup (11,\infty)\,,$$

and thus each state X_i can be associated with a binary vector $x_i \in \mathbb{B}^5$. Based on Theorem 4.2, the original system can be simulated by a binary dynamic system of the form $x(t+1) = f(x(t))$, with $x = (x_1^T, x_2^T, x_3^T)^T$, and

$$\begin{aligned} f(x) = (&x_{1,1} + x_{2,1}x_{3,1}, \cdots, x_{1,5} + x_{2,5}x_{3,5}, \\ &x_{1,1}\bar{x}_{2,1}, \cdots, x_{1,5}\bar{x}_{2,5}, \\ &\bar{x}_{1,1}\bar{x}_{2,1}\bar{x}_{3,1}, \cdots, \bar{x}_{1,5}\bar{x}_{2,5}\bar{x}_{3,5}) \ . \end{aligned} \tag{4.14}$$

The initial state $x(0) = (x_1^T(0), x_2^T(0), x_3(0)^T)^T$ of the corresponding binary dynamic system is obtained by using the encoding map $\mathcal{L}$, i.e.

$$x_i(0) = \mathcal{L}(X_i(0)), \qquad \text{for } i = 1, 2, 3,$$

and is given by

$$x_1(0) = (1,1,0,0,0)\,,\ x_2(0) = (1,0,1,0,0)\,,\ x_3(0) = (0,0,0,1,0)\,.$$

The state of the binary dynamic system after one iteration step, i.e. $x(1) = f(x(0))$, is given by $x(1) = (x_1(1), x_2(1), x_3(1))^T$, with

$$x_1(1) = (1,1,0,0,0)\,,\ x_2(1) = (1,1,0,1,1)\,,\ x_3(1) = (0,0,0,0,1)\,,$$

which corresponds to the original system's state

$$X(1) = \mathcal{L}^{-1}(x(1)) = \begin{pmatrix} Z_1 \cup Z_2 \\ Z_1 \cup Z_2 \cup Z_4 \cup Z_5 \\ Z_5 \end{pmatrix},$$

being clearly equal to the value obtained in Eq. 4.13.

4.4 Convergence Revisited and Completed

In this section, we first show how the encoding technique presented in the previous section gives rise to the same conditions for global convergence that were shown in Section 4.2.

First recall from [67], results on global convergence of binary dynamic systems, involving the notion of *spectral radius* of a binary matrix $A \in \mathbb{B}^{n\times n}$, denoted with $\rho(A)$, which is its biggest eigenvalue in the sense of the order relation $\leq$, and the notion of binary vector distance

$$\begin{array}{l} d : \mathbb{B}^n \times \mathbb{B}^n \to \mathbb{B}^n \\ \qquad (x, y) \mapsto (x_1 \oplus y_1, \cdots, x_n \oplus y_n) \end{array},$$

where $\oplus$ is the exclusive disjunction

$$\begin{array}{l} \oplus : \mathbb{B} \times \mathbb{B} \to \mathbb{B} \\ \qquad (x_i, y_i) \mapsto (\neg x_i \, y_i) + (x_i \, \neg y_i) \end{array}.$$

Theorem 4.3. *A map $f : \mathbb{B}^n \to \mathbb{B}^n$ is contractive w.r.t. the binary vector distance d if, and only if, the following equivalent conditions hold:*

- $\rho(B(f)) = 0$*;*
- *there exists a permutation matrix P s.t. $P^T B(f)\, P$ is strictly lower or upper triangular;*
- $B(f)^q = 0$*, with $0 \leq q \leq n$.*

Moreover, if f is contractive, there exists a positive integer $q \leq n$ s.t. f^q, the composition of f with itself q times, is a constant map, i.e. it does not depend on the input vector. ♦

Consider a SVBDS characterized by a set–valued map $F : \mathcal{P}(\mathcal{X})^n \to \mathcal{P}(\mathcal{X})^n$ and its corresponding binary dynamic system characterized by the

function $f : \mathbb{B}^{n\times\kappa} \to \mathbb{B}^{n\times\kappa}$. We want to show that the properties of global and local contractivity of F can be investigated in the binary domain by studying the same properties of f.

Lemma 4.1. *Having denoted with $B(f)$ the incidence matrix of f, the following equivalence holds $\{B(F)\}_{i,j} = \mathcal{X}$ if, and only if,*

$$\{B(f)\}_{2^{n(i-1)}+1:2^{n(i-1)},2^{n(j-1)}+1:2^{n(j-1)}} = I$$

, where I is the identity matrix and $M_{i:j,k:l}$ indicates a sub-matrix of M obtained by extracting its rows from i to j and its columns from k to l.

Proof. Let us write the map F as follows:

$$F(X_1,\ldots,X_n) = \begin{pmatrix} F_1\left(X_{i_1^1},\ldots,X_{i_{k_1}^1}\right) \\ \vdots \\ F_n\left(X_{i_1^n},\ldots,X_{i_{k_n}^n}\right) \end{pmatrix},$$

where $X_{i_l^j}$ are the variables on which the l–th component of the image of F actually depends. By definition of the encoding map, we have

$$B(f) = \begin{pmatrix} 0\ldots & 0 & \overbrace{I}^{i_1^1} & 0 & \ldots & \overbrace{I}^{i_2^1} & \cdots & \overbrace{I}^{i_{k_1}^1} & \ldots 0 \\ \vdots\ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \vdots \\ 0\cdots & \underbrace{I}_{i_1^n} & 0 & \ldots & \underbrace{I}_{i_2^n} & 0 & \cdots & \underbrace{I}_{i_{k_n}^n} & \ldots 0 \end{pmatrix} \in \mathbb{B}^{n\kappa\times n\kappa},$$

where here 0 and I are the zero and identity matrices, respectively. The thesis easily follows since the matrix $B(F)$, by replacing 0 with $\emptyset$ and I with $\mathcal{X}$, has exactly the same form:

$$B(F) = \begin{pmatrix} \emptyset \dots \emptyset \overbrace{\mathcal{X}}^{i_1^1} \emptyset \dots \overbrace{\mathcal{X}}^{i_2^1} \cdots \overbrace{\mathcal{X}}^{i_{k_1}^1} \dots \emptyset \\ \vdots \dots \dots \dots \dots \dots \dots \dots \dots \dots \vdots \\ \emptyset \cdots \underbrace{\mathcal{X}}_{i_1^n} \emptyset \dots \underbrace{\mathcal{X}}_{i_2^n} \emptyset \cdots \underbrace{\mathcal{X}}_{i_{k_n}^n} \dots \emptyset \end{pmatrix} \in \{\emptyset, \mathcal{X}\}^{n \times n}$$

Theorem 4.4 (Global Convergence). *The dynamic map $F : \mathcal{P}(\mathcal{X})^n \to \mathcal{P}(\mathcal{X})^n$ of a SVBDS is contractive if, and only if, its encoding $\mathcal{L}(F) : \mathbb{B}^{n\kappa} \to \mathbb{B}^{n\kappa}$ is contractive.*

Proof. By Theorems 4.3 and 4.1, it is sufficient to prove that $\mathcal{X} \notin \sigma(B(F))$ if, and only if, $\rho(B(f)) = 0$. By Remark 4.3, $\mathcal{X} \notin \sigma(B(F))$ if, and only if, there exists a permutation matrix P s.t. $P^T B(F) P$ is strictly lower or upper triangular, and Theorem 4.3 assures that $\rho(B(f)) = 0$ if, and only if, there exists a permutation matrix p s.t. $p^T B(f) p$ is strictly lower triangular. By Lemma 4.1, it holds $\{B(F)\}_{ij} = \mathcal{X}$ if, and only if, $\{B(f)\}_{2^{n(i-1)}+1:2^{n(i-1)}, 2^{n(j-1)}+1:2^{n(j-1)}} = I$, which immediately implies that $\mathcal{X} \notin \sigma(B(F))$ if, and only if, $\rho(B(f)) = 0$.

Remark 4.6. $\rho(B(f)) = 0$ if, and only if, $\rho\left(\tilde{B}(F)\right) = 0$, where $\tilde{B}(f)$ is the matrix obtained substituting 1 to $\mathbb{X}$ and 0 to $\emptyset$. This can be easily seen by using the equivalent formulation in terms of permutation matrices given by Theorem 4.3 and Remark 4.3. ♦

Example 4.4 (Cont'd). Consider again the system of Example 4.2. Following the derivation of the associated logical system (Eq. 4.14), the incidence matrix of the corresponding binary dynamic system is

$$B(f) = \begin{pmatrix} I_8 & I_8 & I_8 \\ I_8 & I_8 & 0 \\ I_8 & I_8 & I_8 \end{pmatrix},$$

where I_8 is the identity matrix of dimension 8. According to Theorem 4.4, the system of the Example 4.2 is contractive if, and only if, its binary dynamic system is contractive. Based on Theorem 4.3, this system is not contractive, since $B(f)$ cannot be put in a strictly triangular form by a permutation matrix (there should be a zero row). Then, based on Theorem 4.4, we can conclude that the system of Example 4.2 is not contractive, as we obtained in Section 4.2 by using Theorem 4.1.

We now move on to attack the study of local convergence of a SVBDS. We first recall results from [66] on the local convergence of a binary map f about an *equilibrium point* x s.t. $f(x) = x$.

Definition 4.12 (Von–Neumann Neighborhood (VNN)). Given a point $x \in \mathbb{B}^n$, its VNN is the set $V(x)$ of all points differing from x in at most one component, i.e.

$$V(x) = \{x, \tilde{x}^1, \cdots, \tilde{x}^n\},$$

where $\tilde{x}^j = (x_1, \cdots, x_{j-1}, \neg x_j, x_{j+1}, \cdots, x_n)^T$.

Definition 4.13 (Discrete Derivative). The discrete derivative of a binary map $f : \mathbb{B}^n \to \mathbb{B}^n$ at a point $x \in \mathbb{B}^n$ is a binary matrix $f'(x) = \{f'_{i,j}\}$, s.t. $f'_{i,j} = 1$ if, and only if, a variation in the j–th component of x produces a variation in the i–th component of $f(x)$, i.e.,

$$f'_{i,j}(x) = f_i(x) \oplus f_i(\tilde{x}^j). \tag{4.15}$$

Definition 4.14. An equilibrium point $x \in \mathbb{B}^n$ is said to be *attractive* in its VNN $V(x)$ if the following two relations hold:

- $f(y) \in V(x)$, for all $y \in V(x)$;
- there exists $n \in \mathbb{N}$ s.t., for all $y \in V(x)$, $f^n(y) = x$.

Definition 4.15. A binary map f is said to be *locally convergent* at an equilibrium point x if x is attractive in its VNN.

Theorem 4.5. *An equilibrium point $x \in \mathbb{B}^n$ is attractive in its VNN if, and only if, the following two relations hold:*

- $\rho(f'(x)) = 0$,
- $f'(x)$ *contains at most one non–null element in each column.* ♦

Let us now consider the case of a generic SVBDS.

Definition 4.16. Given a vector $X = (X_1, \ldots, X_j, \ldots, X_n)^T \in \mathcal{P}(\mathcal{X})^n$, its j–th *neighbor* is

$$\tilde{X}^j = (X_1, \ldots, \mathcal{C}(X_j), \ldots, X_n)^T. \blacklozenge$$

Definition 4.17 (Complemented Neighborhood). Given a vector $X \in \mathcal{P}(\mathcal{X})^n$, the complemented neighborhood of X is the set $V(X)$ of all points that differ in at most one complemented component from X:

$$V(X) = \{X, \tilde{X}^1, \cdots, \tilde{X}^n\}. \blacklozenge$$

Definition 4.18. An equilibrium point X of $F : \mathcal{P}(\mathcal{X})^n \to \mathcal{P}(\mathcal{X})^n$ is *attractive* in its complemented neighborhood if

- $F(V(X)) \subset V(X)$;
- $F^n(Y) = X \ \ \forall Y \in V(X)$.

We can give the following result:

Theorem 4.6 (Local Convergence of SVBDS). *Given a SVBDS, where the dynamic map is $F : \mathcal{P}(\mathcal{X})^n \to \mathcal{P}(\mathcal{X})^n$, the equilibrium point $\bar{X} \in \mathcal{P}(\mathcal{X})^n$ is attractive in its complemented neighborhood if, and only if, the equilibrium*

$$\bar{x} = \mathcal{L}(\bar{X})$$

is attractive in the immediate neighborhood for the map $\mathcal{L}(F) : \mathbb{B}^{n\kappa} \to \mathbb{B}^{n\kappa}$.

Proof. The key point of the proof is the fact that, once an initial condition is given, we can equivalently study the dynamics of F in $\mathcal{P}(\mathcal{X})^n$ or that of f in $\mathbb{B}^{n\kappa}$. In particular it is straightforward to verify, by comparing Def. 4.12 and Def. 4.17, that

$$F(V(X)) \in V(X) \;\Leftrightarrow\; f(V(x)) \in V(x)$$

$$F^n(Y) = X,\;\; \forall Y \in V(X) \Leftrightarrow\; f^n(y) = x,\;\; \forall y \in V(x)\,.$$

Now Theorem 4.5 implies the thesis.

Example 4.5 (Cont'd). Consider again the system of Example 4.2, with a generic initial condition given by $X(0) = (A, B, C)^T$, where A, B, C are constant sets in $\mathcal{P}(\mathcal{X})$.

As discussed above, the system is not contractive (the spectrum of its incidence matrix $B(F)$ contains $\mathcal{X}$, or equivalently the spectral radius of the incidence matrix $B(f)$ of its encoding f is 1). Moreover, it is easy to verify that the state vector obtained with $A = B = \mathcal{X}$ and $C = \emptyset$ is an equilibrium of the system. The partition sets of the binary encoding are $Z_1 = Z_3 = \cdots = Z_8 = \emptyset$ and $Z_2 = \mathcal{X}$. Up to reordering of the above non–empty sets, we need to consider only $\kappa = 1$ partition set described by $Z_{1'} = \mathcal{X}$. The encoded binary vector state is $\bar{x} = \mathcal{L}(\bar{X}) = (x_1^T, x_2^T, x_3^T)^T$, with $x_1 = x_2 = 1$ and $x_3 = 0$, which is attractive in its immediate neighborhood for the encoded binary map

$$f(x) = \begin{pmatrix} x_{1,1} + x_{2,1}x_{3,1} \\ \\ x_{1,1}\bar{x}_{2,1} \\ \\ \bar{x}_{1,1}\bar{x}_{2,1}\bar{x}_{3,1} \end{pmatrix}.$$

Therefore, by Theorem 4.6, the equilibrium point $\bar{X}$ is attractive in its complemented neighborhood for the original system.

4.5 Application to Linear Boolean Maps

Consider a linear SVBDS system $X(t+1) = A\,X(t)$ of the form

$$\begin{pmatrix} X_1(t+1) \\ \vdots \\ X_n(t+1) \end{pmatrix} = \begin{pmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,n} \end{pmatrix} \begin{pmatrix} X_1(t) \\ \vdots \\ X_n(t) \end{pmatrix} =$$

$$= \begin{pmatrix} a_{1,1} \cap X_1(t) \;\cup\; \cdots \;\cup\; a_{1,n} \cap X_n(t) \\ \vdots \\ a_{n,1} \cap X_1(t) \;\cup\; \cdots \;\cup\; a_{n,n} \cap X_n(t) \end{pmatrix}.$$

As described in Remark 4.1, we can introduce the following extended system, where the elements of A are considered as additional time–constant variables of the state:

$$\begin{cases} X_1(t+1) = a_{1,1}(t) \cap X_1(t) \;\cup\; \cdots \;\cup\; a_{1,n}(t) \cap X_n(t)\,, \\ \qquad\qquad \vdots \\ X_n(t+1) = a_{n,1}(t) \cap X_1(t) \;\cup\; \cdots \;\cup\; a_{n,n}(t) \cap X_n(t)\,, \\ a_{1,n}(t+1) = a_{1,n}(t)\,, \\ \qquad\qquad \vdots \\ a_{n,n}(t+1) = a_{n,n}(t)\,. \end{cases}$$

By Remark 4.6, we need to study the incidence matrix of the corresponding binary dynamic system, which is given by

$$\left(\begin{array}{c|c} B(A) & T_1(A)\ T_2(A)\ \cdots\ T_n(A) \\ \hline 0_{n^2\times n} & I_{n^2\times n^2} \end{array}\right),$$

where

$$B(A) = \begin{pmatrix} B(a_{1,1}) & \cdots & B(a_{1,n}) \\ \vdots & \ddots & \vdots \\ B(a_{n,1}) & \cdots & B(a_{n,n}) \end{pmatrix},\quad T_i(A) = \begin{pmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \\ B(a_{i,1}) & \cdots & B(a_{i,n}) \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{pmatrix},$$

with $B(a_{i,j}) = 1$ if $a_{i,j} \neq 0$, and $B(a_{i,j}) = 0$ otherwise. By Theorem 4.1, global contractivity is equivalent to the fact that A^q is a constant application, which happens if, and only if, the matrix $B(A)$ is nilpotent. By Remark 4.3, this last condition holds if, and only if, there exists a permutation matrix P s.t. $P^T B(A) P$ is strictly triangular.

Theorem 4.7 (Consensus of Linear SVBDS). *A SVBDS of the form*

$$X(t+1) = A\,X(t)$$

possesses at least a set–valued equilibrium point that is a consensus state if, and only if,

$$\bigcap_{i=1}^{n} a_{i,1} \cup a_{i,2} \cup \cdots \cup a_{i,n} \neq \emptyset\,.$$

Proof: The point $1_n\,\xi$ is a consensus equilibrium state if, and only if, $A\,1_n\,\xi = 1_n\,\xi$, i.e.,

$$\begin{pmatrix} a_{1,1} \cup a_{1,2} \cup \cdots \cup a_{1,n} \\ \vdots \\ a_{n,1} \cup a_{1,2} \cup \cdots \cup a_{n,n} \end{pmatrix} \xi = \begin{pmatrix} \xi \\ \vdots \\ \xi \end{pmatrix} ,$$

which has a solution if, and only if, the intersection of the matrix rows is not the emptyset. ♦

Example 4.6. Consider a linear SVBDS, where the dynamic matrix is

$$A = \begin{pmatrix} [0,3] & \emptyset & \emptyset \\ \emptyset & \emptyset & [0,2] \\ \emptyset & [0,1] & \emptyset \end{pmatrix} .$$

The matrix A clearly satisfies the condition of Theorem 4.7, which implies that the system possesses one or more consensus equilibrium states, $X = 1_3\,\xi$, with $\xi \in \mathcal{P}(\mathcal{X})$. Moreover, any such consensus equilibrium state must satisfy the condition

$$\begin{cases} [0,3] \cap \xi = \xi \\ [0,2] \cap \xi = \xi \\ [0,1] \cap \xi = \xi \end{cases} ,$$

by which it is immediate to conclude that a set–valued vector state of the form $1_3\,\xi$ is a consensus equilibrium if, and only if, $\xi \subseteq [0,1]$.

It is worth noting that the system possesses an infinite number of equilibria that are not consensus states, as e.g. all state vectors of the form $(\xi, \emptyset, \emptyset)^T$, with $\xi \subseteq [0,3]$.

4.6 Application to Distributed Chart Estimation

Consider a mosaicking application involving reconstruction of a geographical chart, by using n balloon stations deployed over the area. Let $\mathcal{Q}$ be the set of points on the Earth surface with latitude and longitude comprised within $30°$N and $75°$N, and $30°$W and $50°$E, respectively, roughly corresponding to the European continent. By acquiring a spotlight–type image of the underneath surface, each station $\mathcal{A}_i$ is able to produce a local *estimated chart* $I_i(0) \subseteq \mathcal{Q}$, composed of a collection of connected sets representing the estimated emerged lands. Moreover, let $V_i(0) \subseteq \mathcal{Q}$ be a region representing the field–of–view of $\mathcal{A}_i$, i.e., the set of points that can be "seen" by $\mathcal{A}_i$. For simplicity we model each $V_i(0)$ as a circle centered at the projection of $\mathcal{A}_i$'s position on the Earth surface and having radius given by a constant d. Within its field–of–view, each $\mathcal{A}_i$ may incorrectly include portions of sea or neglect parts of existing lands in $I_i(0)$.

We assume a minimum *measurement multiplicity constraint* requiring that each point in $\mathcal{Q}$ lays within the intersection of at least $r > 0$ field–of–views; we further assume a *bounded detection error constraint* requiring that, in every set of r stations satisfying the measurement multiplicity constraint, at most γ of these stations may perform an incorrect detection. By assuming that each station is able to share data via communication with other neighboring stations, we seek a solution enabling an end–user on the ground, willing e.g. to use the chart information for navigation purpose, to efficiently and promptly poll its nearest station so as to retrieve a unique and *consistent* chart of the continent's surface.

A first solution can be found by following a *centralized* approach. In this solution a central processor with high computation and memory capacities must receive the estimated charts and visibility regions from all the stations, combine them into a global geographical chart, and then send this chart back to all stations. To cope with incorrect land detection, a well–known result from fault–

tolerance theory can be used [30], requiring that, for every point $q \in \mathcal{Q}$, the central processor uses the estimated charts received from at least $r' = 2\gamma + 1$ different stations including q in their field–of–view (thus it must hold the condition $r \geq r'$). Among these r' estimated charts, if γ is the maximum number of them that are possibly containing detection errors at least $\gamma + 1$ charts – the majority – contain correct information for that point. This procedure is formally described by the formula

$$I^* = \bigcup_{q \in \mathcal{Q}} \left(\bigcup_{H \in S_{\gamma+1}(K_q)} \left(\bigcap_{h \in H} I_h(0) \right) \right) , \tag{4.16}$$

where $S_\alpha(A)$ returns the set of all sets of cardinality α composed of elements in A, and

$$K_q = \{ i \in \{1, \cdots, n\} \,|\, q \in V_i(0) \} .$$

A region of global visibility can also be defined as follows, representing the region for which the centralized process has received sufficient information to perform high accuracy land detection:

$$V^* = \bigcup_{q \in \mathcal{Q}} \left(\bigcup_{H \in S_{\gamma+1}(K_q)} \left(\bigcap_{h \in H} V_h(0) \right) \right) . \tag{4.17}$$

While it effectively solves the problem, this centralized solution is unsatisfactory for at least three reasons: The first is *non–scalability*, since the amount of data to be elaborated by the processor requires computation and memory capacities increasing super–linearly with the number n of stations; secondly, the approach requires an explicit management of *message routing* in order to allow every station to reach and be reached from the central processor; third, it leads to the implementation of a system that has a *single–point of failure*.

By bearing in mind the centralized solution as an indicator of achievable performance, we seek a solution that is fully distributed, i.e., no central processor is used, and that requires no message routing, namely all stations must reach a consensus on the continent chart by exchanging messages only with

their one–hop neighbors. We assume a minimum *communication connectivity constraint* requiring that, for every point $q \in V_i(0)$, each $\mathcal{A}_i$ has at least $2\gamma + 1$ communication neighbors whose field–of–view comprises q. Let the *set–valued variable* $X_i \subseteq \mathcal{Q} \times \mathcal{Q}$ be the state of $\mathcal{A}_i$, and C_i the index set of its communication neighbors. A possible distributed solution can be obtained by using a SVBDS, where $\mathcal{A}_i$'s state is initialized with the value

$$X_i(0) = (I_i(0), V_i(0)) \ ,$$

and then iteratively updated according to the rule

$$\begin{cases} I_i(t+1) = \bigcup_{H \in S_{\gamma+1}(C_i)} \bigcap_{h \in H} (V_h(t) \cap I_h(t)) \ , \\ V_i(t+1) = \bigcup_{H \in S_{\gamma+1}(C_i)} \bigcap_{h \in H} V_h(t) \, . \end{cases} \tag{4.18}$$

We need to prove that, by means of the update rule in Eq. 4.18, each state X_i converges to the state (I^*, V^*). First note that, having defined the set $K = \{1, \cdots, n\}$, Eq. 4.16 and Eq. 4.17 can be rewritten as

$$\begin{cases} I^* = \bigcup_{H \in S_{\gamma+1}(K)} \bigcap_{h \in H} (V_h(0) \cap I_h(0)) \ , \\ V^* = \bigcup_{H \in S_{\gamma+1}(K)} \bigcap_{h \in H} V_h(0) \, . \end{cases}$$

It is straightforward to verify that the state $X^* = \mathbf{1}_n \, (I^*, V^*)$ is an equilibrium for the above SVBDS. While the system is not globally convergent to X^*, it is possible to show that such an equilibrium is attractive in a region that is large enough to tolerate up to γ incorrect land detections. Let us consider the general case in which three assumptions hold: 1) $I^* \neq \emptyset$ and $V^* \neq \emptyset$, indicating that some land exists and is in the field–of–view of at least $2\gamma + 1$ stations; 2) a portion of sea, $\mathcal{C}(I^*)$, is included in the global visibility region V^*; and 3) $V^* \subset \mathcal{Q}$, indicating that a portion of the European continent is not in the field–of–view of at least $2\gamma + 1$ stations. Under these hypotheses, the non–null sets

of the partition described in Section 4.3 are given by

$$
\begin{aligned}
Z_1 &= I_1(0) \cap \cdots \cap I_n(0) \cap V_1(0) \cap \cdots \cap V_n(0) = I^* \cap V^* = I^* \,, \\
Z_2 &= \mathcal{C}(I_1(0)) \cap \cdots \cap \mathcal{C}(I_n(0)) \cap V_1(0) \cap \cdots \cap V_n(0) = \mathcal{C}(I^*) \cap V^* \,, \\
Z_3 &= \mathcal{C}(I_1(0)) \cap \cdots \cap \mathcal{C}(I_n(0)) \cap \mathcal{C}(V_1(0)) \cap \cdots \cap \mathcal{C}(V_n(0)) = \\
&= \mathcal{C}(I^*) \cap \mathcal{C}(V^*) = \mathcal{C}(V^*) \,.
\end{aligned}
$$

In the above equations the property $I^* \subseteq V^*$, which can be evicted by simple reasoning on the definitions of I^* and V^*, has been used. It is possible to provide physical interpretations for each partition set: Z_1 represents the emerged lands that can be detected by using the information available from all stations, Z_2 represents undetected lands in the global visibility region V^*, and Z_3 represents the region not included in the field–of–view of at least $2\gamma + 1$ stations. The original SVBDS can be simulated by a binary dynamic system, where the i–th update rule is

$$
\begin{cases}
i_{i,j}(t+1) = \sum_{H \in S_{\gamma+1}(C_i)} \Pi_{h \in H} \left(v_{h,j}(t)\, i_{h,j}(t) \right) \,, \\
v_{i,j}(t+1) = \sum_{H \in S_{\gamma+1}(C_i)} \Pi_{h \in H} \, v_{h,j}(t) \,,
\end{cases}
\tag{4.19}
$$

for $j = 1, 2, 3$, and the i–th initial state is

$$
x_i(0) = (i_{i,1}(0), \cdots, i_{i,3}(0), v_{i,1}(0), \cdots, v_{i,3}(0)) = (1, 0, 0, 1, 1, 0) \,.
$$

By direct computation it can be shown that the discrete derivative of the map in Eq. 4.19, computed at the equilibrium point $x(0) = (x_1(0)^T, \cdots, x_n(0)^T)^T$, is null, and thus also its spectral radius is null. By Theorem 4.5, the equilibrium point $x(0)$ is attractive in its immediate neighborhood. Based on Theorem 4.6, we can conclude that X^* is attractive for the original SVBDS at least in its complemented neighborhood. Furthermore, consider the region Γ composed

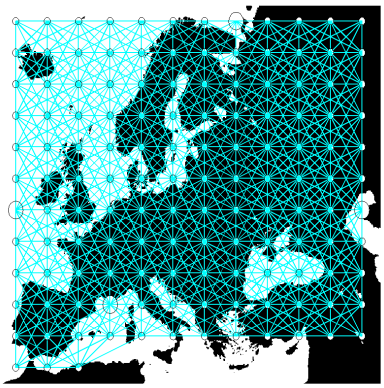


Fig. 4.2 Deployment and connectivity of 135 stations over the European continent (from top to down, $\mathcal{A}_8$, $\mathcal{A}_{73}$, $\mathcal{A}_{84}$, and $\mathcal{A}_{112}$ are represented with bigger circles).

of the states that differ in at most γ components from $x(0)$. It is easy to verify that the value of the map in Eq. 4.19 remains constant for all states in Γ, i.e., for all $\tilde{x} \in \Gamma$, it holds $f(\tilde{x}) = f(x(0)) = x(0)$. This fact tells us that Γ is included in the region of attractiveness of $x(0)$. By projecting back Γ to the original system domain, we can prove that X^* is attractive at least in the set $\mathcal{L}^{-1}(\Gamma)$, which is large enough to tolerate γ incorrect land detections.

A more general case can be considered, where, due to noise increasing with the distance and to local atmospheric conditions, such as the presence of stratus clouds, each $\mathcal{A}_i$ can incorrectly detect points within its field–of–view with probability ε. Each region $V_i(0)$ can thus be interpreted as the initial *region of ε–confidence* of $\mathcal{A}_i$, i.e., the set of points where the probability of land detection error is not greater than ε. Furthermore, we require, for every point in $\mathcal{Q}$, that the probability E of land detection errors in the global chart is bounded as $E \leq \bar{E} < 1$.

For each $\mathcal{A}_i$, the probability of having more than γ land detection errors in $V_i(1)$ is

$$p(\varepsilon) = 1 - \sum_{k=0}^{\gamma} \binom{r}{k} \varepsilon^k (1-\varepsilon)^{r-k} .$$

With the same reasoning, the probability of having more than γ land detection errors in $V_i(2)$ is $p \circ p(\varepsilon)$, and in $V_i(t)$ is $s_\varepsilon(t) = p \circ \cdots \circ p(\varepsilon)$, i.e. the composition of p with itself t times. Therefore, we need to chose a set of sensors with the probability ε satisfying the constraint

$$s_\varepsilon(t) < \bar{E} \quad \text{for all } t \geq 1 . \tag{4.20}$$

It is possible to show that, for $\varepsilon < \frac{1}{2}$ and for all γ, $p(\varepsilon) < \varepsilon$ and the function $s_\varepsilon(t)$ monotonically decreases for $t \geq 1$. Hence, for an admissible error probability $\bar{E} < \frac{1}{2}$, the condition in Eq. 4.20 is implied by $\varepsilon < \bar{E}$.

Let us finally consider a simulative example including $n = 135$ stations with the hypothesis of $\gamma = 3$, $r = 7$, $\varepsilon = 0.05$. By placing the stations on a grid with mesh size δ, we satisfy the measurement multiplicity constraint with $r = \left\lfloor \frac{\pi d(d-1)}{\delta} \right\rfloor + 1$, as known from number theory [69]. Fig. 4.2 is a depiction of the stations' deployment and the available communication graph. The diameter of the communication graph, i.e. the maximum distance between any two nodes on the graph, is 11. Let us consider a case in which $E = 0.02$. Fig. 4.3 shows how the network of stations iteratively update their estimated charts I_i by running an instance of the consensus algorithm described in Eq. 4.18. The first row reports the initial estimated charts of 4 stations obtained from processing of the images taken by their onboard vision systems, while the last row reveals that the stations have successfully converged to the centralized estimated chart I^*.

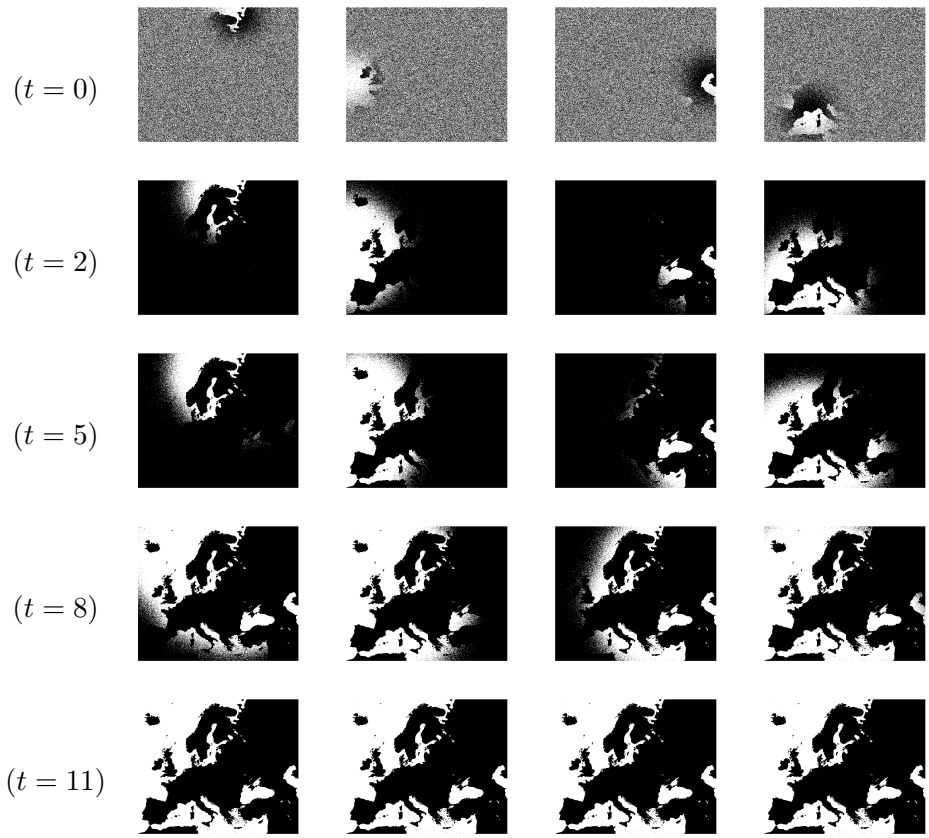


Fig. 4.3 Simulation run with 135 balloon stations and maximum number of faults per pixel given by $\gamma = 3$. Only the evolutions of the charts estimated by 4 stations is reported for space reasons: from left to right, $\mathcal{A}_8$ is placed approximately at latitude $66°$N and longitude $41°$E, $\mathcal{A}_{73}$ at $48°$N and $8°$W, $\mathcal{A}_{84}$ at $45°$N and $32°$E, and $\mathcal{A}_{112}$ at $39°$N and $6°$E. The network of stations effectively consents on a global map of the European continent with reduced noise.

4.7 Application to distributed clock synchronization

The proliferation of robotic devices and the growing number of their potential applications has recently led to an increased interest towards multiple–robot applications, such as consensus, rendezvous, sensor coverage, and simultaneous localization and mapping. All these applications demand for the availability of a global clock, i.e. every node in the network must be able to refer to a unique time. The reason for this necessity is twofold. First, only if a very accurately synchronized clock is available, every node can get a global picture of an event that is sensed by several nodes. Secondly, clock synchronization is essential for reducing node's energy consumption due to communication. Indeed, any two node willing to exchange a message with each other establish a rendezvous. The better the clock synchronization, the less energy is wasted in the necessary guard times to not miss the rendezvous point.

In a centralized system the solution of this problem is trivial: the centralized server will just decide the system's time. In a distributed system, the problem takes on more complexity because a global time is not easily known. In a WSN, clock synchronization poses two major problems. The former is the connectivity, which is related to the fact that nodes of a sensor network cannot directly communicate with each other, and some information may need to be relayed by other nodes. Therefore, it is not possible to choose a reference node to which all other nodes can be synchronized to. The latter problem is related to unpredictable random delays that may normally occur between any pair of nodes. It is indeed known that delivery time of radio messages in WSN is subject to interferences, and node failures which may cause unknown variations to the standard communication time.

In this respect, during the last years clock synchronization in distributed systems has been extensively studied. As a result of this effort, many different approaches have been proposed (see e.g. [70, 71, 72, 73, 74, 75]). Specifically

developed for WSN are the Reference Broadcast Synchronization (RBS) [76], Timing–sync Protocol for Sensor Networks (TPSN) [77] and the Precision Time Protocol (PTP) [78]). More in detail, the RBS exploits the broadcast nature of the physical channel to synchronize a set of receivers with one another. The timestamp of the reception of a broadcast message is recorded at each node and these timestamps are exchanged to calculate relative clock offsets between nodes. The TPSN algorithm builds a spanning tree of the network during the level discovery phase. In the synchronization phase of the algorithm, nodes synchronize to their parent in the tree by a two-way message exchange. PTP is a time-transfer protocol defined in the IEEE 1588 standard that allows precise synchronization of networks (e.g., Ethernet).

More recently, so–called Average TimeSync protocol [79] has been proposed as a different approach to clock synchronization. The main idea underlying this approach is to average local information to achieve a global agreement on a specific quantity of interest, and this is obtained by transposing the synchronization problem into a linear consensus problem. Notwithstanding, the problem of clock synchronization in a WSN is far from been completely solved. As a matter of fact, available solutions, such as NTP, require operating conditions that are not guaranteed in a WSN. These difficulties are due to limited energy and bandwidth availability, that are in turn necessary to allow sensors a longer operating life. Furthermore, the fact that the topology is dynamically changing is another issue that makes the clock synchronization problem in a WSN more difficult than in traditional network, and actually a very challenging one.

In what follows it is proved that the clock synchronization problem may be transposed into a consensus problem on sets. This is achieved by exploiting the idea that uncertain measures of a clock that have been propagated through the network can be represented as interval. This idea was first proposed by Marzullo in [62] that showed a centralized algorithm to determine the smallest confidence interval that is contained in the largest number of sensor mea-

sures. The solution therein proposed now forms the basis of the Network Time Protocol (NTP) [63], a protocol that is implemented in many software platforms and operating systems. NTP sets and maintains the system time of day in synchronism with Internet standard time servers. NTP does most computations in 64-bit floating–point arithmetic and does relatively clumsy 64–bit fixed–point operations only when necessary to preserve the ultimate precision, about 2.32×10^{-10} seconds (232 picoseconds). While the ultimate precision is not achievable with ordinary workstations and networks of today, it may be required with future gigahertz CPU clocks and gigabit LANs. However, Marzullo's solution is actually centralized and is extended to a distributed approach in this thesis.

4.7.1 A Centralized Extension of Marzullo's Algorithm

Consider the clock synchronization algorithm originally proposed by Marzullo in [62]. In this section, we propose an extension of the algorithm and we show how to convert it into a problem involving only operations on sets. Suppose to have n sensors that are able to measure the value of quantity of interest within a confidence interval or set. Let us denote with $u \in U$ this quantity that may range from time, temperature, to the position of an object during a SLAM application, or to the configuration of a neighboring car (as e.g. the Highway example in Chapter 3).

Suppose that one or more of these sensors may fail and produce a confidence interval or set that is not consistent with the others. Then, given these n sets, $U_1, \ldots, U_n$, we want to compute the smallest set Y that is contained in the largest number of such sensor measures. This quantity represents the set that is most likely to contain the real clock value. A solution to this problem can be found that uses only operations on sets (union $\cup$, intersection $\cap$, and complementation $\mathcal{C}(\cdot)$).

Given n agents' indices $1, 2, \ldots, n$, consider the number of combinations of i of such indices out of n being specified by

$$c(n,i) \stackrel{\text{def}}{=} \left(\begin{array}{c} n \\ \\ i \end{array} \right) .$$

Then, consider the sets

$$A_i = \bigcup_{l=1}^{c(n,n-i+1)} \bigcap_{h=1}^{n-i+1} U_{i_{l,h}} ,$$

for $i = 1, 2, \ldots, n$, where $i_{l,1}, \ldots, i_{l,n-i+1}$ are distinct combinations of agents' indices. More explicitly, the sets are given by

- $A_1 = U_1 \cap U_2 \cap \cdots \cap U_n$ (the intersection of all n sensor measures),
- $A_2 = (U_1 \cap U_2 \cap \cdots \cap U_{n-1}) \cup (U_1 \cap U_2 \cap \cdots \cap U_{n-2} \cap \cap U_n) \cup \ldots$ (the union of the n possible intersections of $n-1$ sensor measures),
- $\cdots$,
- $A_n = U_1 \cup U_2 \cup \ldots \cup U_n$ (the union of the n individual sensor measures).

Consider also the following filtering sets:

$$\begin{aligned} \Gamma_1 &= U , \\ \Gamma_i &= \Gamma_{i-1} \cap \begin{cases} U \text{ if } A_{i-1} = \emptyset , \\ \emptyset \text{ if } A_{i-1} \neq \emptyset , \end{cases} \quad \text{for } i = 2, ..., n . \end{aligned} \tag{4.21}$$

Then, the desired confidence set Y is readily given by

$$Y = \bigcup_{i=1}^{n} (\Gamma_i \cap A_i) \stackrel{\text{def}}{=} \mathsf{M}(U_1, \ldots, U_n) . \tag{4.22}$$

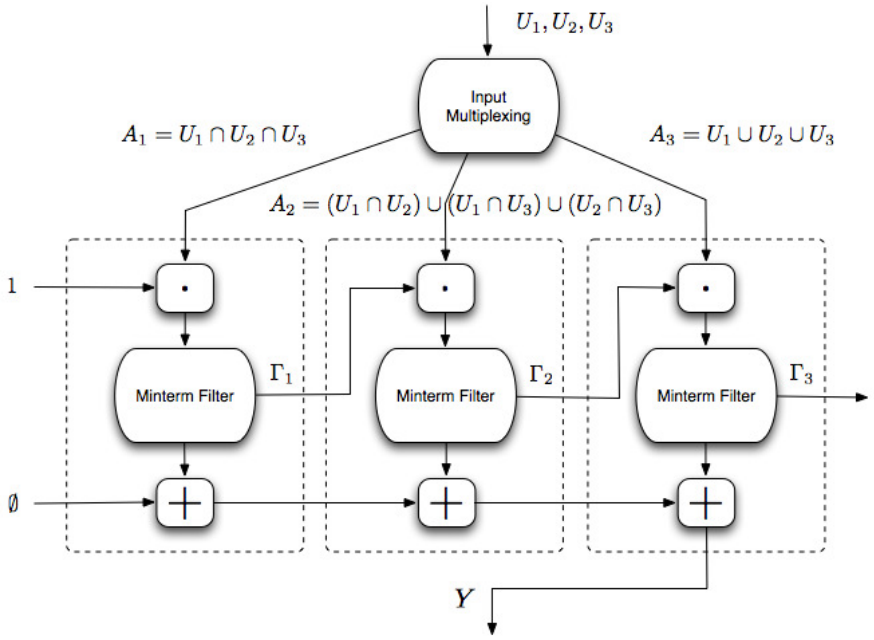


Fig. 4.4 An instance of the algorithm to solve the set–valued formulation of Marzullo's problem with 3 input sets.

We can give the following

Definition 4.19. Given n set measures $U_1, \ldots, U_n$, we say that U_j is *consistent* if it shares an intersection with the smallest set that is common with the maximum number of the other sets, i.e.

$$U_j \cap \mathsf{M}(U_1, \ldots, U_n) \neq \emptyset .$$

Conversely, we say that U_j is *inconsistent*.

A pictorial representation of the algorithm is reported in Fig. 4.4. The figure outlines the modular structure of the algorithm, where the output of a module is the input of the following one.

Once the best estimated set Y has been computed from the initial collection $X_1, \ldots, X_m$, one has typically to extract a scalar value $b \in \mathcal{X}$ to be used in a control loop. For time synchronization, the complete set is $\mathcal{X} = [0, \infty)$, and a common choice is to take the earliest interval $[t_{min}, t_{max}] \subseteq Y$ and extract its middle value

$$b = \frac{t_{max} - t_{min}}{2} .$$

4.7.2 Example

Suppose to have the interval $U = [0, \infty)$, and $m = 3$ sensors providing the following confidence interval: $U_1 = [1, 10]$, $U_2 = [30, 40]$, and $U_3 = [6, 29]$. In this case, map ϕ computes the following 3 intervals:

$$A_1 = U_1 \cap U_2 \cap U_3 = \emptyset ,$$

$$A_2 = (U_1 \cap U_2) \cup (U_1 \cap U_3) \cup (U_2 \cap U_3) = [6, 10] ,$$

$$A_3 = U_1 \cup U_2 \cup U_3 = [1, 29] \cup [30, 40] .$$

The filtering sets are the intervals $\Gamma_1 = U$, $\Gamma_2 = U$, and $\Gamma_3 = \emptyset$. Thus, the smallest interval that is in common to the largest number of the given intervals is

$$\begin{aligned} Y &= (\Gamma_1 \cap A_1) \cup (\Gamma_2 \cap A_2) \cup (\Gamma_3 \cap A_3) = \\ &= ([0, \infty) \cap \emptyset) \cup ([0, \infty) \cap [6, 10]) \cup \\ &\quad \cup (\emptyset \cap ([1, 29] \cup [30, 40])) = \\ &= [6, 10] , \end{aligned}$$

which is contained in U_1 and U_2 (see Fig. 4.7.2). On the contrary, the third sensor's measure is faulty. Indeed, we have:

$$Y \cap U_3 = \emptyset \,.$$

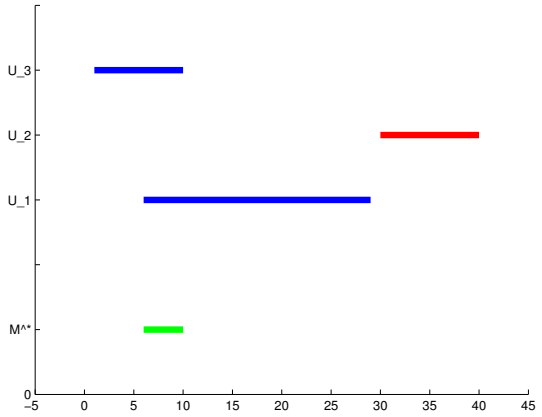
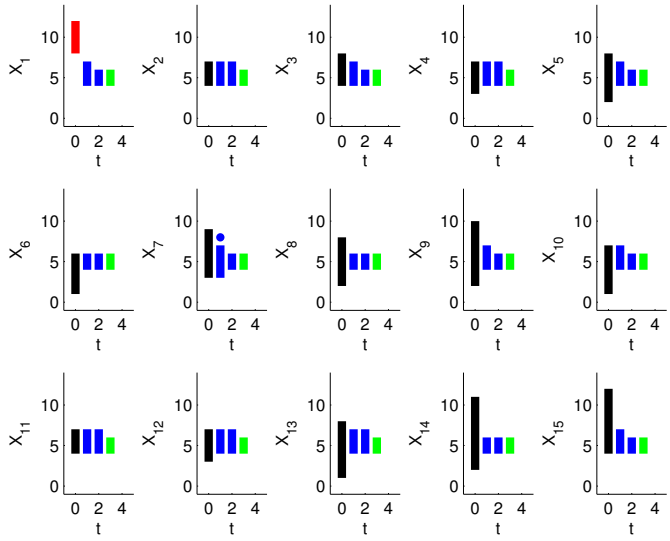


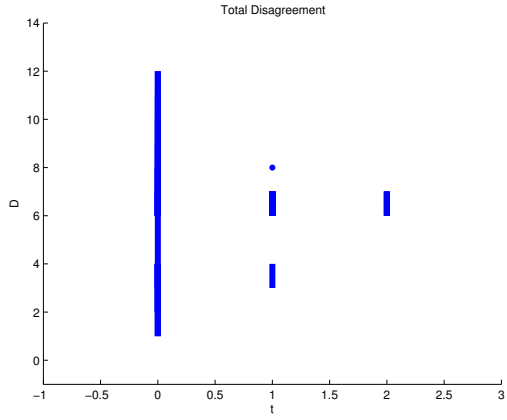
Fig. 4.5 Example of clock interval estimated by the centralized solution of Marzullo's algorithm. The estimated set is in green, the correct measures U_1 and U_2 are in blue, and the inconsistent measure U_3 is in red.

4.7.3 Distributed Clock Synchronization

The algorithm presented in Section 4.7.1 is able to solve the extended Marzullo's problem only in a centralized setting, whereas we want that every agent have a consistent information on the quantity of interest u, so that any of them can be polled by an external user. To this aim, let us suppose that every generic agent $\mathcal{A}_i$ has a state $X_i \subseteq U$ that represents its estimate of the quantity of interest u and is able to share the value of its state with all its neighbors by exchanging a message with them. Suppose that every agent initializes its state with the value of its local estimate of the quantity of interest, i.e.



(a)



(b)

Fig. 4.6 Simulation run of a dynamic systems estimating Marzullo's solution with $\gamma = 1$ (a), possible inconsistencies, and corresponding behavior of the network disagreement w.r.t. Marzullo's centralized decision (b)

$$X_i(0) \leftarrow U_i(0)\,.$$

Then, we want to design a distributed iteration rule of the form

$$X(t+1) = F(X(t))\,,$$

where $X = (X_1, \ldots, X_n)^T$ is the system's state, and t is a discrete time, such that, starting from any initial state $X(0)$, every agent will consent on Marzullo's centralized decision, i.e. there exists a finite time $\bar{t}$ such that

$$X(\bar{t}) = \mathsf{M}(U(0)) = \mathsf{M}(X(0))\,.$$

Furthermore, due to the result stated in [80] and concerning the *impossibility to reach a consensus* with corrupted data, we must add a further hypothesis to the problem guaranteeing that the maximum number of inconsistent measures is at most γ. We will refer to this as the *bounded inconsistency hypothesis*.

From [81], recall the following

Definition 4.20. A graph $G = (V, E)$ is said to be *k–connected* if there does not exist a set of $k-1$ vertices in V whose removal disconnects the graph, i.e. the vertex connectivity of G is greater or equal to k.

Therefore, a connected graph is 1–connected, and a biconnected graph is 2–connected.

First of all, note that the hypothesis of bounded inconsistency implies that there exists at least one combination of $n-\gamma$ sets that share an intersection with Marzullo's centralized decision, i.e.

$$\exists\, i_1, \cdots, i_{n-\gamma} \in \{1, \ldots, n\} \,|\, X_{i_j} \cap M^* \neq \emptyset\,, \tag{4.23}$$

where $M^* = \mathsf{M}(X_1(0), \ldots, X_n(0))$. In case of virtuous scenario with only consistent measures ($\gamma = 0$), this condition implies that

$$X_1(0) \cap \cdots \cap X_n(0) \neq \emptyset \,,$$

and a solution to the problem can be obtained by simply replicating the M-Algorithm on every node according to its communication neighbors, i.e.

$$F_i(X) = \mathsf{M}\left(X_{i_1}, \cdots, X_{i_{n_i}}\right), \text{for } i = 1, \ldots, n \,,$$

where n_i is any number of agent $\mathcal{A}_i$'s neighbors, and $i_1, \cdots, i_{n_i}$ are their indices. Indeed, in the virtuous hypothesis, the algorithm on agent $\mathcal{A}_i$ reduces to a pure intersection of the data received from its communication neighbors. As it is well-known, set intersection is associative, commutative, and idempotent $(X \cap X = X)$. Therefore, given that the underlying communication graph is connected, the network convergence toward the centralized decision X^* is guaranteed ([82]). Thus its distributed application allows the agents to consent on the value of the centralized intersection, which is also the desired solution of the extended Marzullo's problem. Consider the case with $\gamma > 0$. Suppose that all initial measures are represented by *compact* sets so that, if a common intersection exists, the intersection itself is a compact set. Due to this, all points in the Marzullo's centralized decision are in common with the very same initial sets. In this case, the analysis of convergence is more complex. Anyway, we are able to prove the following

Theorem 4.8. *Consider a network of agents evolving according to the iterative rule*

$$X_i(t+1) = F_i(X) = \mathsf{M}(X_{i_1}(t), \cdots, X_{i_{n_i}}(t)) \,,$$

$$X_i(0) = U_i \,,$$

for $i = 1, \ldots, n$, where at most γ initial states may be inconsistent. The network can reach a consensus on the centralized Marzullo decision

$$X^* = (M^*, \cdots, M^*)^T ,$$

$$M^* \stackrel{def}{=} \mathsf{M}(X_1(0), \ldots, X_n(0)) ,$$

if the agents can exchange messages according to a communication matrix C that is at least r–connected, with $r = 2\,\gamma + 1$. The consensus is achieved in at most n steps.

Proof. We have to prove that if $r = 2\,\gamma+1$, the distributed algorithm converges to the centralized version of the Marzullo Algorithm (M^*). First observe that, under bounded inconsistency, the algorithm of Marzullo's centralized decision returns the same value of the algorithm itself truncated at the $\gamma + 1$–th term of the union in Eq. 4.22. This happens because of Eq. 4.23 which implies that $\Gamma_i = \emptyset$, for $i \geq \gamma + 2$. Thus, the algorithm itself reduces to

$$Y = \bigcup_{i=1}^{\gamma+1} (\Gamma_i \cap A_i) \ . \tag{4.24}$$

Moreover, note that, by construction, it holds the general property

$$A_i \subseteq A_{i+1} , \quad \text{for } i = 1, \ldots, n-1.$$

which implies that an algorithm computing only the term A_i returns a value that is upper bounded by an algorithm computing only the term A_{i+1}. We replicate the Algorithm on every node according to its communication neighbors,, i.e.

$$F_i(X) = \mathsf{M}\,(X_{i_1}, \cdots, X_{i_r}) \,, \text{for } i = 1, \ldots, n \,,$$

Consider the worst case, i.e. a node is connected with all the faulty nodes. We define $c_i = \{a, .., f\}$ and $nc_i = \{h, ..., v\}$ with $\#c_i = \gamma + 1$ and $\#nc_i = \gamma$ the sets available at the node i, containing the indices of the consistent measure and the indices of the inconsistent measures respectively. The bounded inconsistency hypothesis guarantees that

$$X_k \cap M^* \neq \emptyset \,, \forall k \in c_i.$$

As X_k and X_j, $k, j \in c_i$ are two consistent measures, then it must hold $M^* \subseteq X_k, X_j$, which also implies that

$$X_k \cap X_j \neq \emptyset \quad \forall k, j \in c_i.$$

In other words we have

$$I_{c_i} = \bigcap_s X_s \neq 0 \,, \quad s \in c_i, \tag{4.25}$$

$$I_{nc_i} = \bigcap_s X_p \quad s \in nc_i, \tag{4.26}$$

being I_{c_i} the intersection between consistent measure, and I_{nc_i} the intersection between inconsistent measure. Note that while I_{c_i} is always different from the empty-set, I_{nc_i} can be an emptyset or not.

Moreover, since I_{nc_i} are inconsistent values, it is easy to verify that

$$I_{nc_i} \cap I_{c_i} = \emptyset$$

$$\bigcup_p (I_{c_i} \cap X_p) = \emptyset \quad p \in nc_i$$

Recalling the update rule in (4.21) we now have

$$M\left(X_{i_1}, \cdots, X_{i_r}\right) = (\bigcap_{s_i} X_{s_i}) = I_{c_i}$$

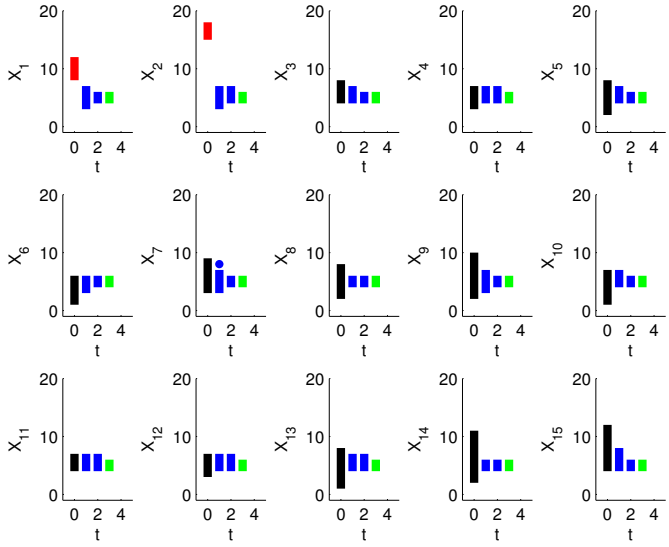
where $s_i \in c_i$. This is because the M-Algorithm computes the smallest confidence set that is contained in the largest number of such sensor measures, and we have that $\#c_i > \#nc_i$. This guarantees that the inconsistent measures do not affect the state X, as it also happens for the centralized execution of M-algorithm (see Section 4.7.1). Under bounded inconsistency hypothesis,

execution of the algorithm at the generic agent $\mathcal{A}_i$ reduces to the intersection of any data received by its neighbors. Furthermore, the update rule in (4.21) is associative, distributive, and idempotent w.r.t. its input arguments. Since the the communication graph is connected, according to [82], the convergence of the dynamic system toward the centralized decision is guaranteed in at most n steps and it proves the thesis.

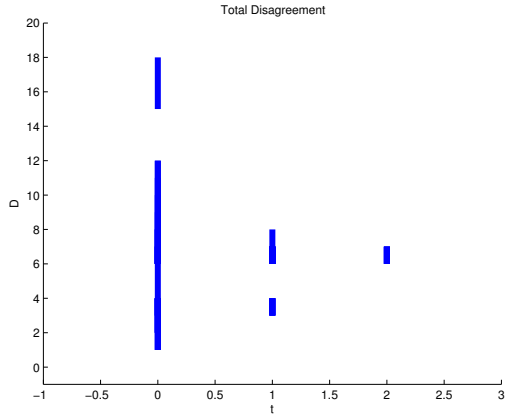
4.7.4 Simulation

The effectiveness of the proposed solution is shown through numerical simulations with $n = 15$ agents, where 2 different scenarios with $\gamma = 1, 2$ possible inconsistent measures are considered. Agents are able to communicate according to a graph, that is not reported for the sake of space, but that is chosen so as to guarantee the minimum redundancy required. Fig. 4.6a reports the behavior of the dynamic system starting from an initial condition where only one agent, agent $\mathcal{A}_1$, has an inconsistent measure. In this case, the required number of neighbors in the communication graph is $r = 2\,\gamma + 1 = 3$. The figure shows that the inconsistent measure is tolerated, and that every agent reach the correct final clock interval. Fig. 4.7a refers to a simulation of a dynamic system that has been design under the hypothesis of $\gamma = 2$ possible inconsistencies. In this case, $r = 5$ neighbors in the communication graph are required. The figure shows that the inconsistent measures of agents $\mathcal{A}_1$ and $\mathcal{A}_2$ are tolerated. In both figures, the final decision coinciding with the Marzullo's centralized estimated interval is reported in green. The inconsistent measures are instead colored in red. Finally, both figures reports the behavior of network's *relative disagreement*

$$E \stackrel{\text{def}}{=} \sum_{i=1}^{n} \mathcal{D}(X_i, \mathsf{M}\,(X(0))\,,$$



(a)



(b)

Fig. 4.7 Simulation run of a dynamic systems estimating Marzullo's solution with $\gamma = 2$ (a), possible inconsistencies, and corresponding behavior of the network disagreement w.r.t. Marzullo's centralized decision (b)

where $\mathcal{D}$ is the vector distance, based on the symmetric difference between two intervals. The figure shows that the disagreement becomes $\emptyset$ in a number of steps less or equal to the diameter of the chosen communication graph. Recall that the diameter of a graph is the maximum distance between any two nodes in the graph, and the distance is the length of the minimum path connecting the nodes.

As we have stated, the proposed solution is valid also when the data that is exchanges is a set. To show this, we conclude with a further example with $U = [0, \infty) \times [0, \infty)$, and $m = 4$ sensors providing the following measured confidence sets:

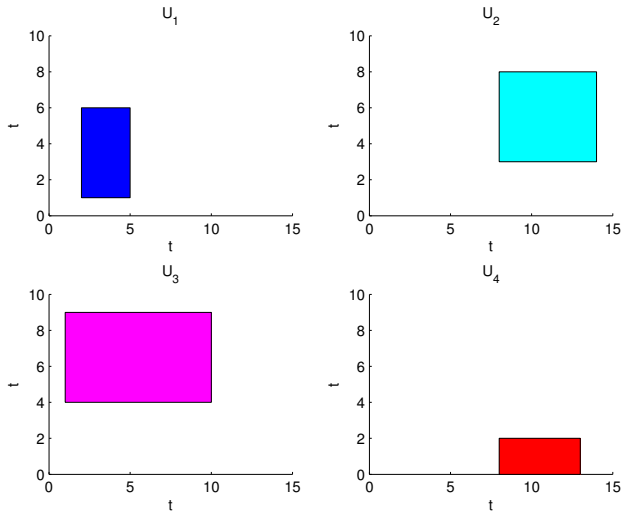
$$U_1 = [2, 5] \times [1, 6]\,,$$

$$U_2 = [8, 14] \times [3, 8]\,,$$

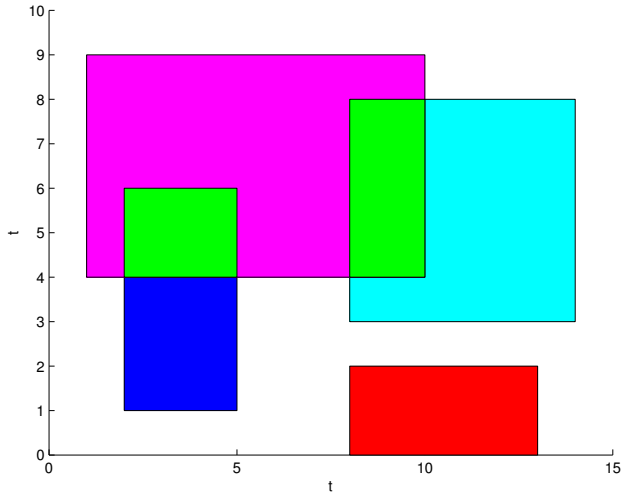
$$U_3 = [1, 10] \times [4, 9]\,,$$

$$U_4 = [8, 13] \times [0, 2]\,.$$

Before showing the simulation results, let us first compute the centralized solution. As described in Section 4.7.1, this involves computation of the following sets



(a)



(b)

Fig. 4.8 Simulation run with $n = 4$ sensors that have four uncertain measures of a quantity of interest on the plane (a). The final result show that the inconsistent set (in red) has been excluded and tolerated (b).

$$
\begin{aligned}
A_1 &= U_1 \cap U_2 \cap U_3 \cap U_4 = \emptyset\,, \\
A_2 &= (U_1 \cap U_2 \cap U_3) \cup (U_1 \cap U_2 \cap U_4) \cup \\
&\quad \cup (U_2 \cap U_3 \cap U_4) = \emptyset\,, \\
A_3 &= (U_1 \cap U_2) \cup (U_1 \cap U_3) \cup (U_1 \cap U_4) \cup \\
&\quad \cup (U_2 \cap U_3) \cup (U_2 \cap U_4) \cup (U_3 \cap U_4) = \\
&\quad = \emptyset \cup ([2,5] \times [4,6]) \cup \emptyset \cup \\
&\quad \cup ([8,10] \times [4,8]) \cup \emptyset \cup \emptyset = \\
&\quad = ([2,5] \times [4,6]) \cup ([8,10] \times [4,8])\,, \\
A_4 &= U_1 \cup U_2 \cup U_3 \cup U_4 = \\
&\quad = ([2,5] \times [1,6]) \cup ([8,14] \times [3,8]) \cup \\
&\quad \cup ([1,10] \times [4,9]) \cup ([8,13] \times [0,2]).
\end{aligned}
$$

The filtering sets are $\Gamma_1 = U$, $\Gamma_2 = U$, $\Gamma_3 = U$, and $\Gamma_4 = \emptyset$. Thus the smallest set that is in common to the largest number of the given sets is

$$
\begin{aligned}
Y &= (\Gamma_1 \cap A_1) \cup (\Gamma_2 \cap A_2) \cup (\Gamma_3 \cap A_3) \cup (\Gamma_4 \cap A_4) = \\
&= ([0,\infty) \cap \emptyset) \cup ([0,\infty) \cap \emptyset) \cup \\
&\cup ([0,\infty) \cap A_3) \cup (\emptyset \cap A_4) = \\
&= ([2,5] \times [4,6]) \cup ([8,10] \times [4,8])\,,
\end{aligned}
$$

which is partially contained in U_1 and U_3, and partially contained in U_2 and U_3. On the contrary, the forth sensor's measure is faulty. Indeed, we have:

$$
Y \cap U_4 = \emptyset\,.
$$

Fig. 4.8 refers to a simulation of the dynamic system that has been design under the hypothesis of one possible faulty node. Fig 4.8a shows the initial status $X(0)$ of the agents, while Fig. 4.8b reports the final decision coinciding with the Marzullo's centralized estimated interval is reported in green. The inconsistent measure is instead colored in red. Simulation show that the centralized estimate is achieved also by distributed execution of the algorithm.

Chapter 5

Distributed Synthesis of Robust Logical Consensus Systems

The notion of Boolean consensus systems would represent a unifying framework for achieving consensus on Boolean information (not only including binary data) . In fact, what really prevents, in my opinion, a wide exploitation of the multi–agent paradigm is the lack of a systematic approach to the design of a generic consensus algorithm that is applicable in a vast number of scenarios. This is well–known to computer scientists that have studied consensus on generic data and provided efficient solutions that can tolerate even the presence of misbehaving or simply faulty agents (see e.g. Lynch's book [83]).

In this vein, at a suitable abstraction level, every multi–agent system requires that agents consent on a centralized logical decision e.g. one can imagine that the behaviors of agents described according to the protocol defined in Chapter 3 depends on a set of input events on which agents have partial visibility and therefore have to be estimated by the agents themselves. In [60], so–called logical consensus approach was proposed, by which a network of agents that are able to exchange binary values representing their local estimates of the events, is able to reach a unique and consistent decision. The approach is based on the construction of an iterative map, whose computation is centralized and guaranteed under suitable conditions on the input visibility and graph connectivity. Under the same conditions, this works presents a procedure allowing the construction of a logical linear consensus system in a

fully distributed way. Such a distributed synthesis is instrumental in mobile networked robots, where the presence of a centralized supervisor is impossible or the hypothesis of a fixed communication topology is unrealistic. The solution consists of so–called Self Routing Network Protocol (SRNP) allowing a set of (possibly mobile) robots randomly deployed, to self configure their communication network in such a way that a unique and consistent decision can be reached.

The limitation of the above solution is that the obtained map only converges to the correct value if every agent process and send correct information. Therefore, to cope with possible faults of some agents, eventually leading the system to incorrect or disconnected decisions, this works presents a fully distributed synthesis procedure that generates a logical nonlinear consensus map that is able to tolerate misbehaving agents that may send incorrect information, due to spontaneous failure or even tampering. The procedure is formalized as a distributed protocol, called SR^{2}NP, which is guaranteed to converge under similar visibility and communication conditions. The protocol is based on the known result of [30] ensuring that redundant minimum–length paths from a generic input to every agent of the network can be found if the number of faults is bounded. The effectiveness of the proposed methods is showed through the real implementation of a wireless sensor network, that uses low–cost communication and computation devices, as a framework for the surveillance of an urban area. The results reported in this chapter can be found in [A4, A6]

5.1 Problem Formulation

In the following, this thesis considers the application scenarios requiring the computation of a set of p *decisions*, $y_1, \ldots, y_p$, that depend on m logical *events*, $u_1, \ldots, u_m$. Such events may represent e.g. the presence of an intruder or of a fire within an indoor environment. More precisely, for any given com-

bination of input events, this work considers a *decision task* that requires computation of the following system of logical functions:

$$\begin{cases} y_1 = f_1(u_1, \ldots, u_m)\,, \\ \ldots \\ y_p = f_p(u_1, \ldots, u_m)\,, \end{cases} \tag{5.1}$$

where each $f_i : \mathbb{B}^m \to \mathbb{B}$ consists of a logical condition on the inputs. Let us denote with $u = (u_1, \ldots, u_m)^T \in \mathbb{B}^m$ the input event vector, and with $y = (y_1, \ldots, y_p)^T \in \mathbb{B}^p$ the output decision vector. Then, it is possible to write $y = f(u)$ as a compact form of Eq. 5.1, where $f = (f_1, \ldots, f_p)^T$, with $f : \mathbb{B}^m \to \mathbb{B}^p$, is a logical vector function. It is worth noting that computation of f is *centralized* in the sense that it may require knowledge of the entire input vector u to determine the output vector y.

The approach to solve the decision task consists of employing a collection of n agents, $\mathcal{A}_1, \ldots, \mathcal{A}_n$, that are supposed to cooperate and possibly exchange locally available information. It is required that all agents reach an agreement on the centralized decision $y = f(u)$, so that any agent can be *polled* and provide consistent complete information. In this perspective, we pose the problem of reaching a *consensus on logical values*. We assume that each agent is described by a triple $\mathcal{A}_i = (\mathcal{S}_i, \mathcal{P}_i, \mathcal{C}_i)$, where $\mathcal{S}_i$ is a collection of sensors, $\mathcal{P}_i$ is a processor that is able to perform elementary logical operations such as $\{\texttt{and}, \texttt{or}, \texttt{not}\}$, and $\mathcal{C}_i$ is a collection of communication devices allowing transmission of only sequences of binary digits, 0 and 1, namely strings of bits. Although we assume that every agent has the same processing capability, i.e. $\mathcal{P}_i = \mathcal{P}$ for all i, we consider situations where agents may be *heterogeneous* in terms of sensors and communication devices. Due to this diversity as well as the fact that agents are placed at different locations, a generic agent i may or may not be able to measure a given input event u_j, for $j \in 1, \ldots, m$. There-

fore, one can conveniently introduce a *visibility matrix* $V \in \mathbb{B}^{n\times m}$ such that we have $V_{i,j} = 1$ if, and only if, agent $\mathcal{A}_i$ is able to measure input event u_j, or, in other words, if the i–th agent is directly *reachable* from the j–th input. Moreover, for similar reasons of diversity and for reducing battery consumption, each agent is able to communicate only with a subset of other agents. This fact is captured by introducing a *communication matrix* $C \in \mathbb{B}^{n\times n}$, where $C_{i,k} = 1$ if, and only if, agent $\mathcal{A}_i$ is able to receive a data from agent $\mathcal{A}_k$. Hence, agents specified by row $C_{i,:}$ will be referred to as C–neighbors of the i–th agent.

In this view, we can imagine that each agent $\mathcal{A}_i$ has a local *state* vector, $X_i = (X_{i,1}, \ldots, X_{i,q}) \in \mathbb{B}^q$, that is a *string of bits*. Denote with $X(t) = (X_1^T(t), \ldots, X_n^T(t))^T \in \mathbb{B}^{n\times q}$ a matrix representing the network state at a discrete time t. Hence, we assume that each agent $\mathcal{A}_i$ is a *dynamic node* that updates its local state X_i through a *distributed* logical update function F that depends on its state, on the state of its C–neighbors, and on the reachable inputs, i.e. $X_i(t+1) = F_i(X(t), u(t))$. Moreover, we assume that each agent $\mathcal{A}_i$ is able to produce a logical output decision vector $Y_i = (y_{i,1}, \ldots, y_{i,p}) \in \mathbb{B}^p$ through a suitable distributed logical output function G depending on the local state X_i and on the reachable inputs u, i.e. $Y_i(t) = G_i(X_i(t), u(t))$. Let us denote with $Y(t) = (Y_1^T(t), \ldots, Y_p^T(t))^T \in \mathbb{B}^{p\times q}$ a matrix representing the network output at a discrete time t. Therefore, the network evolution can be modeled as the *distributed finite–state iterative system*

$$\begin{cases} X(t+1) = F(X(t), u(t))\,, \\ \qquad Y(t) = G(X(t), u(t))\,, \end{cases} \tag{5.2}$$

where we have $F = (F_1^T, \ldots, F_n^T)^T$, with $F_i : \mathbb{B}^q \times \mathbb{B}^m \to \mathbb{B}^q$, and $G = (G_1^T, \ldots, G_n^T)^T$, with $G_i : \mathbb{B}^q \times \mathbb{B}^m \to \mathbb{B}^p$.

In a fully decentralized setting, every agent is unaware of all inputs and all other agents' existence, and it only knows the index list $v_i \stackrel{\text{def}}{=} \{v_{i,1}, v_{i,2}, \ldots\} \subseteq \{1, \ldots, m\}$ of the events that it can "see" and the index list $c_i \stackrel{\text{def}}{=} \{c_{i,1}, c_{i,2} \ldots\} \subseteq$

$\{1, \ldots, n\}$ of its neighbors. In this case, the above mentioned centralized visibility and communication matrices, $V = \{V_j(i)\}$ and $C = \{C_{i,j}\}$, can be reconstructed according to the rules

$$V_j(i) = \begin{cases} 0 \text{ if } i \notin v_j \\ 1 \text{ otherwise}, \end{cases} \qquad C_{i,j} = \begin{cases} 0 \text{ if } i \notin c_j \\ 1 \text{ otherwise}. \end{cases}$$

Therefore, we are interested in solving the following

Problem 5.1 (Distributed Synthesis of Consensus Maps). Given a decision system as in Eq. 5.1, the visibility and communication lists, v_i and c_i, for all i, design a distributed logical consensus system of the form in Eq. 5.2, that is distributed, i.e. every agent directly uses only information compatible with its own v_i and c_i, and that converges to the centralized decision $y^* = f(u)$, i.e. $Y(t) = \mathbf{1}_n (y^*)^T$, for all initial network state $X(0)$ and inputs u.

5.2 Centralized Consensus Map Synthesis

In this section, we recall a solution to the centralized version of the synthesis problem from [60], where the visibility and communication matrices are supposed to be available during the consensus design.

First consider vectors $C^k V_j$, for $k = 0, 1, \ldots$, each containing 1 in all entries corresponding to agents that are reachable from input u_j after *exactly* k steps. The i–th element of $C^k V_j$ is 1 if, and only if, there exists a *path* of length k from any agent directly reached by u_j to agent $\mathcal{A}_i$. Recall that, by definition of graph diameter, all agents that are reachable from an initial set of agents are indeed reached in at most $\mathrm{diam}(G)$ steps, with $\mathrm{diam}(G) \leq n - 1$. Let us denote with κ the *visibility diameter* of the pair (C, V_j) being the number of steps after which the sequence $\{C^k V_j\}$ does not reach new agents. Agents that can be reached by the j–input are specified by non–null elements of the Boolean

vector $I_j = \sum_{k=0}^{n-1} C^k V_j$, that contains 1 for all agents for which there exists at least one path originating from an agent that is able to measure u_j. Suppose, for simplicity, that only agent $\mathcal{A}_1$ is able to measure u_j. Then, a straightforward and yet optimal *strategy to allow the information on u_j flowing* through the network is obtained if agent $\mathcal{A}_1$ communicates its measurement to all its C–neighbors, which in turn will communicate it to all their C–neighbors without overlapping, and so on. In this way, we have that every agent $\mathcal{A}_i$ receives u_j from exactly one minimum–length path originating from agent $\mathcal{A}_1$. The vector sequence $\{C^k V_j\}$ can be exploited to this aim. Indeed, it trivially holds that $C^k V_j = C(C^{k-1} V_j)$, meaning that agents reached after k steps have received the input value from agents that were reached after exactly $k-1$ steps. Then, any consecutive sequence of agents that is extracted from non–null elements of vectors in $\{C^k V_j\}$ are (C, V_j)–compliant by construction. A consensus strategy would minimize the number of steps (or *rounds*) to reach an agreement if, and only if, at the k–th step, all agents specified by non–null elements of vector $C^k V_j$ receives the value of u_j from the agents specified by non–null elements of vector $C^{k-1} V_j$. Nevertheless, to minimize also the number of messages, only agents specified by non–null elements of vector $C^k V_j$ and that have not been reached yet must receive u_j. If vector $I_j = \sum_{i=0}^{i=k} C^i V_j$ is iteratively updated during the design phase, then the set of all agents that must receive a message on u_j are specified by non–null elements of vector $C^k V_j \wedge \neg I_j$. By doing this, an optimal pair (C^*, V_j^*) allowing a consensus to be established over the reachable subgraph is obtained. Observe that is $C^* = S\,C \leq C$, where S is a suitable selection matrix.

This procedure actually gives us only a suggestion on how to build a consensus system that solves the centralized version of Problem 5.1. Theorem 5.1, stated below in this section, allows us to say that a simple logical linear consensus algorithm of the form

$$x(t+1) = F_j\, x(t) + B_j\, u_j(t)\,, \tag{5.3}$$

where $F_j = C^*$, $B_j = V_j^*$, and $x \in \mathbb{B}^n$, allows consensus to be reached over the entire reachable subgraph.

In all cases where a unique generic agent $\mathcal{A}_i$ is directly reachable from input u_j, an optimal communication matrix C^* for a linear consensus of the form of Eq. 5.3 can be iteratively found as the incidence matrix of a *input–propagating spanning tree* having $\mathcal{A}_i$ as the root. Then, an optimal pair (C^*, V_j^*) can be written as $C^* = P^T\,(S\,C)\,P$, and $V_j^* = P^T\,V_j$, where S is a selection matrix, and P is a permutation matrix. Furthermore, C^* has the following lower–block triangular form:

$$C^* = \left(\begin{array}{cccc|c} 0 & 0 & \cdots & 0 & 0 \\ \tilde{C}_{i,1} & 0 & \dots & 0 & 0 \\ \vdots & & & \vdots & \vdots \\ 0 & \cdots & \tilde{C}_{i,\kappa_i} & 0 & 0 \\ \hline 0 & \cdots & 0 & 0 & 0 \end{array}\right), \tag{5.4}$$

and $V_j^* = P^T\,V_j = (1, 0, \dots, 0)^T$.

In the general case with ν, $1 \leq \nu \leq n$ agents $A = \{i_1, \dots, i_\nu\}$ that are reachable from input u_j, the optimal strategy for propagating input u_j consists of having each of the other agents receive the input measurement through a path originating from the nearest reachable agent in A. This naturally induces a network partition into ν disjoint subgraphs or spanning trees, each directly reached by the input through a different agent. Let us extract ν independent vectors $V_j(i_1), \dots, V_j(i_\nu)$ from vector V_j having a 1 in position i_h. Then, the sequences $\{C^k V_j(i_h)\}$ are to be considered to compute the optimal partition. Let us denote with κ_i, for all $i \in A$ the number k of steps for the sequence $\{C^k V_j(i)\}$ to become stationary. Therefore, we have that the visibility diameter of the pair (C, V_j) is $\texttt{vis-diam}(C, V_j) = \max_i\{\kappa_i\}$. Without loss of generality, we can image that $\kappa_1 \geq \kappa_2 \geq \cdots \geq \kappa_\nu$. Therefore, for the generic

case, there exist a permutation matrix P and a selection matrix S such that an optimal pair (C^*, V_j^*) can be obtained as $C^* = P^T (S\,C)\,P$, $V_j^* = P^T\,V_j$, where

$$C^* = \mathrm{diag}(C_1, \ldots, C_\nu)\,, V_j^* = (V_{j,1}^T, \ldots, V_{j,\nu}^T)^T\,, \tag{5.5}$$

and where each C_i and $V_{j,i}$ have the form of the Eq. 5.4. The actual optimal linear consensus algorithm is obtained choosing $F_j = P\,C^*$, and $B_j = P\,V_j^*$. A procedure describing the design algorithm can be found in [60], from which we also recall the following

Theorem 5.1 (Global Stability of Linear Consensus). *A logical linear consensus system of the form $x(t+1) = C^*\,x(t) + V_j^*\,u_j(t)$, where C^* and V_j^* are obtained as in Eq. 5.5 from a reachable pair (C, V_j), converges to a unique network agreement given by $\mathbf{1}_n\,u_j$ in at most $\mathtt{vis\text{-}diam}(C, V_j)$ rounds.*

5.3 Distributed Consensus Map Synthesis – The Self Routing Network Protocol (SRNP)

A complete exploitation of the logical consensus approach requires the optimal communication matrix C^* be computed in a fully distributed way. To this aim, we assume as in [84] that agents are able to exchange messages through a synchronous communication scheme. The input–propagation spanning tree strategy, described in the previous section, can be reproduced by requiring that every agent that is able to see the j–th input event u_j send a *supply message* offering its connection to all its neighbors. Agents sending this supply message are those specified by non–null elements of V_j, whereas agents receiving the message are specified by non–null elements of CV_j. Upon receiving a supply message, every agent send back a *confirmation message* to the agent with the lower index. After k steps, agents sending supply messages are those specified by non–null elements of $C^{k-1}V_j$ and those sending confirmation messages are

specified by non–null elements of $C^k V_j$. Therefore, having denoted with $C^*_{i,:}$ the i–th row vector of the optimal communication matrix, the agents that are able to set their communication row vector after k steps are specified by non–null elements of $\neg\left(C^{k-1}V_j\right) \wedge C^k V_j$. Denote with $m_{i,j}(k)$, for $i, j = 1, \ldots, n$, a Boolean variable taking the value 1 if, and only if, agent $\mathcal{A}_i$ receives a supply message from agent $\mathcal{A}_j$ at time k. It is straightforward to show that, for the matrix $M = \{m_{i,j}\}$, it holds

$$M(k) = \left(C^k V_j\right)^T \wedge C = Adj(i)^T \wedge C\,.$$

By construction, the row vector

$$w_i(k) \stackrel{\text{def}}{=} \left(m_{i,1}, m_{i,2} \wedge \neg m_{i,1}, \ldots, m_{i,n} \wedge \neg m_{i,n-1}\right)\,,$$

is either 0 or it contains only an entry set to 1 representing the agent with lowest index from which $\mathcal{A}_i$ must receive the value of the j–th input. Then, a generic agent $\mathcal{A}_i$ can set its communication row vector as $C_{i,:} \leftarrow w_i$. After having set its communication row vector, i.e. as soon as $C_{i,:} \neq 0$, or equivalently $\mathit{\Pi}_{h=1}^{n} \neg(C^*_{i,h}(k)) = 0$, an agent forwards the supply message to its neighbors and avoids any further modification of its communication row vector. Then, we can prove the following

Theorem 5.2 (Self–Routing Protocol). *A network of n agents running the distributed protocol*

$$\begin{cases} C^*_{i,:}(k+1) = C^*_{i,:}(k) \vee \left(\mathit{\Pi}_{h=1}^{n} \neg(C^*_{i,h}(k)\right) \neg V_j(i) \wedge w_i(k)\,, \\ C^*_{i,:}(0) = 0\,, \end{cases}$$

consents in finite time on the centralized communication matrix C^*.

Proof. The proof follows straightforwardly from the above description. In fact, if the generic agent $\mathcal{A}_i$ is able to "see" the j–th input, i.e. $V_j(i) = 1$, the

iterative rule reduces to $C^*_{i,:}(k+1) = C^*_{i,:}(k)$ and the initial null value is maintained. Otherwise, as soon as a supply message is received, the element in $w_i(k)$ corresponding to the message sender becomes 1, and the communication row vector is accordingly set. Finally, the term $\mathit{\Pi}_{h=1}^{n} \neg(C^*_{i,h}(k))$ prevents any further modification.

Example 5.1. Consider a network of $n = 5$ agents and the following pair (C, V_j) with $\nu = 2$:

$$C = \begin{pmatrix} 1\,1\,0\,0\,1 \\ 1\,0\,1\,0\,1 \\ 0\,1\,1\,1\,1 \\ 0\,1\,1\,1\,1 \\ 1\,0\,1\,0\,1 \end{pmatrix} \quad V_j = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} . \tag{5.6}$$

By applying the centralized algorithm described in [60], the following optimal linear consensus algorithm is obtained:

$$\begin{cases} x_1(t+1) = u(t)\,, \\ x_2(t+1) = u(t)\,, \\ x_3(t+1) = x_2(t)\,, \\ x_4(t+1) = x_2(t)\,, \\ x_5(t+1) = x_1(t)\,. \end{cases} \tag{5.7}$$

By applying the distributed rule of Theorem 5.2, we have

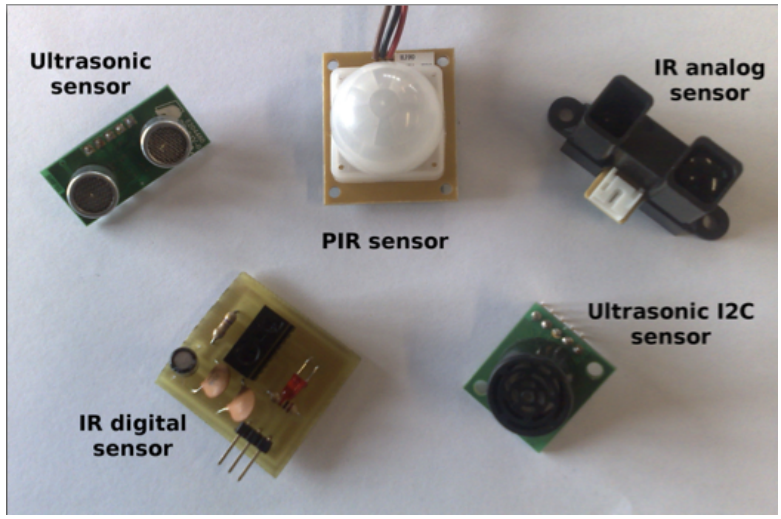


Fig. 5.1 The different types of sensors that have been used during the experiments.

$$M(0) = \begin{pmatrix} 1\,1\,0\,0\,0 \\ 1\,0\,0\,0\,0 \\ 0\,1\,0\,0\,0 \\ 0\,1\,0\,0\,0 \\ 1\,0\,0\,0\,0 \end{pmatrix}, \; W(0) = 0\,,$$

$$C^*_{i,:}(0) = 0 \;\; \forall i\,,$$

Algorithm 1: Distributed Synthesis of the Linear Consensus System

Input: $V_j(i)$
Output: $C^*_{i,:}$

$\tilde{C}_i \leftarrow 0$;
while $\Pi_h \neg C^*_{i,h} \wedge \neg V_{ji}$ **do**
`receive(`*buf, src*`);`
`; ◁ waiting for a supply message from neighbors of i`
$\tilde{C}_i \leftarrow$ src;
`;` `◁ index of the supply message sender`
$\mathcal{C}^*_{i,j} \leftarrow \{1\,,\ \forall j \in \tilde{C}_i\}$;
end
`send(`*buf, broadcast*`);`
`;` `◁ sends a supply message in broadcast`

$$M(1) = C\,,\ W(1) = \begin{pmatrix} 1\,0\,0\,0\,0 \\ 1\,0\,0\,0\,0 \\ 0\,1\,0\,0\,0 \\ 0\,1\,0\,0\,0 \\ 1\,0\,0\,0\,0 \end{pmatrix}\,,$$

$$C^*_{i,:}(1) = W(1)\ \ \forall i\,,$$

and $C^*_{i,:}(k) = C^*_{i,:}(1)$ for all $k \geq 1$, which gives the same logical linear consensus system as in Eq. 5.7.

Algorithm 1 reports the above described procedure, that is later referred to as the Self Routing Network Protocol (SRNP). Its asymptotic *computational complexity* is in the very worst case $O(n^2)$, where n is the number of agents,

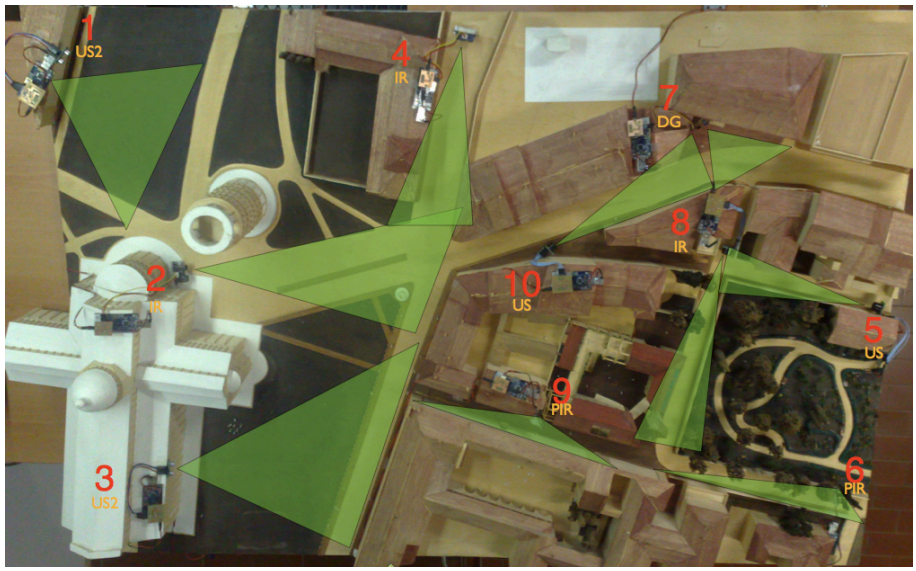


Fig. 5.2 Scale model of the area nearby Pisa's Leaning Tower. Green cones represent the visibility areas of every agents.

and its *space complexity* in terms of memory required for its execution is $\Omega(n)$. However, its implementation can be very efficient since it is based on Boolean operations on bit strings. Finally, *communication complexity* of a run of the consensus protocol in terms of the number of rounds is $\Theta(\texttt{vis-diam}(C, V_j))$.

5.4 Application to the Surveillance of an Urban Area

5.4.1 Experimental Setup

The effectiveness of SRNP has been shown through the following experimental setup related to the scenarios presented in [85, 60]. Consider a scale model

representing the urban area $\mathcal{W}$ nearby Pisa's Leaning Tower, where 10 agents A_i, each represented by a Sentilla Tmote–Sky [86], have been deployed to detect a possible intruder, represented by a radio controlled mini car. Agents are equipped with different sensors and are supposed to monitor fixed safety areas $\mathcal{W}_i, i = 1, \cdots, 10$ (Fig. 5.2). As in [60], the presence or absence of an intruder in region $\mathcal{W}_j$ can be modeled as a logical input u_j and every agent is required to estimate the $p = m$ decisions $y_i(t) = u_i(t), i = 1, \cdots, 10$. Due to limited sensing range, every agent is able to detect the presence of intruders only within its visibility areas, and thus a visibility matrix $V \in \mathbb{B}^{10\times 10}$ can be defined with $V_{i,j} = 1$ if, and only if, agent $\mathcal{A}_i$ can "see" in the area $\mathcal{W}_j$. The alarm state of the system is $X \in \mathbb{B}^{10\times 10}$, with $X_{i,j} = 1$ if agent $\mathcal{A}_i$ reports an alarm about the presence of an intruder in the area $\mathcal{W}_j$. The alarm can be set because an intruder is actually detected by an agent, or because of communications with neighboring monitors. The communication graph is assumed to be connected as required from theory (see Section 5.3).

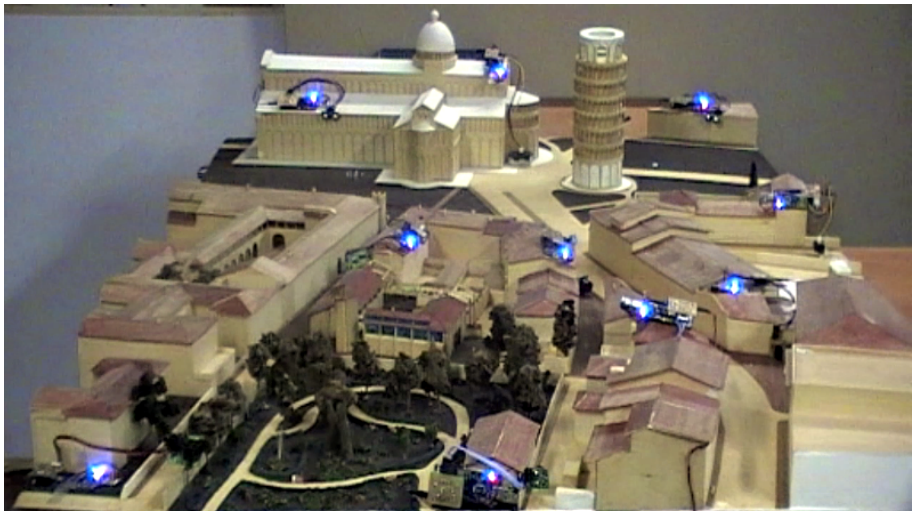


Fig. 5.3 Snapshot of the experiment: execution of the SRNP.

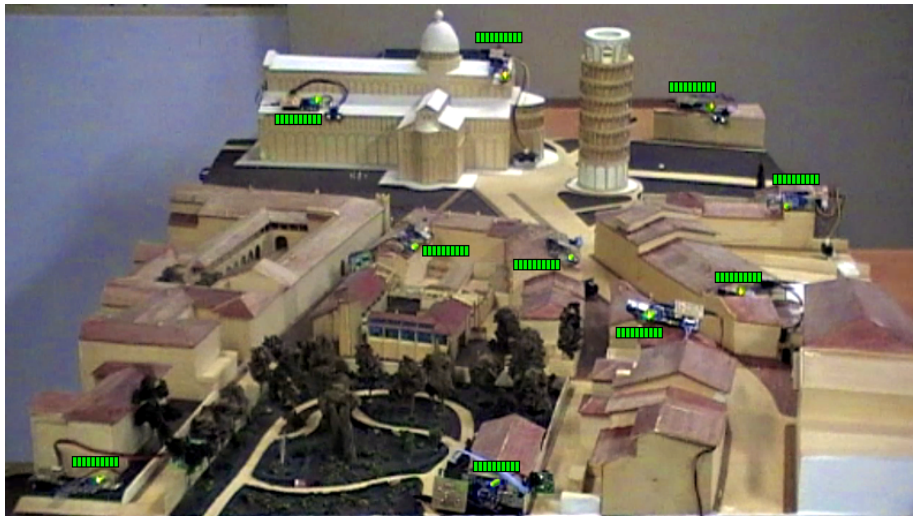


Fig. 5.4 Snapshot of the experiment: agents in intrusion detection mode. The colored flag of 10 squares represents the alarm state X_i of a generic agent $\mathcal{A}_i$ and shows its knowledge about the presence of an intruder in every visibility areas.

This type of mote is an MSP430–based battery–powered board, with an 802.15.4–compatible CC2420 radio chip. Every mote runs Contiki 2.0 operating system and uses μIP communication protocol [87, 88]. The Contiki OS is optimized for embedded systems with limited hardware resources and wireless connectivity, such as these motes, and it enables multi-thread programming. A Tmote-Sky represents a natural platform for implementing a logical consensus system. The platform is equipped with three colored LEDs and two light sensors, and it is provided with connectors for installing extra sensors by I^2C bus and analog to digital converter. The set of extra sensors used in the experiments comprises ultrasonic sensors, IR range finder, and PIR–based motion detectors (Fig. 5.1).

Each agent is endowed with the ability to construct its row vector $C^*_{i,:}$, by executing of the algorithm presented in Section 5.3. In the current implementation, communication follows a *round-robin* scheme, which requires pre–

synchronization of the agents' clocks (via e.g. the solutions in [79, A2]) and allows each agent to send a message to its neighbors during a pre–allocated time–slot (its duration is $9 \cdot 10^{-2}$ sec on the available hardware). Moreover, the length of the entire round depends on the number of agents participating in the communication, which has to be negotiated every time that a node joins or leaves the network (for 10 agents, a possible round length is 1 sec). Limitations of the adopted scheme are apparent but does not affect the general applicability of SRNP.

5.4.2 Experimental Results

The reader may refer to the site `http://www.youtube.com/watch?v=QoGwuWoSgCw` for the complete simulation run. The aim of the experiment is to show two main properties of SRNP: 1) its ability to reconfigure upon en-

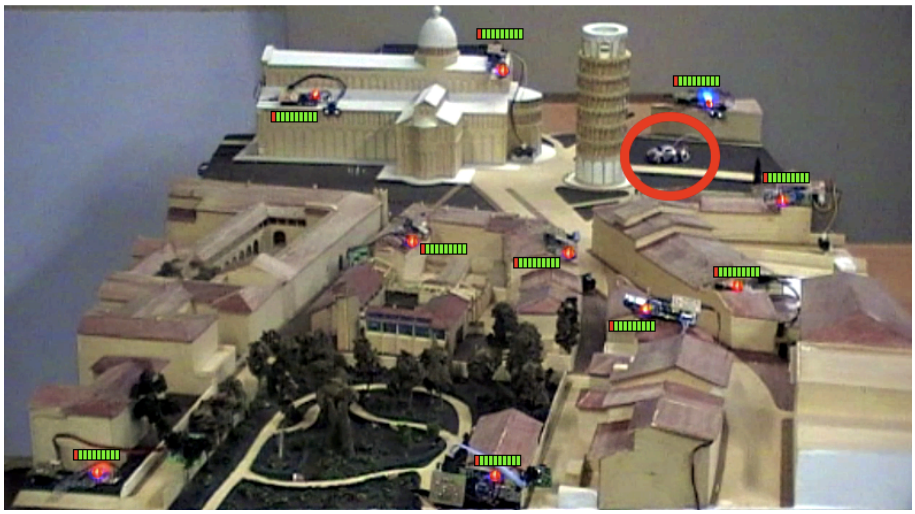


Fig. 5.5 Snapshot of the experiment: detection of the intruder by the agent $\mathcal{A}_1$.

trance of a new agent, and 2) its ability to realize a distributed intrusion detection system through execution of a logical consensus system. In the video, the following conventions are adopted to represent the different operation phases of the agents: blue represents the self–routing phase, green means that agents is operating in intrusion detection mode and sees no intruder, red is turned on when the agent is informed of the existence of an intruder, red and blue LEDs simultaneously turned on mean that the agent has seen the intruder in its visibility area.

The simulation starts with all agents running the SRNP so as to establish a communication matrix C^* that enables consensus on the different input events $u_1, \cdots, u_{10}$ (Fig. 5.4). At conclusion of this phase, agents are ready to detect and consent on the presence of an intruder (Fig. 5.4.1). Afterward, a new agent is added, which triggers a reconfiguration of the network including first a re–synchronization and then a re–execution of the SRNP. An intruder, represented by the radio controlled car is introduced. As soon as the car becomes close to agent $\mathcal{A}_1$, the agent detects it (its blue and red LEDs are both turned on) and the information is dispatched through the network according to the logical consensus scheme (Fig. 5.4.1). Therefore, every single agent can be polled to know about the presence and the location of the intruder. Whenever the intruder reaches an uncovered area, its presence is not detected (Fig. 5.4.2), which would require the introduction of a new agent in the system.

5.5 Failure of Linear Consensus

The limitation of the approach of the above solution is that the obtained map only converges to the correct value if every agent process and send correct information.

Here, the objective here is to further extend the presented approach by describing a fully distributed synthesis procedure that generates a logical nonlin-

Fig. 5.6 Snapshot of the experiment: the intruder has moved in an area that is out of every agents' range.

ear consensus map that is able to tolerate misbehaving agents that may send incorrect information, due to spontaneous failure or even tampering. The procedure is formalized as a distributed protocol, called SR^{2}NP, which is guaranteed to converge under similar visibility and communication conditions. The protocol is based on the known result of [30] ensuring that redundant minimum–length paths from a generic input u_j to every agent of the network can be found if the number of faults is bounded by $\gamma \in \mathbb{N}$ and the communication graph is at least $(2\gamma + 1)$–connected.

In this perspective we want to solve the following:

Problem 5.2. [Robust Design] Given a decision system as in Eq. (5.1), a visibility matrix V, a communication matrix C, and a maximum number γ of faulty agents, design a robust logical consensus system allowing all correct agents to consent on the centralized decision $\tilde{y} = f(u)$, i.e.

$$Y_i(t) = (\tilde{y})^T \,,\ \forall i \notin \mathit{\Gamma}, t \geq \bar{N} \,,$$

where $\mathit{\Gamma}$ is the set of faulty agents, and $\bar{N}$ is a sufficiently step number.

5.5.1 Robust Logical Information Flow

Failure of an agent can lead the linear logical consensus system to an incorrect and disconnected decision. In this section, we firstly discuss how to generate robust logical consensus maps solving Problem 5.2 in a centralized way. Consider that temporary faults, occurring when e.g. the measures of some agents are corrupted by noise, can be modeled as variations in the initial state $x(0)$. These faults pose no problem even in the case of linear logical consensus maps, which were shown to possess a unique and globally stable equilibrium $\mathbf{1}_n u_j$, where $\mathbf{1}_n$ is a vector with every elements to 1 [60]. This implies that temporary false alarms are canceled out by the system itself.

The problem becomes more difficult and interesting in case of permanent faults that occur whenever one or more agents are damaged, due to e.g. a spontaneous failure, or even tampering, and do not correctly execute the consensus algorithm. In this case, the convergence of the algorithm to the correct value may not be reached and the system may not be able to consent on the global decision. This motivates the development of techniques for synthesizing robust logical consensus systems. Suppose that a maximum number of $\gamma \in \mathbb{N}$ faulty agents have to be tolerated. The key to solve such a problem is in redundancy of input measurement and communication. Intuitively, a minimum number r of such sensors must be able to measure the $j-th$ input u_j and/or confirm any transmitted data x on u_j. In particular, to tolerate up to γ faults it is sufficient to chose $r = 2\gamma + 1$ ([30]). Therefore, we are concerned with the following:

Definition 5.1 (Reachability with redundancy).
An agent $\mathcal{A}_i$ is said to be reachable from input u_j with redundancy $r \in \mathbb{N}$, or

shortly r–reachable, if, and only if, it can receive at least r measurements of u_j between a direct measurement of u_j and messages of other agents that are r–reachable from u_j.

Moreover, *redundant minimum–length* paths are to be found such that information on u_j can robustly flow through the network, but such paths cannot be found by considering successive powers of $C^k\, V_j$, because only r–reachable agent are permitted to propagate information over the network. Thus, we need a procedure for finding to which agents the value of input u_j can be *robustly* propagated. Given a pair (C, V_j), we can conveniently introduce the *Robust Reachability matrix* ($R_j^2 \in \mathbb{B}^{n\times l}$), assigned with input u_j, whose k–th column represents the nodes which are r–reachable from input u_j at the k–th step. Algorithm 2 shows an iterative procedure allowing the evaluation of the Robust Reachability matrix, which is explained below.

Denote with S_k the set of agents that are able to "securely" estimate the value of input u_j. First observe that we have $S_0 = \{i_1, i_2, \cdots i_c\}$, whose elements are the indices of the non–null elements of the vector V_j. Moreover, we have $S_1 = S_0 \cup \{i_{c+1}, \cdots, i_{c+p}\}$, where the new indices corresponds to agents that can receive a message containing the value of u_j from at least r agents in S_0. These new indices are the non–null elements of the vector

$$\tilde{W}_j^1 = T_r(C * V_j)\,,$$

where $*$ is the integer product between two matrices and T_r is the threshold map

$$T_r : \mathbb{N}^n \to \mathbb{B}^n$$
$$w_i \mapsto \begin{cases} 1 \text{ if } w_i \geq r \\ 0 \text{ otherwise} \end{cases} .$$

The set of agents that are r– reachable after 1 step are then given by the non–null elements of the vector $W_j^1 = \tilde{W}_j^1 + V_j$. In general, S_k is obtained as the

set of agents that are specified by non–null elements of the vector

$$W_j^k = T_r(C * W_j^{k-1}) + W_j^{k-1}\,,$$

with $W_j^0 = V_j$. Note that W_j^k is by definition the $(k+1)$–th column of R_j^2. Computation of R_j^2 can stop as soon as the sequence becomes stationary, i.e., $W_j^k = W_j^{k-1}$.

Algorithm 2: Algorithm for Robust Reachability matrix.

Input: C,V_j,γ
Output: R_j^2

$r = 2\gamma + 1$;
$k = 1$;
$W_j^k = V_j$;
do
    $W_j^{k+1} = T_r(C * W_j^k) \vee W_j^k$; ◁ `r-reachable nodes after k-steps`
    $k = k + 1$;
while $W_j^k \neq W_j^{k-1}$◁ `no cycle condition`;
$R_j^2 = [W_j^1, \cdots, W_j^{k-1}]$;

Therefore we can prove the following:

Theorem 5.3. *A pair (C, V_j) is r–reachable if, and only if the logical product of the elements of the last column of the matrix R_j^2 is equal to 1, i.e. $\wedge_{p=1}^n (r_{p,l}) = 1$, where $[r_{1,l}, r_{2,l}, \cdots, r_{n,l}]^T$ is the last column of R_j^2.*

Proof. The proof of sufficiency straightforwardly follows from the procedure for the construction of the R_j^2 matrix. Indeed, since the k–th column of that matrix represents nodes that are r–reachable at the k–th step, the non–null element of the last column indicates which nodes are eventually r–reachable.

For this reason if all the elements of the last column of R_j^2 are non-null all nodes are r-reachable, i.e. the pair (C, V_j) is r–reachable.

To show that the condition is also necessary, let us proceed by absurd. Suppose that a node is r–reachable, while the corresponding element of the last column of R_j^2 equals 0. This means that such a node cannot be reached by at least r messages containing the value of u_j, which contradicts the hypothesis of r–reachability of the network.

Example 5.2. Consider the network of $n = 5$ agents depicted in Fig. 5.7a with

$$C = \begin{pmatrix} 1\,0\,1\,0\,0 \\ 0\,1\,1\,1\,0 \\ 1\,1\,1\,1\,0 \\ 0\,0\,0\,0\,0 \\ 1\,1\,1\,0\,1 \end{pmatrix}, \text{ and } V_j = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}.$$

By applying Algorithm 2, we can verify whether the pair (C,V_j) is 3–reachable or not, i.e. if it is feasible to construct a map that is robust to $\gamma = 1$ faulty agents. First we have $W_j^0 = V_j = (1,1,0,1,0)^T$. At the first step ($k = 1$), we have $W_j^1 = \tilde{W}_j^1 + W_j^0$, i.e.

$$W_j^1 = T_r \begin{pmatrix} 1 \\ 2 \\ 3 \\ 0 \\ 2 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}.$$

Analogously at the second step ($k = 2$) we have $W_j^2 = (1,1,1,1,1)^T$. As $W_j^2 \neq W_j^1$, the procedure proceeds to the next step. At $k = 3$ we obtain $W_j^3 = W_j^2$ and thus the procedure can stop. The resulting Robust Reachability Matrix is

$$R_j^2 = \begin{pmatrix} 1\,1\,1 \\ 1\,1\,1 \\ 0\,1\,1 \\ 1\,1\,1 \\ 0\,0\,1 \end{pmatrix}.$$

Remark 5.1. Once the robust reachability matrix R_j^2 of a pair (C, V_j) has been computed by using Algorithm 2, a logical consensus map tolerating up to γ faults can be built by choosing $r = 2\,\gamma + 1$ ([30]). The explicit construction of the map will be discussed below for the distributed design setting.

5.6 Distributed Synthesis of a Robust Consensus Map: the Self–Routing Robust Network Protocol (SR^{2}NP)

A practical exploitation of the logical consensus approach requires that an optimal r–reachable communication pair (C^*,V_j) is computed in a fully distributed way. To this aim, we assume that agents are able to exchange messages through a synchronous communication scheme. The correct multi–input–propagation spanning tree strategy, can be reproduced by requiring that every agent that is able to "robustly see" the j–th input event u_j sends a *supply message* offering its connection to all its neighbors. A distributed procedure, that can be used by the network agents to correctly propagate the information through the network,

i.e. to find the C^* matrix, is described in Algorithm 3. Starting from C_i^* vectors, each agent is able to build a suitable distributed nonlinear iteration map F_i to combine its information with the ones of its neighbors. An agent $\mathcal{A}_i$ that is able to measure u_j can update its state x_i via the local rule $F_i = u_j$. Any other agent needs to rely on information received from its C–neighbors, including possible compromised data. Denote with $\rho = \{k \ : \ C_i^*(k) = 1\}$ the set of C^*–neighbors of agent $\mathcal{A}_i$. Denote also with $S_i \in \mathbb{N}^{\sigma \times (\gamma+1)}$ an integer matrix containing the $\sigma = \binom{r}{\gamma+1}$ combinations of $\gamma + 1$ elements extracted from a subset with cardinality r of the set of $\mathcal{A}_i$'s C^*–neighbors, i.e. $S_i = \mathcal{C}_{\gamma+1}^r(\rho)$. A robust update can be obtained by computing the (logical) sum of the $\sigma = \binom{r}{\gamma+1}$ terms, with $r = 2\gamma+1$, that are composed of the logical product of $\gamma+1$ states x_s extracted from its C–neighbors, i.e.

$$x_i(t+1) = F_i(x(t), u_j) \tag{5.8}$$

with

$$\begin{aligned} F_i : \mathbb{B}^n \times \mathbb{B} &\to \mathbb{B} \\ (x, u_j) &\mapsto \begin{cases} u_j & \text{if } V_{i,j} = 1, \\ \sum_{h=1}^{\sigma} \prod_{k=1}^{\gamma+1} x_{s_{h,k}} & \text{otherwise} \end{cases} \end{aligned} \tag{5.9}$$

where $s_{h,k}$ is the k–th element of the h–th row of the S_i.

In the remainder of the work, we need the following:

Lemma 5.1. *The local update rule $x_i(t+1) = F_i(x(t))$ where F_i is constructed as in Eq.* (5.9) *converges to the consensus value $\bar{x}_i(t) = \alpha, \ t \geq \bar{N}$, if the at least $\gamma + 1$ components of the state vector $x(t) \in \mathbb{B}^n$ equals $x_k = \alpha$.*

Proof. Let us proceed by absurd. Suppose that $x(t)$ is composed of γ values equal to $\neg\alpha$, whereas the remaining $\gamma + 1$ values are equal to α, and suppose that $\bar{x}_i(t) = \neg\alpha, \ t \geq \bar{N}$.

Let us consider two different cases: $\alpha = 1$ and $\alpha = 0$. If $\alpha = 1$, we have $\bar{x}_i(t) = 0$. Indeed, as F_i is the logical sum of σ terms consisting of the logical

product of $\gamma+1$ states $x(t)$, in order to have $\bar{x}_i(t)=0$, all the σ logical terms must be equal to 0. It trivially holds that this is not possible because one of the σ terms is the logical product of $\gamma+1$ values $x_k=\alpha$, and so in this case the lemma follows.

If $\alpha=0$ we have $\bar{x}_i(t)=1$. In order to have this, at least one of the σ terms of the logical sum must be equal to 1. Since all the σ terms are composed of $\gamma+1$ values, it turns out that at least one of these values in the logical product is equal to 0. Therefore, it trivially holds that all the σ terms are equal to zero. This contradicts the hypothesis and proves the thesis. ■

Algorithm 3: Distributed Algorithm for SR^{2}NP

Input: $V_{i,j}$,γ
Output: C_i^*, F_i

$r \leftarrow 2\gamma+1$;
$\sigma \leftarrow \binom{r}{\gamma+1}$;
$C_i^* = 0^{1\times n}$;
while $(\texttt{SUM}(C_i^*) < r) \wedge (\neg\, V_{i,j})$ **do** ◁ verify if the update of C_i^* is necessary
  receive(*id*);
  $C_i^*(id) = 1$; ◁ update Connecting Neighbors Set
end
$\rho \leftarrow \{k : C_i^*(k) = 1\}$;
$S_i \leftarrow \mathcal{C}^{\nabla}_{\gamma+\infty}(\rho)$;
$F_i \leftarrow \sum_{h=1}^{\sigma} \prod_{k=1}^{\gamma+1} x_{s_{h,k}} + V_{i,j}\, u_j$;
send(*i*);

Note that that, by using Algorithm 3, a nonlinear logical consensus system as in Eq. (5.8) can be obtained, which is able to tolerate up to γ permanent faults. In this perspective the following theorem is a solution to Problem 5.2:

Theorem 5.4. *Given an r–reachable pair (C, V_j), the protocol described in Algorithm 3 produces a robust logical nonlinear consensus system of the form*

$x(t+1) = F(x(t), u_j(t))$, *which has a unique equilibrium. If the number of permanent faults is upper bounded by* γ, *the system's equilibrium is given by* $\mathbf{1}_n u_j$.

Proof. Firstly, we need to prove that the robust logical nonlinear consensus system constructed according to Algorithm 3, has a unique equilibrium point. Recalling the procedure described in Algorithm 3, we have that agents directly reachable from u_j *are specified by non–null elements of vector* V_j. *Let us denote with* $S_0 = \{i_1, i_2, \cdots i_c\}$ *the index set of these agents. Then, the strategy to allow the information on* u_j *flowing through the network is obtained if agents in* S_0 *communicates their measurement to all their neighbors, which in turn, if* r*–reachable, will communicate it to all their neighbors, and so on. In this way, we have that every agent* $\mathcal{A}_i$ *receives* u_j *from a path originating from agents in* S_0. *Indeed, it trivially holds that agents reached after* k *steps have received the input value from agents that were "secure" reached in the previous* $k-1$ *steps. By reordering the agents according to the order that they are reached by the input propagation, the pair* (C^*, V_j^*) *can be rewritten as* $(\tilde{C}^*, \tilde{V}_j^*)$ *where* $\tilde{C}^* = P^T (S\, C) P$ *is a lower triangular block matrix, and* $\tilde{V}_j^* = P^T V_j$, *where* S *is a selection matrix and* P *is a permutation matrix. More precisely, we have:*

$$\tilde{C}^* = \left(\begin{array}{cccc|c} 0 & 0 & \cdots & \cdots & 0 \\ \tilde{C}_{1,1} & 0 & \cdots & \ldots & 0 \\ \tilde{C}_{2,1} & \tilde{C}_{2,2} & \ddots & \vdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \tilde{C}_{l,1} & \cdots & \cdots & \tilde{C}_{l,l} & 0 \end{array}\right), \tilde{V}_j^* = \begin{pmatrix} 1_\nu \\ 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \end{pmatrix} \tag{5.10}$$

where $\nu \geq r$ are the agents which directly measure the input u_j. Since the robust nonlinear consensus map is constructed as in Eq. (5.9) *by using the pair (C^*, V_j^*), the incidence matrix of the iteration map, has a strictly lower–triangular form. Thus, from Chapter 4, it follows that the network reaches an agreement on a unique equilibrium in a finite number of steps.*

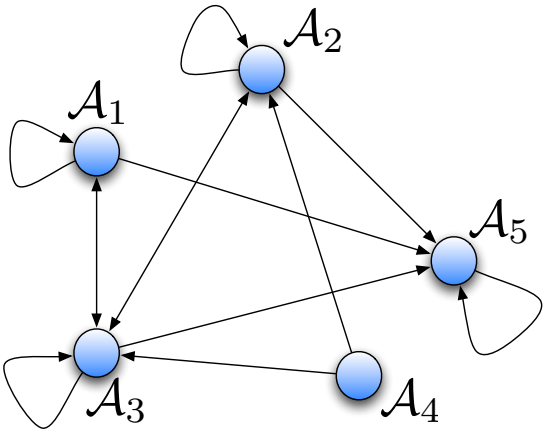
Finally, we need to prove that if the number of permanent faults is upper bounded by γ, then the equilibrium point of the system is $\mathbf{1}_n\, u_j$. Observe that, referring to the blocks in Eq. (5.10), according to the first elements of V_j^*, the first row block gives the simple relation $x_{i,0}(t) = u(t)$. Then, the second block corresponds to a set of agents that are updated after 1 step. In particular, as they are receiving the value from the agents in the first block, by Lemma 5.1 we have: $x_{i,1}(t) = x_{i,0}(t) = u_j$. At the generic iteration k, a block of variables $x_{i,k}$ are updated through the k–th matrix $\tilde{C}_{i,k}$. Hence, as they are receiving the value from the agents in the previous blocks, by Lemma 5.1, we have $x_{i,k}(t) = x_{i,k-1}(t) = u_j$. By repeating this procedure for all blocks, and since all agents that are directly reachable from input u_j read the same value u_j, by Lemma 5.1 we can prove that the entire network reaches an agreement on the unique global equilibrium $x^* = \mathbf{1}_n\, u_j$ in a finite number of steps. ■

Example 5.3. Consider the same scenario as in Example 5.2 and Fig. 5.7. The robust distributed nonlinear iteration map is obtained by using the procedure described in Algorithm 3. In particular, at step $k = 1$, agents $\mathcal{A}_\infty, \mathcal{A}_\in$, and $\mathcal{A}_\triangle$ set

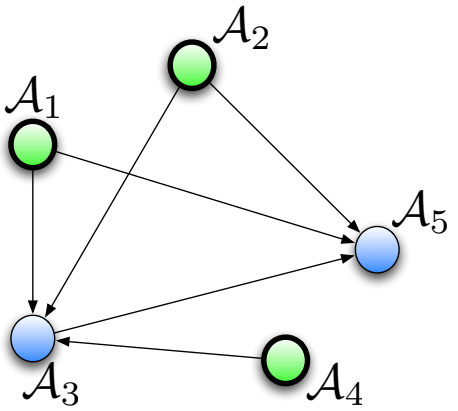
$$C_1^* = C_2^* = C_4^* = \Big(0,\, 0,\, 0,\, 0,\, 0 \Big),$$

and send a supply message to their neighbors. Since at step $k = 2$, the agent $\mathcal{A}_\ni$ receives three messages from $\mathcal{A}_\infty, \mathcal{A}_\in, \mathcal{A}_\triangle$, it sets

$$C_3^* = \Big(1,\, 1,\, 0,\, 1,\, 0 \Big),$$



(a)



(b)

Fig. 5.7 Example of network with $n = 5$ agents (5.7a), and its robust communication graph (5.7b), where green nodes are able to directly measure input u_j.

and is able to send a supply message to its neighbors. At the same step, $\mathcal{A}_{\bigtriangledown}$ sets

$$C_5^* = \Big(1, 1, 0, 0, 0\Big),$$

and, in order to be able to exit out of the while loop condition, i.e. to be r–reachable, waits to receive an other message. Finally, at step $k = 3$, since $\mathcal{A}_{\bigtriangledown}$ receives a message from $\mathcal{A}_{\ni}$, it is able to set

$$C_5^* = \Big(1, 1, 1, 0, 0 \Big) ,$$

and thus leaves the loop. Afterward, each agent $\mathcal{A}_{\rangle}$ can evaluate its own non-linear logical map, F_i as in Eq. (5.9). Therefore we have

$$\begin{cases} x_1(t+1) = u_j(t) \,, \\ x_2(t+1) = u_j(t) \,, \\ x_3(t+1) = x_1(t)\, x_2(t) + x_1(t)\, x_4(t) + x_2(t)\, x_4(t) \,, \\ x_4(t+1) = u_j(t) \,, \\ x_5(t+1) = x_1(t)\, x_2(t) + x_1(t)\, x_3(t) + x_2(t)\, x_3(t) \,. \end{cases}$$

whose communication graph is showed in Fig. 5.7b.

5.7 Application to the Surveillance of an Urban Area (continued)

5.7.1 Experimental Setup

The effectiveness of SR^{2}NP is shown through a modified version of the experimental setup already presented in Section 5.4.1, and involving a scale model representing the urban area W nearby Pisa's Leaning Tower. In the model, 14

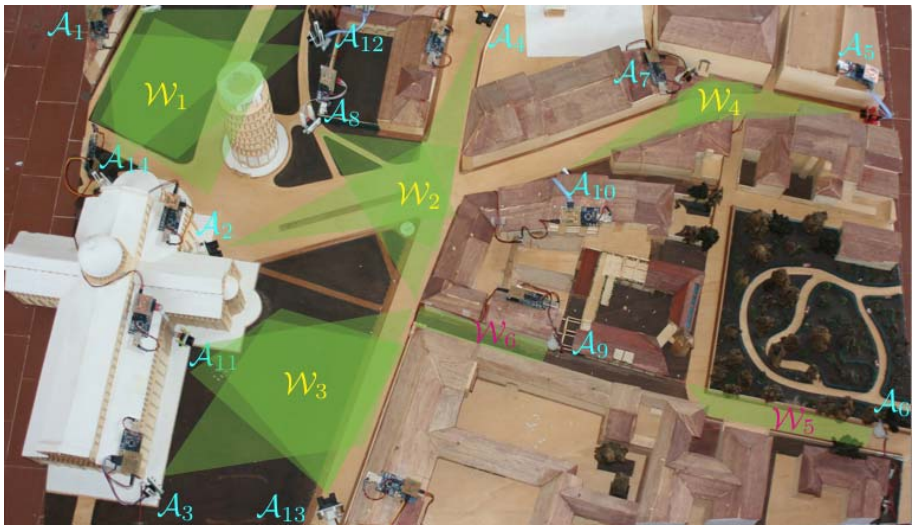


Fig. 5.8 Scale model of the urban area nearby Pisa's Leaning Tower: sensor nodes location and shared areas (yellow $\mathcal{W}_\rangle$).

agents $\mathcal{A}_1, \cdots, \mathcal{A}_{14}$ have been deployed to detect a possible intruder, represented by one or more radio controlled mini car and some agents may fail.

As in the previous experiment, agents are represented by Sentilla Tmote–Sky nodes equipped with different kind of sensors (including e.g. ultrasonic sensors, IR range finder, and PIR–based motion detectors), which are used in the experiments to monitor shaped and limited safety areas $\mathcal{W}_|$, $j = 1, ..., 6$ (Fig. 5.8)

Due to limited sensing range, every agent is able to detect the presence of intruders only within its visibility areas, and thus a visibility matrix $V \in \mathbb{B}^{14\times 6}$ can be defined with $V_{i,j} = 1$ if, and only if, agent $\mathcal{A}_i$ is able to monitor the area $\mathcal{W}_|$. The presence or absence of an intruder in region the $\mathcal{W}_|$ is modeled as a logical input u_j and each agent is responsible for running its corresponding row, f_i, of the logical map obtained through the execution of the SR^{2}NP. So the alarm will be set if, and only if an intruder is detected at least by $\gamma + 1$ agents, between those able to directly monitor the area $\mathcal{W}_|$, and those are r–

Fig. 5.9 Snapshot of the experiment: execution of SR^2NP.

reachable. The protocol presented in this thesis is general and is applicable with any collection of heterogeneous agents.

Fig. 5.10 Snapshot of the experiment: agents entering in intrusion detection mode.

Fig. 5.11 Snapshot of the experiment: intruder (yellow circle) is detected by agents $\mathcal{A}_1$,$\mathcal{A}_{12}$ and the intrusion is correctly dispatched to every agent even though the faulty agent $\mathcal{A}_{14}$ (red circle) transmits an incorrect information.

5.7.2 Experimental Results

The reader may refer to the site `http://www.youtube.com/watch?v=ptk0wDwIqKs` for the complete simulation run. Snapshots of the experiment are reported in Fig. 5.9, 5.10, 5.11, and 5.12, which reveal the ability of SR^{2}NP to: 1) tolerate γ faults due to proximity malfunction sensors, 2) reconfigure upon entrance of a new robot, and 3) realize a distributed intrusion detection system through execution of a logical consensus system.

The following conventions are adopted to represent the different operation phases of the agents: blue LEDs represent that the self–routing phase is in progress; green LEDs indicate that agents are running the monitoring phase and no intruders have been detected yet, while red LEDs turn on whenever an agent is informed of the existence of an intruder. Red and blue LEDs on the same agent indicate that an intruder has been detected in its visibility area.

The experiment starts with all agents running the SR^{2}NP so as to establish a communication matrix C^* that is used to find a suitable distributed nonlinear map that enables consensus on the shared input events $u_1, \cdots, u_6$ (Fig. 5.9). After completion of the self–routing phase, agents run the monitoring phase so that they are ready to detect and consent on the presence of an intruder (Fig. 5.10). During the monitoring phase, whenever new agents join in the sensor network, a new self–routing phase starts along with a new clock synchronization. In the experiment, as soon as the intruder enters the area $\mathcal{W}_\infty$, agents $\mathcal{A}_1$ and $\mathcal{A}_{12}$ correctly detect its presence (see the red and blue LEDs on) and inform their neighbors, while a faulty agent $\mathcal{A}_{14}$, does not detect it and sends the incorrect information to the others (Fig. 5.11). Despite of this fault, the other agents correctly detect the intruder based on their own robust logical map. When the intruder moves out of region $\mathcal{W}_\infty$ and enters into region $\mathcal{W}_\ni$, a new consensus is achieved and all agents become aware of the new intruder's position (Fig. 5.12).

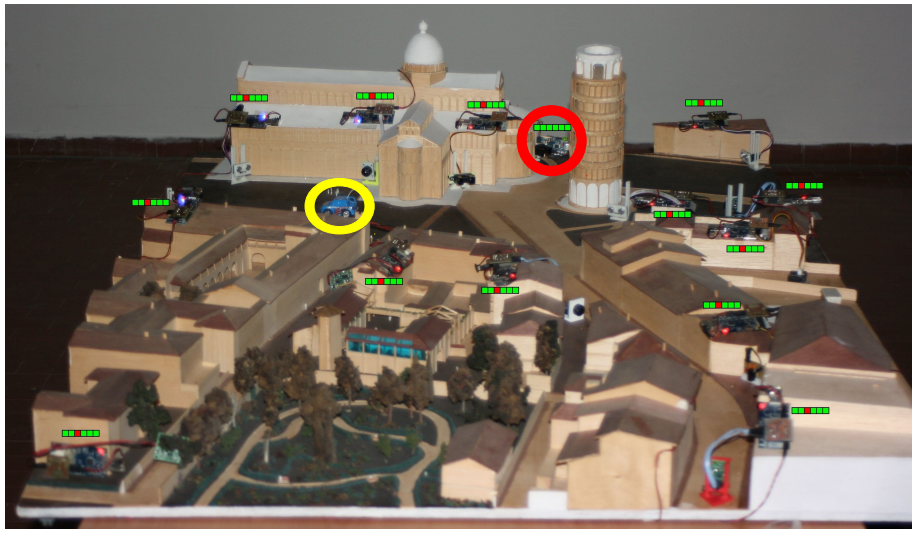


Fig. 5.12 Snapshot of the experiment: intruder, now in area $\mathcal{W}_\ni$, is correctly detected by agents $\mathcal{A}_3$,$\mathcal{A}_{11}$,$\mathcal{A}_{13}$.

Chapter 6

Behavior Classification and Security

Human and animal societies are based on the establishment of "rules of interaction" among the individuals of their societies which may consist of either collaborative or non–collaborative *behaviors* that each (possibly different) individual may display towards the others [12]. The adoption of similar notions of "sociality" and heterogeneity in Robotics is advantageous in many tasks, where a coordination among agents with analogous or complementary capabilities is necessary to achieve a shared goal. More specifically, "robotic societies" are distributed multi–agent systems where each agent is assigned with a possibly different local goal, but needs to coordinate its actions with other neighboring agents. The flexibility and robustness of such distributed systems, and indeed their ability to solve complex problems, have motivated many works that have been presented in literature (see e.g., [13, 14, 15, 16, 17, 18]). In this view, at the base of interaction between agents in a society it stands the ability of each individual (agent) to distinguish or *classify* the other neighboring members (agents) of the society that it gets in contact with, as belonging to the same group or species. In particular some robots may experience sensor or actuator failure, and thus be unable to follow the protocol rules. Moreover, since robot communication is based on a wireless network, an adversary could easily drop on communication as well as inject/modify packets. In such cases, safety requires that the society must be able to recognize such failures or intrusions

in order to activate countermeasures to preserve the overall system and its individuals. All these cases can be recast as behavior classification problems, i.e. the problem to classify heterogeneous agents that "behave" in a different way, due to their own physical dynamics or to the rules of interaction they are obeying, as belonging to a different species. The objective is ambitious and indeed very difficult to be achieved without a priori knowledge of the interaction rules, but a viable solution can be found if e.g. the models describing the behavior of the different species are known in advance. In the following will be presented different techniques allowing agents to classify other agents whose effectiveness is shown through the two examples of robot societies introduced in the Chapter 3: an automated transportation system including different types of drivers (Sec. 3.3), and a colony of the polymorphic tree dwelling ant, *Daceton Armigerum*, during the foraging process (Sec. 3.2).

6.1 Distributed Classification

In this section after having formalized the classification problem, this thesis presents a systematic procedure allowing to build a decentralized classification system once the models describing the behavior of the different species are given. It is based on a decentralized identification mechanism, by which every agent classifies its neighbors using only locally available information. Moreover, this work presents a consensus algorithm allowing agents to reach an agreement on the species to which other neighboring agents belong by sharing the information extracted by using their local classifier. Some of the results reported in this chapter can be found in [A5].

6.1.1 Classification Problem

Based on the protocol introduced in Section 3.1 we can give the following

Definition 6.1. A *behavior* in the time interval $[0, t)$is the trajectory

$$\phi_{\mathcal{H}_i^{(r)}}(q_i^0, \sigma_i^0, \tilde{I}_i(t)) = (\bar{q}_i(t), \bar{\sigma}_i(t_{k+1})$$

solution of the system in Eq. 3.2, subject to the input $\tilde{I}_i(t)$ being the history of its neighbor configuration set $I_i(\tau)$, for $\tau \in [0, t)$.

Definition 6.2. $\mathcal{A}_i$ is said to be *compatible* with the species S^r if its behavior $(\bar{q}_i(t), \bar{\sigma}_i(t_{k+1})$ is close enough to the evolution of the hybrid model $\mathcal{H}^{(r)}$, i.e.

$$\|\bar{q}_i(t) - \pi_{\mathcal{Q}}(\phi_{\mathcal{H}_i^{(r)}}(\bar{q}_i(t_k), \bar{\sigma}_i(t_k), I_i(t)))\| \leq \epsilon, \quad \forall t,$$

where $\|\cdot\|$ is the Hausdorff distance, $\pi_{\mathcal{Q}}$ is the projector over the set $\mathcal{Q}$, and ϵ is an *accuracy* based on the quality of available sensors.

Definition 6.3. An agent is said to *belong* to the species S^r if, and only if it is compatible only with that species.

A group of agent $\mathcal{A} = \{\mathcal{A}_1, \cdots, \mathcal{A}_n\}$ trying to learn the species to which another agent $\mathcal{A}_i$ belongs needs to solve the following

Problem 6.1. Given the complete description of the p species, a measure of the behavior $\bar{q}_i$ and a partial measure of the agent's neighbor configuration set $I_i^h = I_i \cap \mathcal{V}_h \ h \in \{1, \cdots, n\}$, design a Distributed Species Classification System (DSCS) of the form

$$C_i^h = \mathsf{classifier}(\bar{q}_i, I_i^h)\,,$$

allowing agents exchanging outputs of their local classifiers to reach a global consensus on the species their neighbors belong to.

6.1.2 Local Classifier

To illustrate what challenges a local classifier have to cope with, continue referring to the running example. Suppose that the white car (third lane) in Fig. 3.7 is trying to learn what species the pink car belongs to. The pink car is executing a FAST maneuver in the second lane, which can be considered as part of overtaking of the black car if the vehicle belongs to the RH (E) species, as part of overtaking of the white car if the vehicle belongs to the LH (E)species. Moreover, the local classifier on the white car has complete visibility on the rear, left, and front regions, but cannot see the black car in the right region that is partially visible. Therefore, based on such local knowledge, the three species remain possible.

This is an example of a general local classifier that can be formally described as follows. Consider how a generic agent $\mathcal{A}_h$ can learn to which species another agent $\mathcal{A}_i$ belongs by using only locally available information. If a complete description of all species is available, $\mathcal{A}_h$ needs to determine to which models the observed behavior, or physical motion, of $\mathcal{A}_i$ best corresponds to. Approaches based on complete knowledge of a model's inputs (see e.g. [89]) cannot be applied as $\mathcal{A}_i$'s neighborhood N_i is generally, only partially known to $\mathcal{A}_h$. The proposed classifier is a two–step process: first, $\mathcal{A}_h$ computes an a–priori prediction of the set of possible behaviors that $\mathcal{A}_i$ can execute based on local information, for each species (*prediction phase*); then, the predicted behaviors are compared against the one actually executed and measured by $\mathcal{A}_h$ and those resulting close enough are selected (*classification phase*).

The prediction phase involves constructing a predictor for each species S^r, which is represented by an uncertain hybrid model $\tilde{\mathcal{H}}^{(r)}$. The model is composed of a nondeterministic automaton whose state $\tilde{\sigma}_i \in \Sigma_i$ represents the set of actions that $\mathcal{A}_i$ can perform based on local information, and whose transitions $\tilde{\delta}$ are the same as in δ. The main challenge in the construction of the

automaton is the estimation of an upper approximation $\tilde{c}_i$ of each detector condition c_i, that is achieved through the following results.

Proposition 6.1. *Given a detector condition c_i composed of a unique topology, i.e. $c_i = s_{i,1}$, a visibility-based upper approximation of it is*

$$\tilde{c}_i = s_{i,1}^{(h)}\, v_{h,1} + \neg\, v_{h,1}\,,$$

where $v_{h,1}$ is the event visibility of an observer onboard agent $\mathcal{A}_h$.

Proof. Based on the observer's visibility $v_{h,1}$, the topology can be written as

$$s_{i,1} = \underbrace{\left(\sum_{q_k \in I_i^h} \mathbf{1}_{\eta_{i,1}(q_i)}(q_k)\right)}_{s_{i,1}^{(h)}} + \underbrace{\left(\sum_{q_k \in I_i \setminus I_i^h} \mathbf{1}_{\eta_{i,1}(q_i)}(q_k))\right)}_{n_{i,1}^{(h)}}\,.$$

To prove the proposition, consider factorizing the event expression as follows. If $n_{i,1}^{(h)} = 0$, the event reduces to $c_i = s_{i,1}^{(h)}$, whereas if $n_{i,1}^{(h)} = 1$, it becomes $c_i = s_{i,1}^{(h)} + 1 = 1$. Then, the event expression can be factorized as

$$c_i = s_{i,1}^{(h)} n_{i,1}^{(h)} + 1\, n_{i,1}^{(h)}.$$

Moreover, in the case of $v_{i,1} = 1$, it holds $n_{i,1}^{(h)} = 0$ (the observer has complete visibility of the topologies) that implies $c_i = s_{i,1}^{(h)}$, whereas nothing can be said on the value of $n_{i,1}^{(h)}$. Therefore, the event c_i can be factorized w.r.t. the observer's visibility as

$$c_i = s_{i,1}^{(h)}\, v_{h,1} + \left(s_{i,1}^{(h)} \neg\, n_{i,1}^{(h)} + n_{i,1}^{(h)}\right) \neg v_{h,1}$$

Hence, it holds

$$\tilde{c}_i = \max_{n_{i,1}^{(h)} \in \mathbb{B}} c_i = \\ = s_{i,1}^{(h)}\, v_{h,1} + \max_{n_{i,1}^{(h)} \in \mathbb{B}} \underbrace{\left(s_{i,1}^{(h)} \neg\, n_{i,1}^{(h)} + n_{i,1}^{(h)} \right)}_{A} \neg v_{h,1}\,.$$

Finally, it holds $A = \max_{n_{i,1}^{(h)} \in \mathbb{B}} \left\{ s_{i,1}^{(h)}, 1 \right\} = 1$*, which gives the thesis.*

Proposition 6.2. *Given a detector map* c_i *of the type* $c_i = \neg s_{i,1}$ *, its visibility–based event is*

$$\tilde{c}_i = s_{i,1}^{(h)}\,,$$

where $v_{h,1}$ *is the event visibility of an observer onboard agent* $\mathcal{A}_i$*.*

Proof. Based on the observer's visibility $v_{h,1}$*, the topology can be written as*

$$s_{i,1} = \underbrace{\left(\neg \sum_{q_k \in I_i^h} \mathbf{1}_{\eta_{i,1}(q_i)}(q_k) \right)}_{s_{i,1}^{(h)}} \underbrace{\left(\neg \sum_{q_k \in I_i \setminus I_i^h} \mathbf{1}_{\eta_{i,1}(q_i)}(q_k) \right)}_{n_{i,1}^{(h)}}\,.$$

To prove the proposition, consider factorizing the event expression as follows. If $n_{i,1}^{(h)} = 0$*, the event reduces to* $c_i = 0$*, whereas if* $n_{i,1}^{(h)} = 1$*, it becomes* $c_i = s_{i,1}^{(h)}$*. Then, the event expression can be factorized as*

$$c_i = 0\, n_{i,1}^{(h)} + s_{i,1}^{(h)}\, n_{i,1}^{(h)} = s_{i,1}^{(h)}\, n_{i,1}^{(h)}.$$

Moreover, if $v_{h,1} = 1$*, it holds* $n_{i,1}^{(h)} = 1$ *that implies* $c_i = s_{i,1}^{(h)}$*, whereas nothing can be said on the value of* $n_{i,1}^{(h)}$ *and hence* $c_i = s_{i,1}^{(h)} \neg n_{i,1}^{(h)}$*. Therefore, the event* c_i *can be factorized w.r.t. the observer's visibility as*

$$c_i = s_{i,1}^{(h)}\, v_{h,1} + s_{i,1}^{(h)}\, n_{i,1}^{(h)} \neg v_{h,1}$$

Hence, it holds

$$\begin{aligned}\tilde{c}_i &= \max\nolimits_{n_{i,1}^{(h)}\in\mathbb{B}} c_i = \\ &= s_{i,1}^{(h)}\, v_{h,1} + s_{i,1}^{(h)} \underbrace{\max_{n_{i,1}^{(h)}\in\mathbb{B}} \left(s_{i,1}^{(h)}\right)}_{=1} \neg v_{h,1} = \\ &= s_{i,1}^{(h)}\, v_{h,1} + s_{i,1}^{(h)} \neg\, v_{h,1} = s_{i,1}^{(h)}\, (v_{h,1} + \neg\, v_{h,1}) = s_{i,1}^{(h)}\,,\end{aligned}$$

which gives the thesis.

The result can be given on the general form in the following

Theorem 6.1. *The visibility based upper estimation of a generic detector map* $c_i = \mathit{\Pi}_\gamma s_{i,k}\, \mathit{\Pi}_\rho \neg s_{i,j}$ *is*

$$\tilde{c}_i = \left(\mathit{\Pi}_{j=1}^{\kappa}\, s_{i,j}^{(h)} v_{h,j} + \neg v_{h,j}\right)\, \left(\mathit{\Pi}_{j=\kappa+1}^{k}\, s_{i,j}^{(h)}\right)\,.$$

Proof. Let us proceed by induction. First assume $\rho = \emptyset$*. Prop. 6.1 shows that the thesis holds for card*$(\gamma) = 1$ *topologies. Supposing that the thesis holds for* n *topologies, i.e., given an event of the form* $c_i = \mathit{\Pi}_{j=1}^{n} s_{i,j}$*, with* s_j *of existence type, the visibility–based event is* $\tilde{c}_i = \left(\mathit{\Pi}_{j=1}^{n}\, s_{i,j}^{(h)} v_{h,j} + \neg v_{h,j}\right)$*, it is necessary to prove it for* $k+1$ *topologies. First consider that*

$$\begin{aligned}c_i = \mathit{\Pi}_{j=1}^{k+1} s_{i,j} &= \underbrace{\left(\mathit{\Pi}_{j=1}^{k} s_{i,j}\right)}_{z}\, s_{i,k+1} = \\ &= z\, s_{i,k+1} = z\, \left(s_{i,k+1}^{(h)} + n_{i,k+1}^{(h)}\right).\end{aligned}$$

As above, consider factorizing the event as follows. If $n_{i,k+1}^{(h)} = 0$*, the event reduces to* $c_i = z\, s_{i,k+1}^{(h)}$*. If* $n_{i,k+1}^{(h)} = 1$*, the event becomes* $c_i = z$*. Therefore, the event can be rewritten as*

$$c_i = z \neg n^{(h)}_{i,k+1} + z\, s^{(h)}_{i,k+1}\,.$$

W.r.t. the observer's visibility on the last topology, $v_{h,k+1}$*, the event can be factorized as follows. If* $v_{h,k+1} = 1$*, it results* $n^{(h)}_{i,k+1} = 0$ *and* $c_i = z\, s^{(h)}_{i,k+1}$*, whereas if* $v_{h,k+1} = 0$*, it results* $c_i = z \neg n^{(h)}_{i,k+1} + z\, s^{(h)}_{i,k+1}$*. This yields*

$$c_i = z\, \left(s^{(h)}_{i,k+1}\, v_{h,k+1} + \left(s^{(h)}_{i,k+1} + \neg\, n^{(h)}_{i,k+1} \right) \neg v_{h,k+1} \right)\,.$$

The visibility–based event is $\tilde{c}_i = \max_{n_{i,j} \in \mathbb{B}, j=1,\cdots,k+1} c_i$*, whose direct computation gives*

$$\max_{n_{i,j} \in \mathbb{B}, j=1,\cdots,k} z$$

$$\max_{n_{i,k+1}} \left(s^{(h)}_{i,k+1} v_{h,k+1} + \left(s^{(h)}_{i,k+1} + \neg n^{(h)}_{i,k+1} \right) \neg v_{h,k+1} \right) =$$

$$\left(\Pi^{k}_{j=1} s^{(h)}_{i,j}\, v_{h,j} + \neg v_{h,j} \right) \left(s^{(h)}_{i,k+1} v_{h,k+1} + \neg v_{h,k+1} \right),$$

from which the part of the thesis relative to existence topology follows.

Consider now the case $\kappa = \emptyset$*. Prop. 6.2 shows that the thesis holds for one topology of such a type. The inductive step requires considering*

$$c_i = \Pi^{k+1}_{j=1} s_{i,j} = z\, s^{(h)}_{i,k+1}\, n^{(h)}_{i,k+1}.$$

This event can be factorized as follows. If $n^{(h)}_{i,k+1} = 0$*, the event reduces to* $c_i = 0$*, whereas, if* $n^{(h)}_{i,k+1} = 1$*, it becomes* $c_i = z\, s^{(h)}_{i,k+1}$*. Therefore, the event can be rewritten as*

$$c_i = z\, s^{(h)}_{i,k+1}\, n^{(h)}_{i,k+1}\,.$$

W.r.t. the observer's visibility on the last atom, $v_{h,k+1}$*, the event can be factorized as follows. If* $v_{h,k+1} = 1$*, it results* $n^{(h)}_{i,k+1} = 1$ *and* $c_i = z\, s^{(h)}_{i,k+1}$*, whereas if* $v_{h,k+1} = 0$*, it results* $c_i = z\, s^{(h)}_{i,k+1}\, n^{(h)}_{i,k+1}$*. This yields*

$$c_i = z \left(v_{h,k+1} + n^{(h)}_{i,k+1} \neg v_{h,k+1} \right) s^{(h)}_{i,k+1} .$$

The visibility–based event is $\tilde{c}_i = \max_{n_{i,j} \in \mathbb{B}, j=1,\cdots,k+1} c_i$*, whose direct computation gives*

$$\begin{gathered} \max_{n_{i,j} \in \mathbb{B}, j=1,\cdots,k} z \\ \max_{n_{i,k+1}} \left(v_{h,k+1} + n^{(h)}_{i,k+1} \neg v_{h,k+1} \right) s^{(h)}_{i,k+1} = \\ = \left(\Pi^{k}_{j=1} s^{(h)}_{i,j} \right) s^{(h)}_{i,k+1} , \end{gathered}$$

from which also the second part of the thesis follows. The result straightforwardly extends to the case with $\rho, \kappa \neq \emptyset$ *(the proof is omitted for space), which concludes the thesis.*

The predictor is initialized with the value $\tilde{\sigma}_i(0) = \Sigma_i$, which corresponds to the most conservative hypothesis on the activation of c_i. The estimated state $\tilde{\sigma}_i$ is updated according to the rule

$$\tilde{\sigma}^{(r)}_i(t_{k+1}) = \tilde{\delta}^{(r)} \left(\tilde{\sigma}^{(r)}_i(t_k), \tilde{e}^{(r)}_i(\tilde{c}_i(t_{k+1})) \right) .$$

The predictor $\tilde{\mathcal{H}}^{(r)}(\tilde{q}, \tilde{\sigma}, \tilde{I})$ of the r–th species can be constructed by using the same decoder $u^{(r)}$, encoder $s^{(r)}$, dynamics $f^{(r)}$, and the other components of the nominal model, and is valid for any visibility $\mathcal{V}_h$ of the classifying agent and any neighborhood configuration set I_i of the target agent. The set of possible behaviors $\tilde{q}^{(r)}_i$ that $\mathcal{A}_i$ can execute compatibly with the r–th species and the local knowledge of $\mathcal{A}_h$ is the solution of

$$\left\{ \tilde{q}^{(r)}_i(t_k) = \bar{q}^{(r)}_i(t_k) . \right. \tag{6.1}$$

Note that, as the cardinality of $\tilde{\sigma}_i(t_k)$ is finite and equal to $\kappa(t_k)$, then $\tilde{q}^{(r)}_i = \{\tilde{q}^{(r)}_{i,1}, \ldots, \tilde{q}^{(r)}_{i,\kappa(t_k)}\}$ can be directly computed. This is iterated for all different

species for which the agent's compatibility has not been excluded yet, based on the assumption that an agent belongs always to the same species with which it is initialized.

The second step, the classification phase, starts with determining for which species there exists at least one behavior that is sufficiently close to the observed one. This can be achieved by evaluating the test

$$\|\bar{q}_i(t) - \pi_{\mathcal{Q}}(\phi_{\tilde{\mathcal{H}}^{(r)}}(\bar{q}_i(t), \bar{\sigma}_i(t_k), I_i^h(t)))\| \leq \epsilon, \quad \forall \quad t \in [t_k, t_{k+1}), \tag{6.2}$$

that returns true if it is satisfied by at least one behavior in $\tilde{q}_i^{(r)}$. If none of the predicted behaviors satisfies the test, for a given species, then $\mathcal{A}_h$ is not compatible according to Def. 6.2 with that species. If, on the contrary, the test is satisfied by some behaviors — denote with $l_1, \ldots, l_m \in \mathbb{N}$ their indices — the agent is possibly compatible with the species. An estimate of agent $\mathcal{A}_i$'s current action is given by $\tilde{\sigma}_i^{(r)}(t_k) \leftarrow \{\tilde{\sigma}_{i,l_1}^{(r)}, \ldots, \tilde{\sigma}_{i,l_m}^{(r)}\}$. Finally, this allows computing a logical vector $C_i^{(h)} = \left(C_{i,1}^{(h)}, \cdots, C_{i,p}^{(h)}\right)$, where $C_{i,j}^{(h)}$ is true iff agent $\mathcal{A}_i$ is compatible with species S^r, based on local knowledge of $\mathcal{A}_h$. As soon as $C_i^{(h)}$ contains exactly one element set to true, agent $\mathcal{A}_i$ can be classified as belonging to the corresponding species, according to the assumption above. The local classifier is obtained as the series of the prediction system (Eq. 6.1) and the classification test (Eq. 6.2) described above.

6.1.3 Local Classifier Agreement and Distributed Species Estimation

As shown in the previous section, an agent $\mathcal{A}_h$ with partial visibility on the left, right, front, or rear regions of a neighboring agent $\mathcal{A}_i$ is unable to correctly determine the correctness of the estimate on the neighborhood. In this

section it will be shown how the agents can share the information extracted by using their local classifier and can reach an agreement on the species to which agent $\mathcal{A}_i$ belongs. It is assumed that agents can communicate via one–hop links in order to reduce their detection uncertainty and "converge" to a unique network decision. In this respect in the following concepts involving procedures and algorithms aiming to reach an agreement in networks are introduced.

6.1.3.1 Consensus algorithm

Consider a network whose communication topology is represented by the undirected graph G which is composed by a set of nodes $v = \{v_1, \cdots, v_n\}$ linked by edges s.t. an edge $G_{i,j}$ means that the node v_i is able to communicate with node v_j, i.e. $\mathcal{A}_i \in N_j = \{k | e_{k,j} \in G\}$ being N_j the neighbor set of the agent $\mathcal{A}_j$.

Now recall the following

Definition 6.4. Given a graph G, a *consensus algorithm* is an iterative interaction rule that specifies how each node v_i updates its estimates of the generic information $s \in S$ shared among neighbors based on any received value v_j, i.e. it specifies the function $\xi : S \times S \to S$ which is used to compute

$$s_i^+ = \xi(s_i, s_j), \qquad \text{for } i, j = 1, \cdots, n.$$

Typical consensus algorithms involving scalar values are obtained through very simple updating rules such as weighted average, or maximum occurrence value. In more general cases the quantities of interest could be possibly non-convex sets, intervals, or logical values. Motivated by this fact, it is necessary to involve a more general class of consensus algorithms so as to permit agents sharing locally collected information and eventually "converge" to a unique network decision. Referring to our scenario, nodes are robots that are monitor-

ing a common neighbor and that are supposed to communicate as in G in order to reach an agreement on the species of such neighbor.

Consider the matrix

$$C_i(t_k) = \begin{bmatrix} C_i^{(1)}(t_k) \\ C_i^{(2)}(t_k) \\ \vdots \\ C_i^{(n)}(t_k) \end{bmatrix} \tag{6.3}$$

which is obtained by using all monitors' logical classification vectors introduced in Section 6.1.2 after t_k steps of the consensus iterative procedure. Our aim is to design a distributed consensus algorithm allowing us to have $C_i(\infty) = \mathbf{1}^* C_i$, where ${}^*C_i$ is the *centralized classification vector* defined as the vector that would be constructed by a monitor collecting all initial measures and combining them according to ξ.

A possible solution allowing us to reach an agreement consists in letting agents to share the locally estimated *encoder map* s_i. In other words, this thesis provide a solution where agents share any information that is directly measured or reconstructed by inspecting its neighborhood through logical consensus [60]. After having established an agreement for the value of the encoder map for a generic agent, they will use the same decision rule and hence decide for the same classification vector.

6.1.3.2 Logical Consensus as a framework for classification

The classification problem is recast as a control problem requiring the computation of a set of ν_i *decisions* $y = c_i = (c_{i,1}, \cdots, c_{i,\nu_i}) \in \mathbb{B}^{1\times\nu_i}$ that depend on κ_i logical values $U = (u_1, \cdots, u_{\kappa_i}) \in \mathbb{B}^{1\times\kappa_i}$. As introduced in Chapter 5, one can imagine that each agent stores the values of all events into a state vector $X_h = (x_{h,1}, \cdots, x_{h,\kappa_i}) \in \mathbb{B}^{1\times\kappa_i}$, that is a string of bits, and it is assumed

that the matrix $X(t) = (X_1(t)^T, \cdots, X_n(t)^T)^T$ represents the network state at the time t. In practice, each agent become a dynamic node that updates its local state X_h through a distributed logical update function F that depends on its state, on the state of its neighbors and on the observed inputs which is used to initialize the value of the state, i.e. $X_h(t+1) = F_h(X(t))$. Moreover, it is assumed that every agent is able to produce the logical output decision vector $Y = (Y_1(t)^T, \cdots, Y_n(t)^T)^T \in \mathbb{B}^{n \times \nu_i}$ by using an output function D depending on the local state, i.e. $Y_h = D_h(X_h) = c_i^{(h)} = (c_{i,1}^{(h)}, \cdots, c_{i,\nu_i}^{(h)})$.

Referring to our scenario, such event $u_j = s_{i,j}$, $j \in \{1, \cdots, \kappa_i\}$ may represent the presence of an agent in the corresponding topology η_{ij}, and each output decision $y_{i,i}$ may represent the detector condition $c_{i,i}$. Furthermore, as introduced before, since situations where agents may be *heterogeneous* in terms of sensors and communication devices as well as the fact that agents are placed at different locations w.r.t. the target agent are considered, the generic agent h may or may not be able to measure the value of $u_j = s_{i,j}$. In this sense, we can conveniently introduce a *visibility matrix* $V \in \mathbb{B}^{n \times \kappa_i}$ such that it results $V_{h,j} = 1$ if, and only if, agent $\mathcal{A}_h$ is able to measure $s_{i,j}$. Moreover, each agent is able to communicate only with a subset of other agents. Therefore, to effectively accomplish the given decision task, it is necessary that such an information *flows* from one agent to another, consistently with available communication paths.

In other words, for any given combination of input values, it is considered a task that requires computation of the following system of logical functions:

$$\begin{cases} X(t+1) = F(X(t)) \\ X_h(0) \quad\;\; = \tilde{U}^{(h)} \\ Y(t) \quad\;\; = D(X(t)) \end{cases} \tag{6.4}$$

where $F : \mathbb{B}^{\kappa_i} \to \mathbb{B}^{\kappa_i}$, $D = (d, \cdots, d)$ with $d : \mathbb{B}^{\kappa_i} \to \mathbb{B}^{\nu_i}$, and $\tilde{U}^{(h)} : \mathcal{Q} \to \mathbb{B}_i^{\kappa}$ represents a lower approximation of $s_i = (s_{i,1}, \cdots, s_{i,\kappa_i})$ based only on observation of the agent's neighborhood operated by the h–th agent.

The following results can be stated

Theorem 6.2. *Given a connected communication graph G, n initial estimates $X(0) = (\tilde{s}_i^{(1)T}, \cdots, \tilde{s}_i^{(n)T})^T$, and a visibility matrix V with non–null columns, the distributed logical consensus system*

$$\begin{cases} x_{h,j}^{+} = V_{h,j}\, x_{h,j} + \neg V_{h,j} \left(\sum_{k \in N_h} x_{k,j} \right) , \\ x_{h,j}(0) = \tilde{s}_{i,j}^{(h)} , \end{cases} \tag{6.5}$$

with $h = 1, \cdots, n$, $j = 1, \cdots, \kappa_i$ converges in at most diam(G) steps to the consensus state $X = \mathbf{1}_n\, s_i$.

Proof. To prove the proposition, consider factorizing the update rule as follows. If $V_{h,j} = 0$, agent h is unable to autonomously compute $s_{i,j}$. In this case Eq. 6.5 reduces to

$$x_{h,j}^{+} = \neg V_{h,j} \left(\sum_{k \in N_h} x_{k,j} \right) = \sum_{k \in N_h} x_{k,j} \, .$$

Moreover, since column $V_{j,:}$ is non–null by hypothesis, there is at least an agent, say the q–th, with complete visibility on the j–th topology $\eta_{i,j}$, which implies $\tilde{s}_{i,j}^{(q)} = s_{i,j}$. Observe that it must also holds that $\tilde{s}_{i,j}^{(q)} \geq \tilde{s}_{i,j}^{(h)}$ for every h. Since G is connected, the real value $s_{i,j}$ is propagated from agent q to the rest of the network, which implies that there exists a time $\bar{N} < \infty$ after which

$$x_{h,j} = \sum_{k=1}^{n} x_{k,j} = \tilde{s}_{i,j}^{(q)} = s_{i,j} \, .$$

If instead $V_{h,j} = 1$, agent h has complete knowledge of the j–th topology $\eta_{i,j}$ and its update rule (Eq. 6.5) specializes to

$$x_{h,j}^{+} = V_{h,j}\, x_{h,j} = x_{h,j}\,,$$

and its initial estimate is $\tilde{s}_{i,j}^{(h)} = s_{i,j}$. It trivially holds that $x_{h,j} = s_{i,j}$, which proves the theorem.

Theorem 6.3. *Given a connected communication graph G, a visibility matrix V with non-null columns, an output function $D = (d, \cdots, d)$ s.t. $d(s_i) = c_i$, the distributed logical consensus system (6.4) where F is given by (6.5) solves Problem 6.1.*

Proof. To prove the theorem it is necessary to prove that the matrix (6.3) is such that $C_i^{(h)}(\bar{N}) = \mathbf{1}^* C_i$ with $\bar{N} < \infty, \quad h = 1, \cdots, n$. With Theorem 6.2 it has been proved that $X(\bar{N}) = \mathbf{1}_n\, s_i$. This means that $D_h(X_h) = d(s_i) = c_i$ and the predictor ${}^h\tilde{\mathcal{H}}_i^{(r)}({}^h\tilde{q}_i, {}^h\tilde{\sigma}_i, {}^h\tilde{I}_i),\ h = 1, \cdots, n$, of the r–th species is initialized with the value ${}^h\tilde{\sigma}_i(0) = {}^h\Sigma_i$ which corresponds to the most conservative hypothesis on the activation of C_i which is the same for all monitoring agents, i.e. ${}^h\tilde{\sigma}_i(0) = {}^*\tilde{\sigma}_i(0), \quad h = 1, \cdots, n$ where ${}^*\tilde{\sigma}_i(0)$ is the value of $\tilde{\sigma}_i(0)$ in the case that $\tilde{c}_i = c_i$. Thus, the estimated state ${}^h\tilde{\sigma}_i, h = 1, \cdots, n$ becomes

$$\begin{aligned} {}^h\tilde{\sigma}_i^r(t_{k+1}) &= \tilde{\delta}^r({}^h\tilde{\sigma}_i^r(t_k), \tilde{e}_i^{(r)}({}^h\tilde{c}_i(t_{k+1}))) \\ &= \tilde{\delta}^r({}^*\tilde{\sigma}_i^r(t_k), \tilde{e}_i^{(r)}(c_i(t_{k+1}))) = {}^*\tilde{\sigma}_i^r(t_{k+1}) \end{aligned}$$

According to this, it results

$$\begin{aligned} \|\bar{q}_i(t) - \pi_{\mathcal{Q}}(\phi_{{}^h\tilde{\mathcal{H}}^{(r)}}(\bar{q}_i(t), {}^h\bar{\sigma}_i(t_k), I_i^h(t)))\| = \alpha_r, \\ \forall \quad t \in [t_k, t_{k+1})\ , h = 1, \cdots, n \end{aligned} \tag{6.6}$$

with $\alpha_r \in R$.

If $\alpha_r < \varepsilon$ it results that $\mathcal{A}_i$ is compatible with the r–th species, i.e. $C_{i,r}^{(h)} = 1,\ h = 1, \cdots, n$. By iterating this procedure for all species, it follows that $C_i^{(h)} = (C_{i,1}^{(h)}, \cdots, C_{i,p}^{(h)}) = \mathbf{1}^* C_i,\ h = 1, \cdots, n$, which proves the theorem.

6.1.4 Application of the Distributed Classification System

Effectiveness of the proposed distributed classifier is shown through application to a biologically–inspired system and a robotic multi agent, where the very same procedure has been applied. The reader may refer to the site `http://www.youtube.com/watch?v=KXNuBWoLctQ` for the complete simulation run.

6.1.4.1 Ant Classification Example

Consider an example with 5 ants of the Green species and 5 of Red one. The goal of each ant is to recognize its nestmates to cooperate in the foraging process of the colony. Fig. 6.1-a shows the initial situation, where it is shown a classifying ant (green in figure) with insufficient information to classify its neighbors. As the simulation proceeds the ant gathers more information and is able to correctly recognize and recruit nestmates (Fig. 6.1-b).

6.1.4.2 Vehicle Classification in Highways

Consider 5 vehicles in a 3–lane highway (Fig. 6.2–a). In the simulation, an SOS vehicle (white vehicle in the figure) changes from FAST to LEFT maneuver to overtake another vehicle (purple vehicle in the figure). Moreover, there are three vehicles that are running local classifiers to classify the SOS car. Note

(a)

(b)

Fig. 6.1 An ant of the Green species classifying its neighbors. The simulation starts with no a priori knowledge (a) and concludes with the ant that has correctly classified all other ants (b).

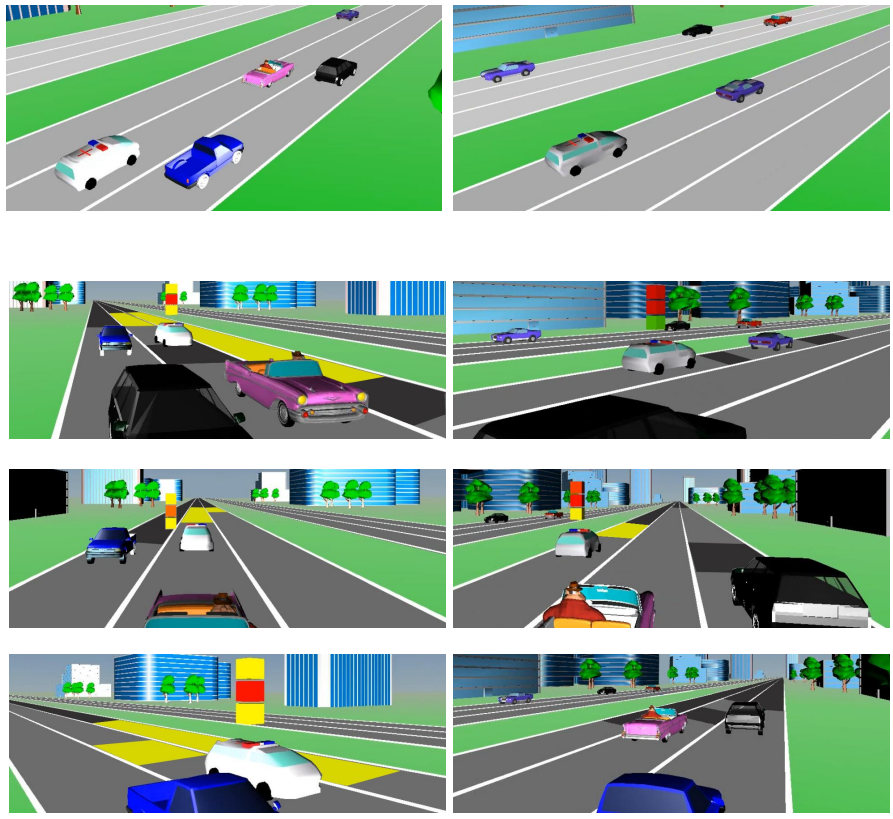
that a FAST to LEFT transition of an agent of right-hand, or emergency traffic rules species implies that its frontal area is busy, while its left area is free (Fig. 3.7). In the example, the classifying vehicles, having with visibility of the influence region the emergency vehicle, are unable to classified it and remain uncertain, but still they can conclude for its compatibility with both species. On the contrary, a FAST to LEFT transition for a left–hand traffic rules vehicle implies that the frontal area is busy, while the left area is free, which is false in the example. Therefore, it is possible to exclude the left–hand species. The classification result is reported in Fig. 6.2–a, where $C_i^{(h)}$ is specified by a flag on the target agent. From top to down, the cells of the flag represents the classification w.r.t. the right-hand, left-hand, and emergency vehicle traffic rules species. Adopted colors are green, yellow, and red for the values compatible,

uncertain, or incompatible, respectively. Dark gray and light gray regions represent $\mathcal{V}_i$ and $\bar{\mathcal{V}}_i$. Yellow regions are regions in $\bar{\mathcal{V}}_i$ that are essential to decide about the classification. For the sake of completeness, consider a successive time in the simulation, when the emergency vehicle is performing a FAST to RIGHT transition to overtake another vehicle (violet vehicle in the figure). The result of the local classification is depicted Fig. 6.2–b. In this case, the target agent is correctly classified as belonging to the emergency traffic rules species by two local classifiers, while a third one is unable to reach this due to its limited visibility. This shows the limit of the proposed technique and represents the motivation for future work, in which local classifiers will be allowed exchanging information and reaching a unique global decision.

6.2 Probabilistic Classification

The above solution's main limitation is related to the need of a priori knowledge of the models describing each possible behavior and interaction. To overcome this limitation, every agent should be endowed with the ability to autonomously and iteratively learn the different possible behaviors and interactions by using local observation of its neighbors.

Furthermore, the problem of learning behaviors through demonstrated action sequences from a human performer, a.k.a. apprenticeship learning [31, 32, 33], has been largely studied in the literature. A Mimesis–based approach is presented in [33] allowing abstraction of observed behaviors as symbols, to consequently recognize others' behavior and generate motion patterns through the symbols themselves. These solutions are mainly based on learning the parameters of a statistical model — represented e.g. by a Hidden Markov Model (HMM) —, but the scope of these approaches is limited to achieving an accurate imitation of the performer's actions by the robotic system. In the following, a solution to the classification problem is proposed, that is applicable



(a) (b)

Fig. 6.2 Snapshots from a simulation of the highway example with vehicles belonging to the right-hand, left-hand, and emergency vehicle traffic rules species.

for societies of robots, where agents have unknown behaviors and interaction forms. It is assumed that agents displaying different behaviors, due to their dynamics or to the set of rules they obey to, belong to a different species. The solution, partially built upon the results on model learning, consists of a procedure that each agent can run to iteratively grow a knowledge base of all

possible species that are present in the "society". More precisely, each "estimated" species is represented by a Hidden Markov Model (HMM), whose structure and parameters are iteratively learnt based on the actually measured trajectory of the target agent. The aim of the learning phase is to obtain a sufficiently accurate HMM that approximates the nominal behavior of the agent. By assuming that a sufficiently rich observation of every agent's behavior is available, a classification procedure allowing agents to classify its neighbors will be proposed.

6.2.1 Problem Statement

Consider a "society" composed of n heterogeneous agents belonging to different species and sharing an environment $\mathcal{Q}$. As described in Chapter 3, each *species* consists of a motion model, a model of neighborhood, a set of interaction rules, and local low–level controllers for every agent. More precisely, a generic agent $\mathcal{A}_i$ is member of a species S^r is described by a state $(q_i, \sigma_i) \in \mathcal{Q} \times \Sigma_i$, where Σ_i is a set of possible symbols, and evolves according to the hybrid model

$$\begin{cases} (\dot{q}_i(t), \sigma_i(t_{k+1})) = \mathcal{H}_i^r(q_i(t), \sigma_i(t_k), I_i(t)) \,, \\ (q_i(0), \sigma_i(0)) = (q_i^0, \sigma_i^0) \,, \end{cases} \tag{6.7}$$

where $\mathcal{H}_i^r : \mathcal{Q} \times \Sigma_i \times \mathcal{Q}^{n_i} \to T_{\mathcal{Q}} \times \Sigma_i$, n_i is the current number of neighbors, $T_{\mathcal{Q}}$ is space tangent to $\mathcal{Q}$, and I_i is its neighbor configuration set. Accordingly, the agent's *behavior* is the solution

$$(q_i(t), \sigma_i(t_k)) = \phi_{\mathcal{H}_i^r}(q_i^0, \sigma_i^0, \tilde{I}_i(t))$$

of Eq. 6.7, where $\tilde{I}_i(t)$ is the history of its neighbor configuration set $I_i(\tau)$ for $\tau = 0, \cdots, t$. Note that the behavior of the agent depends on logical conditions on its neighborhood, e.g. on the presence or absence of other neighboring agents.

Definition 6.5 (Species Membership). Given a metric $Z : \Sigma \times \Sigma \to \mathbb{R}$ and an accuracy ε, we say that two agents, $\mathcal{A}_i$ and $\mathcal{A}_j$, are *member* of the same species, if there exist two initial discrete states $\hat{\sigma}_i^0, \hat{\sigma}_j^0 \in \Sigma$ s.t.

$$Z(\hat{\sigma}_i(t), \hat{\sigma}_j(t)) \leq \varepsilon \,,$$

where $\hat{\sigma}_k(t)$, for $k = i, j$, is the discrete state sequence solving the agents' dynamics. ♦

We want to solve the following:

Problem 6.2 (Classification Problem). Given the behaviors $\bar{y}_i = q_i$, for $i = 1, \cdots n$, measured by an observer, a metric $Z : \Sigma \times \Sigma \to \mathbb{R}$ and an accuracy ε, we want to classify every agent, i.e. to distinguish if any two agents are member of the same species w.r.t. the metric Z.

6.2.2 Proposed Solution

Consider an agent $\mathcal{A}_h$ (later referred to as the *observer*) trying to learn to which species another neighboring agent $\mathcal{A}_i$ is member of. We assume that agents are dynamical systems evolving according to the model in Eq. (6.7), but no information is available for $\mathcal{A}_h$ concerning the map $\mathcal{H}_i^r$. Therefore, an approximation of the model of each species needs to be iteratively constructed based on observation. In what follows we assume that the neighbor configuration set I_i is known to the classifier and thus we will omit it for simplicity. As a consequence, the behavior of $\mathcal{A}_i$ depends only on its internal state q_i. Moreover,

Algorithm 4: Algorithm for Probabilistic Classification

Input: $q \in \mathcal{Q}^n$
Input: ε
Output: $\tilde{S}$
Output: $C \in \mathbb{N}^n$

$k \leftarrow 1$;
$\tilde{S} \leftarrow \emptyset$;
while $k \leq n$ **do**
  $\tilde{\sigma} \leftarrow \Phi(q_k)$; ◁ `Feature Extraction`
  $(\bar{S}, \bar{p}) \leftarrow \zeta(\tilde{\sigma}_k)$;
  if $\bar{p} < \varepsilon$ **then**
    $\bar{S} \leftarrow \#\tilde{S} + 1$;
    Learning of a new species;
  end
  $C_k \leftarrow \bar{S}$;
  $k \leftarrow k + 1$;
end

we assume that $\mathcal{A}_h$ is able to store a set $\tilde{S}$ of models summarizing the different species which have been observed so far (this set is empty at the beginning). More precisely, each "estimated" species is represented by a *Hidden Markov Models* (HMMs) [90], whose structure and parameters are iteratively learnt based on the actually measured trajectory $\tilde{q}_i$. The aim of the learning phase is to obtain a sufficiently accurate HMM that approximates the nominal map $\mathcal{H}_i^r$.

Let us denote with $\tilde{\Xi}$ a (possibly infinite) set of *features* that characterize each species. One needs to introduce a map $\Phi : \mathcal{Q} \to \tilde{\Xi}$ assigning every agent configuration q_i with a feature in Ξ, i.e.

$$\xi_i(t) = \Phi(q_i(t)) \,.$$

Given the trajectory $\tilde{q}_i(t)$ of a generic agent $\mathcal{A}_i$, denote with $\tilde{\xi}_i(t_k)$ the corresponding discrete–time feature sequence obtained via the map Φ. Moreover, we need the following:

Definition 6.6 (Level of Confidence). We say that an agent with behavior $\tilde{q}_i$ is *member* of the approximated species $\tilde{S}^r$ with *level of confidence* l if the probability p_i^r that $\tilde{q}_i$ is generated by S^r equals l. ♦

Denote with $\zeta : \mathcal{Q} \to [1, \cdots, s] \times [0, 1]$ a function returning the index of the approximated species with maximum membership and the corresponding level of confidence, i.e.

$$\begin{aligned} \zeta : \tilde{\Xi} &\to [1, \cdots, s] \times [0, 1] \\ \tilde{\xi}_i &\mapsto (\arg\max_r p_i^r, \max_r p_i^r) \end{aligned} .$$

If there exists a pair $(\bar{S}, \bar{p}) = \zeta(\tilde{q}_i)$ with $\bar{p} > \varepsilon$, agent $\mathcal{A}_i$ is said to be member of $\bar{S}$ (see Def. 6.6) and the corresponding model $\bar{S}$ is enriched with the new data relative to $\tilde{q}_i$. If, on the contrary no existing model is able to described the observed behavior $\tilde{q}_i$ with sufficient accuracy, another approximated model is created and added to the knowledge base of the observer $\mathcal{A}_h$ (See Algorithm 4 for a description of this process). The above described procedure represents the general framework by which Problem 6.2 can be solved. In the following, the process is specialized for one possible instance of the classification problem where the features are extracted with the objective to classify agents that are members of different species due to different rules that characterize the interaction protocol they are obeying.

As stated above, a HMM is a statistical model where states are "hidden" to the observer. Every state $X_i \in X$, where $X = \{X_1, X_2, \cdots, X_n\}$ has a *transition probability* $t_{i,j}$ to move towards the state X_j and a probability $e_{i,j}$ to emit the output symbol $Y_j \in Y$, where $Y = \{Y_1, Y_2, \cdots, Y_m\}$. More precisely, we have

$$t_{i,j} = P(X(t+1) = X_j | X(t) = X_i)$$
$$e_{j,k} = P(Y(t) = Y_k | X(t) = X_j)\,.$$

A HMM is fully described by the triple $\lambda = (\pi, T, E)$, where $\pi \in \mathbb{R}^{1\times n}$ represents the initial state probabilities, $T = \{t_{i,j}\} \in \mathbb{R}^{n\times n}$ and $E = \{e_{i,j}\} \in \mathbb{R}^{n\times m}$ are the transition and output probabilities matrices, respectively.

As specified in Algorithm 4, the (possibly) continuous neighborhood of $\mathcal{A}_i$ is discretized as a finite number of states or *locations*, in which we suppose that the agent may display different types of behaviors. Based on this discretization, a *feature extraction map* $\Phi : \mathcal{Q} \to \tilde{\Sigma}$, where $\tilde{\Sigma}$ a finite set, is defined, which returns a symbol representing its current location for every agent's configuration, i.e.

$$\sigma_i(t) = \Phi(q_i(t))\,.$$

Once a sequence $\tilde{\sigma} \in \mathbb{R}^{1\times z}$ has been observed, the sequence itself is divided into s subsequences $\tilde{\sigma}_j \in \mathbb{R}^{1\times\kappa}$ with $\kappa \leq z$ and $z = s\kappa$, where κ is a parameter whose choice depends both on the value of ε and on the observation length.

Moreover, to the aim of the classification, in this case p_i^r can be defined as the sum of the probability that each subsequence is generated by the model λ_i, i.e.

$$p_i^r = \sum_j^s P(\tilde{\sigma}_j | \lambda_i)\,,$$

which can be computed by using the so–called *Forward Algorithm* [91] that is briefly reported below. Let the forward variable $\alpha_t(j)$ be the probability of sequence $\tilde{\sigma}_i(t) = (\tilde{\sigma}_i(1), \cdots, \tilde{\sigma}_i(k))$ to be in state X_j at time t, i.e.

$$\alpha_j(t) = P(\tilde{\sigma}_i(1), \cdots, \tilde{\sigma}_i(t) | X(t) = X_j). \tag{6.8}$$

The value of the variable is initialized as

$$\alpha_j(1) = \pi_j P(\tilde{\sigma}_i(1) \,|\, X(1) = X_j), \qquad \begin{array}{c} \pi_j = P(X(1) = X_j) \\ 1 \leq j \leq n \end{array}$$

and is updated according to the rule

$$\alpha_j(t+1) = \left(\sum_{p=1}^{n} \alpha_p(t)\, t_{p,j} \right) P(\tilde{\sigma}_i(t+1) | X(t+1) = X_j)$$

with $1 \leq j \leq n$ and $1 \leq t \leq k-1$. Thus, the probability of occurrence of a particular sequence $\tilde{\sigma}_i$ is given by:

$$P(\tilde{\sigma}_i|\lambda) = \sum_{p=1}^{n} \alpha_p(k).$$

If the maximum value of p_i^r for all r (say $\bar{p}_{MAX}$) is greater than the threshold ε, the agent is member of the associated species and the corresponding model is updated. Otherwise a new approximated model for that species is created by using the Baum–Welch Algorithm [92]. The algorithm aims at finding

$$\hat{\lambda} = \underset{\lambda}{\operatorname{argmax}}\, P(\tilde{Y}|\lambda)\,.$$

Firstly the parameters $\lambda = (\pi, T, E)$ has to be initialized, then by using λ and the observed sequence $\tilde{\sigma}$, a new model $\hat{\lambda}$ is computed. Let ϵ an arbitrarily small positive quantity, the algorithm repeats these steps until

$$\log P(\tilde{\sigma}|\hat{\lambda}) - \log P(\tilde{\sigma}|\lambda) \leq \epsilon.$$

To approximate the new species with a model $\hat{\lambda}$, the transition probability between states ($\hat{t}_{i,j}$) and the emission probability ($\hat{e}_{j,k}$) need to be estimated. To this aim, let $\beta_i(t)$ be the backward variable defined as

$$\beta_i(t) = P(\tilde{\sigma}(t+1), \tilde{\sigma}(t+2), \cdots, \tilde{\sigma}(k) | X(t) = X_i, \hat{\lambda}).$$

Furthermore, let $\xi_{i,j}(t)$ be the probability for the model $\hat{\lambda}$ to be in state X_i at time t and in state X_j at time $t+1$ given the observed sequence $\tilde{\sigma}$.

$$\begin{aligned}\xi_{i,j}(t) &= P(X(t) = X_i, X(t+1) = X_j | \tilde{\sigma}, \hat{\lambda}) \\ &= \frac{\alpha_i(t)\, t_{i,j} P(Y(t+1) | X(t+1) = X_j)\, \beta_j(t+1)}{P(\tilde{\sigma} | \hat{\lambda})}\end{aligned}$$

Thus, by letting $\gamma_i(t)$ be the probability of the model $\hat{\lambda}$ to be in the state X_i at time t given the observed sequence $\tilde{Y}$ and model, we have

$$\gamma_i(t) = P(X(t) = X_i | \tilde{\sigma}, \hat{\lambda}) = \sum_{j=1}^{n} \xi_{i,j}(t)$$

Moreover, by arbitrarily initializing parameters we have

$$\begin{aligned}\hat{\pi} &= \gamma_i(1) \\ \hat{t}_{i,j} &= \frac{\sum_{t=1}^{k-1} \xi_{i,j}(t)}{\sum_{t=1}^{k-1} \gamma_i(t)} \\ \hat{e}_{j,k} &= \frac{\sum_{t=1}^{k-1} \gamma_j(t)}{\delta_{\tilde{\sigma}(t)=k}} \sum_{t=1}^{k-1} \gamma_j(t)\end{aligned}$$

where

$$\delta_{\tilde{\sigma}(t)=k} = \begin{cases} 1 \ \text{ if } \tilde{\sigma}(t) = Y_k \\ 0 \ \text{ otherwise} \end{cases}$$

6.2.3 Application

6.2.3.1 Probabilistic Classification for Ant Colonies

Recall the ant society introduced in Chapter 3, and consider an example with 4 ants belonging to the Green species and 4 ants belonging to the Red one. Suppose that a new ant belonging to the green species entering the system aims at classifying with a confidence ε the other ants in order to recognize its nestmates so as to cooperate in the foraging process of the colony. To this aim, suppose that the classifying ant is able to measure the other ant $\mathcal{A}_i$'s configuration, i.e. $q_i = (x_i, y_i, \theta_i, \dot{x}_i, \dot{y}_i)$ for sufficiently large time T (Fig. 6.3 reports the observed trajectory).

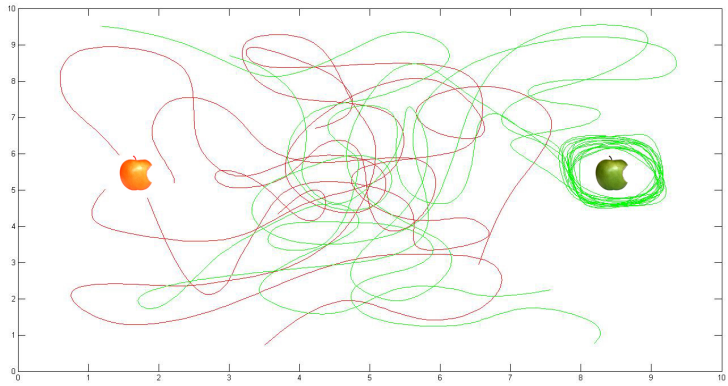


Fig. 6.3 Trajectory of the ants.

According to the procedure described in Algorithm 4, we need to choose a function Φ in order to map the motion of the ants into a discrete sequence of symbols $\tilde{\sigma}(t) \in \tilde{\Sigma} = \Phi(q_i(t))$. In this case we can choose Φ such that $\tilde{\Sigma} = \{\sigma_1, \cdots, \sigma_9\}$ with $\tilde{\sigma}_i(t)$ depending on the orientation of the motion of ant at time t w.r.t. world reference frame.

For example, in the case that the pray is found, a green ant moves around it and the most probable output symbols are $\tilde{\sigma} = \{\sigma_1, ..., \sigma_3, ..., \sigma_7, ..., \sigma_9, \}$, while since a red ant stop in front of it, the most probable observed symbol is σ_5. Afterwards, by applying the forward algorithm to all the HMM whose descriptors are present in the memory, it is possible to evaluate the level of confidence according to Def. 6.6. Fig. 6.4 reports the result of the classification of the eight ants. In particular, since at the initial time, no HMMs representing any species are present, the first ant is classified as belonging to first species and its observed behavior is used to estimate a HMM for the species. Then, the behavior of the ant 2 is compared with the existing model and it results as compatible. Since the behavior of the ant 3 is too different w.r.t. the estimated HMMs of the available species, a new species is created. The ant 4 can be correctly classified as belonging to the green species and so on.

6.2.3.2 Probabilistic Vehicle Classification

Recall the society introduced in Chapter 3 composed of vehicles traveling along a highway with m lanes, and consider now an example where 3 vehicles following the RH driving rules, 2 vehicles following the LH driving rules, and one emergency vehicle are sharing the highways. Suppose that a new vehicle entering the system aims at classifying neighboring agents within a confidence ε so as to improve safety for the entire system. To this aim suppose that the classifying vehicle is able to measure the other vehicles $\mathcal{A}_i$'s configuration vector, i.e. $q_i = (x_i, y_i, \theta_i,)$ for a sufficiently large time T.

According to the procedure described in Algorithm 4, we need to choose a function Φ in order to map the motion of the vehicles into a discrete sequence of symbols $\tilde{\sigma}(t) \in \tilde{\Sigma} = \Phi(q_i(t))$. As before, we can choose Φ such that $\tilde{\Sigma} = \{\sigma_1, \cdots, \sigma_9\}$ with $\tilde{\sigma}(t)$ depending on the orientation of the motion of vehicles at time t w.r.t. world reference frame.

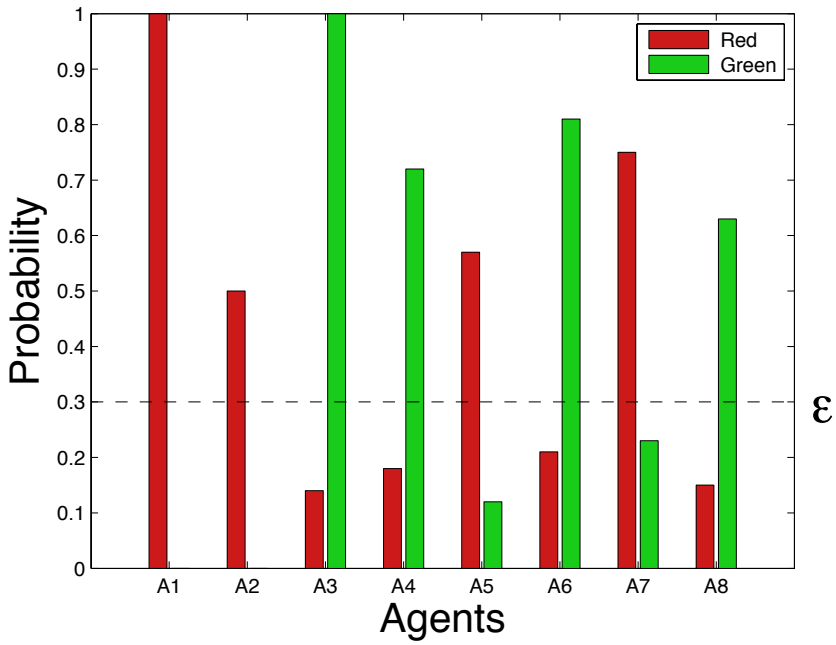
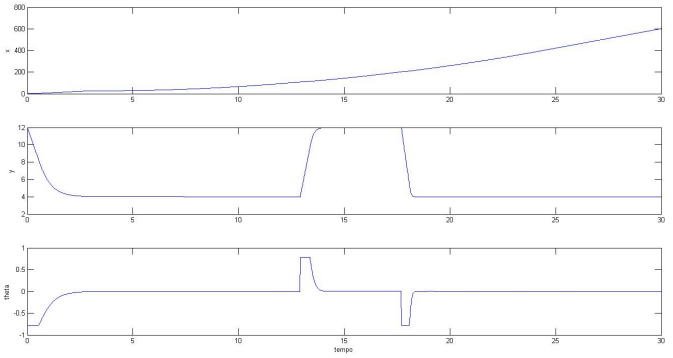


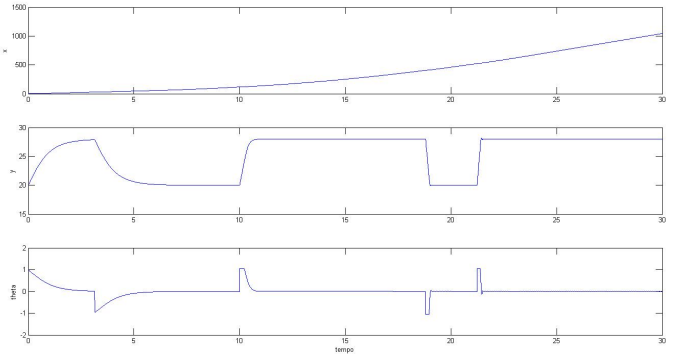
Fig. 6.4 Results of Ants Classification.

Figure 6.4 reports the result of the classification of the vehicles. In particular, since at the initial time no HMMs representing species are present, the behavior of the first observed vehicle is used to estimate a new HMM for the first species. Then, the behavior of the vehicle 2, is compared with the existing model and it results as not consistent. For this reason a new species is created and the behavior of the vehicle 2 is used to estimate the HMM parameters. Then, vehicle 3 is classified as belonging to the first species and so on. Since the behavior of the fifth vehicle is too different w.r.t. the estimated models of the species, a new species is created.

It is worth noticing, that a left overtaking for the RH species and a right overtaking after coming back to the rightmost lane for the LH species (see Fig.6.2.3.2,6.2.3.2) are very similar if we do not consider the neighborhood



(a)



(b)

Fig. 6.5 Two observed data (x, y, θ) belonging to two different vehicle classified to the same HMM.

conditions that according to the cooperation protocol defined in Chapter 3 influence the behavior of agents. For this reason, in order to be able to reconstruct the cooperation model from the observed behavior, we need to tackle the problem with a different approach which will be presented in the remainder of the chapter.

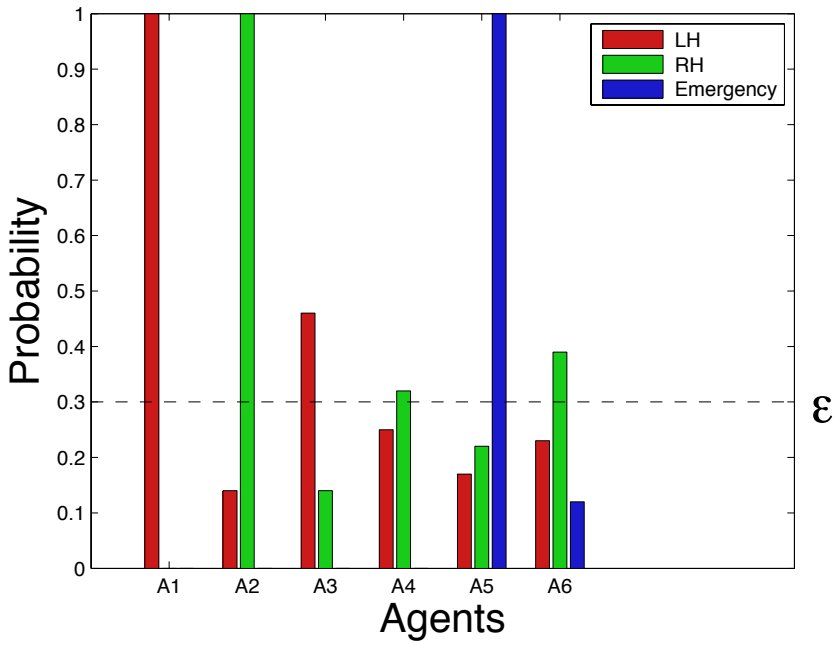


Fig. 6.6 Results of Vehicle Classification.

6.3 Representation Learning of Logical Maps

Recalling the cooperation model as in Chapter 3, the assumption of having complete knowledge of every agent's dynamics is too restrictive, for large and open distributed systems, where agents join in or leave the system dynamically. Upon entering the system, a new agent may need to *learn* the others' current interaction structure. In this context, our identification problem becomes that of estimating the structure of the decision function f_i, on the basis of the measured correspondence over time between its inputs $\bar{u}_{i,1}(t_k), \cdots, \bar{u}_{i,\kappa_i}(t_k)$ and output $\bar{y}_i(t_k)$.

A naive approach to solve the problem is to iteratively store the entries of the Boolean function's truth table as they are observed. Upon recognition of

an already measured input–output pair, the observer can use such a table to obtain an estimate of the value of the logical function. This solution has the main drawback that, if the truth table is represented as a string of the input configurations for which the observed function has returned 1, its size increases with time. As a consequence, complexity reduction techniques must be used, such as the well–known Karnaugh mapping [93] and Quine–McCluskey algorithm [94]. The former is efficient for use in computer algorithms, but it is applicable only when dealing with less than five variables; the latter is more general, but it has still a limited range of use since the problem it solves is NP–hard. Hence, none of them are suitable for online identification.

The proposed approach is based on a canonical form of representation, so–called *Algebraic Normal Form* (ANF), for Boolean functions [34]. According to this representation, every binary function can be specified as a unique combination of the value of a finite set of binary coefficients. Two algorithms for dynamic estimation of such coefficients are provided: the former is valid when the truth table of the Boolean function is completely available, while the latter allows the coefficients to be iteratively updated as the entries of the truth table are observed. Both algorithms are formalized as binary dynamic systems and their convergence toward the desired ANF coefficients are studied by using tools from cellular automata theory [67, 95]. The advantages of the algorithms are that the length of the ANF coefficient vector is constant, and that its update rule involves only logical operations, which can be quickly computed even on low–cost computers.

6.3.1 Boolean Polynomial Representation

Recalling from [34] the following proposition on the representation of a Boolean map:

Proposition 6.3 (Algebraic Normal Form (ANF)). *A multivariate polynomial of n arguments $u_1, \cdots, u_n$ over $\mathbb{B}$ has a unique representation as disjunctive sum of monomials, i.e.,*

$$f(u_1, \cdots, u_n) = \oplus_{H \in C_n} a_H \, \Pi_{h \in H} \, u_h \,,$$

where C_n is the set of all combinations of indices that can be constructed with at most n elements from the set $\{1, \cdots, n\}$, and where $a_H \in \mathbb{B}$ are uniquely determined. More explicitly, the ANF of a Boolean function $f : \mathbb{B}^n \to \mathbb{B}$ is given by

$$\begin{aligned} f(u_1, \cdots, u_n) = a_0 \oplus \\ \oplus \, a_1 \, u_1 \oplus \cdots \oplus a_n \, u_n \oplus \\ \oplus \, a_{1,2} \, u_1 \, u_2 \oplus \cdots \oplus a_{n-1,n} \, u_{n-1} \, u_n \oplus \\ \vdots \\ \oplus \, a_{1,2,\cdots,n} \, u_1 \, u_2 \cdots u_n \,, \end{aligned}$$

with coefficients $a_{i_1,\cdots,i_k} \in \mathbb{B}$.

An operative definition of the ANF of a Boolean function $f : \mathbb{B}^n \to \mathbb{B}$ is obtained through the recursive formula:

$$\begin{aligned} f(u_1, u_2, \cdots, u_n) = \\ = g(u_2, \cdots, u_n) \oplus u_1 \, h(u_2, \cdots, u_n) \,, \end{aligned} \tag{6.9}$$

where

$$g(u_2, \cdots, u_n) = f(0, u_2, \cdots, u_n) \,,$$

$$h(u_2, \cdots, u_n) = f(0, u_2, \cdots, u_n) \oplus f(1, u_2, \cdots, u_n) \,.$$

Example 6.1. Consider the following logical function

$$\begin{aligned} f : \mathbb{B}^3 &\to \mathbb{B} \\ (u_1, u_2, u_3) &\mapsto \neg u_1 \neg u_2 + u_1 \neg u_3 \end{aligned} .$$

Direct application of the operative definition in Eq. 6.9 yields

$$\begin{aligned} f(u) = {} & f(0,0,0) \oplus \\ & \\ & \oplus u_1 (f(0,0,0) \oplus f(1,0,0)) \oplus \\ & \oplus u_2 (f(0,0,0) \oplus f(0,1,0)) \oplus \\ & \oplus u_3 (f(0,0,0) \oplus f(0,0,1)) \oplus \\ & \\ & \oplus u_1 u_2 (f(0,0,0) \oplus f(1,0,0) \oplus \\ & \quad \oplus f(0,1,0) \oplus f(1,1,0)) \oplus \\ & \oplus u_1 u_3 (f(0,0,0) \oplus f(1,0,0) \oplus \\ & \quad \oplus f(0,0,1) \oplus f(1,0,1)) \oplus \\ & \oplus u_2 u_3 (f(0,0,0) \oplus f(0,1,0) \oplus \\ & \quad \oplus f(0,0,1) \oplus f(0,1,1)) \oplus \\ & \\ & \oplus u_1 u_2 u_3 (f(0,0,0) \oplus f(0,0,1) \oplus \\ & \quad \oplus f(0,1,0) \oplus f(0,1,1) \oplus f(1,0,0) \oplus \\ & \quad \oplus f(1,0,1) \oplus f(1,1,0) \oplus f(1,1,1)) = \end{aligned}$$

$$= 1 \oplus u_2 \oplus u_1 u_2 \oplus u_1 u_3 \,. \qquad \blacklozenge$$

6.3.2 Iterative Identification of a Logical Function

We first need to present a convenient form for the computation of the ANF coefficients a_i of a Boolean map f. It is indeed possible to provide the following result (the proof is omitted for space reasons):

Proposition 6.4 (ANF Coefficients). *Given a Boolean function $f : \mathbb{B}^n \to \mathbb{B}$, its ANF coefficients can be written as*

$$a_\gamma = \oplus_{H \in C_{h-1}(\gamma)} a_H \oplus \left(\Pi_{p \in \gamma} u_p \, \Pi_{p \in N \setminus \gamma} \neg u_p \right) f(u) \,,$$

for all γ, where $C_h(\gamma)$ is the set of all combinations of indices that can be constructed with at most h elements from the set γ, and $N = \{1, \cdots, n\}$.

Let us exemplify with the following:

Example 6.2 (Cont'd). Consider again the system of Ex. 6.1. The ANF coefficients of f can be obtained as described in Prop. 6.4. Indeed, it results:

$$
\begin{aligned}
a_0 &= f(0,0,0) = 1\,,\\
a_1 &= a_0 \oplus f(1,0,0) = 0\,,\\
a_2 &= a_0 \oplus f(0,1,0) = 1\,,\\
a_3 &= a_0 \oplus f(0,0,1) = 0\,,\\
a_{1,2} &= a_0 \oplus a_1 \oplus a_2 \oplus f(1,1,0) = 1\,,\\
a_{1,3} &= a_0 \oplus a_1 \oplus a_3 \oplus f(1,0,1) = 1\,,\\
a_{2,3} &= a_0 \oplus a_2 \oplus a_3 \oplus f(0,1,1) = 0\,,\\
a_{1,2,3} &= a_0 \oplus a_1 \oplus a_2 \oplus a_3 \oplus a_{1,2} \oplus a_{1,3}\\
&\quad \oplus a_{2,3} \oplus f(1,1,1) = 0\,. \quad \blacklozenge
\end{aligned}
$$

The recursive relation for the ANF coefficients expressed by Prop. 6.4 allows us to find a first solution to the identification problem. This solution is valid when the entire truth table, i.e. the correspondence between input vector u and output value y of the Boolean function, is fully available. In this sense we say that the procedure is *centralized*. This is stated in the following:

Theorem 6.4 (Centralized Estimator). *Given the complete truth table of a Boolean map $f : \mathbb{B}^n \to \mathbb{B}$, the dynamical system with binary state $\hat{a} \in \mathbb{B}^\kappa$, $\kappa = 2^n$, evolving as*

$$
\begin{cases}
\hat{a}_\gamma(t+1) = \oplus_{H \in C_{h-1}(\gamma)} \hat{a}_H(t) \,\oplus\, \left(\Pi_{p\in\gamma}\, u_p\, \Pi_{p\in N\setminus\gamma} \neg u_p \right) f(u)\,,\\
\qquad \hat{a}_\gamma(0) = \hat{a}^0_\gamma\,,
\end{cases}
\tag{6.10}
$$

for all γ, with $\hat{a}^0_\gamma$ the initial estimate of the γ–th coefficient, globally converges to the equilibrium point

$$\bar{a} = (a_0, a_1, \cdots, a_{1,2,\cdots,n}),$$

in at most κ steps.

Proof. The vector $\bar{a}$ of the ANF coefficient of f is an equilibrium and is globally stable. Indeed, the equilibrium condition, $\hat{a} = F(\hat{a})$, where $F : \mathbb{B}^{2^n} \to \mathbb{B}^n$ is the dynamic relation in Eq. 6.10, can be easily verified by using Prop. 6.4. Moreover, the incidence matrix of F has the form

$$B(F(\hat{a})) = \begin{pmatrix} 0 & 0 & \cdots & 0 & 0 \\ \tilde{C}_1 & 0 & \ldots & 0 & 0 \\ \vdots & & & \vdots & \vdots \\ 0 & \cdots & \tilde{C}_{2^n-1} & 0 & 0 \\ 0 & \cdots & 0 & \tilde{C}_{2^n} & 0 \end{pmatrix}, \tag{6.11}$$

where $\tilde{C}_j$ is a block matrix with 1 at the entries indicating the index on which the estimated coefficient a_j depends. Since $B(F(\hat{a}))$ has a strictly lower–block triangular form, by Theorem 4.1, the equilibrium is globally stable. It also follows from the theorem that the convergence time is 2^n.

Now, it is possible to move on to providing a second solution by which the ANF coefficient of a Boolean function f can be *iteratively* estimated. We suppose that an observer can measure the value of input vector $\bar{u}$ and the corresponding value $\bar{y}$ through the function$f$. We say that an input vector sequence $U = \bar{u}(0)\,\bar{u}(1)\cdots\bar{u}(2^n)$ and the corresponding output vector sequence $Y = \bar{y}(0)\,\bar{y}(1)\cdots\bar{y}(2^n)$ are said to be *complete*, if all they cover the entire truth table of f.

We are now ready to provide the main result for the iterative estimator of ANF coefficients:

Theorem 6.5 (Iterative Estimator). *Give an unknown logical function* $f : \mathbb{B}^n \to \mathbb{B}$ *with ANF coefficients given by* $a_0, a_1, \cdots a_{1,2,\cdots,n} \in \mathbb{B}$. *Let* $U = \bar{u}(0)\,\bar{u}(1)\,\cdots\,\bar{u}(2^n)$ *be any complete vector sequence of measured inputs and* $Y = \bar{y}(0)\,\bar{y}(1)\cdots\,\bar{y}(2^n)$ *the corresponding output sequence according to* f. *The dynamical system with binary state* $\hat{a} \in \mathbb{B}^{2^n}$ *and nonlinear logical update rule given by*

$$\begin{cases} \hat{a}_\gamma(t+1) = \hat{a}_\gamma(t) \oplus \neg\hat{y}(t)\, Y(t)\, \Pi_{j \in N \setminus \gamma} \neg U_j(t)\,, \\ \hat{a}_\gamma(0) = 0\,, \end{cases} \tag{6.12}$$

where $\hat{y}(t) = \hat{y}(\hat{a}(t), U_j(t))$ *is the estimated value of the logical function and* $U_j(t)$ *is the t–th measured input vector, converges to the equilibrium state*

$$\bar{a} = (a_0, a_1, \cdots, a_{1,2,\cdots,n})\,,$$

in at most 2^n *steps for any input sequence pair* (U, Y).

Proof. To prove the theorem, let us consider first the case of a decision system depending only on one input, i.e. $y = f(u)$ with $u, y \in \mathbb{B}$ and $f : \mathbb{B} \to \mathbb{B}$. Its ANF is

$$y = a_0 \oplus a_u\, u\,,$$

where $a_0, a_u \in \mathbb{B}$. All possible Boolean functions with one input and the corresponding ANF coefficients are reported in Table 6.2. The ANF iterative estimator in Eq. 6.12 becomes

$$\begin{cases} \hat{a}_0(t+1) = \hat{a}_0(t) \oplus \neg\hat{y}(t)\, y(t)\, \neg u(t)\,, \\ \hat{a}_u(t+1) = \hat{a}_u(t) \oplus \neg\hat{y}(t)\, y(t)\,, \end{cases} \tag{6.13}$$

where $\hat{y}(t) = \hat{a}_0(t) \oplus \hat{a}_u(t)\, u(t)$. By under–sampling the dynamic system in Eq. 6.13 every 2 steps, we obtain

$y = f(u)$	a_0	a_u
$y = 0$	0	0
$y = 1$	1	0
$y = u$	0	1
$y = \bar{u}$	1	1

Table 6.2 ANF Coefficients for one–input decision functions $y = f(u)$.

$$\begin{cases} \hat{a}_0(t+2) = \hat{a}_0(t) \oplus y(t)\,\neg u(t) \oplus y(t+1)\,\neg u(t+1) = \\ \qquad\qquad = \hat{a}_0(t) \oplus a_0(t)\,(\neg u(t) \oplus \neg u(t+1))\,, \\ \hat{a}_u(t+2) = \hat{a}_u(t) \oplus y(t) \oplus\ y(t+1) = \\ \qquad\qquad = \hat{a}_u(t) \oplus a_u(t)\,(u(t) \oplus u(t+1))\,, \end{cases}$$

where y has been replaced with its ANF. The discrete derivative of the Boolean function $F : \mathbb{B}^2 \to \mathbb{B}$ of the under–sampled systems is

$$F' \begin{pmatrix} a_0 \\ \\ a_u \end{pmatrix} = \begin{pmatrix} \neg a_0 & 0 \\ \\ a_u & \neg a_0 \oplus a_u \end{pmatrix},$$

which is a function depending on the ANF coefficients of f. We thus need to discuss all possible combinations of values for a_0 and a_u. By Theorem 4.5, it is possible to prove that the equilibrium point $\hat{a} = (a_0, a_u)$ is attractive in its VNN, if $a_0 = a_u = 0$. In this case we have $\rho(F') = 0$ and every columns of F' contain at most one entry to 1. Since the origin belongs to this region, we can conclude that the estimator in Eq. 6.12 can correctly estimate the ANF coefficients of f in at most 2 steps. For all other cases in which the theorem is not conclusive, it is possible to proceed by explicitly computing the

system's evolution. By doing this, for any possible input sequences (namely $u(0) = 0, u(1) = 1$ and $u(1) = 1, u(0) = 0$), we obtain the unique solution $\hat{a}_0(2) = a_0,\ \hat{a}_u(2) = a_u$.

The general proof involving a Boolean function with n inputs can be obtained by induction. The above discussed case represents the basic induction step, and the convergence proof with $n + 1$ inputs can be reduced to one with n input via the recursive Def. 6.9. The explicit procedure is omitted here for space reasons.

Example 6.3 (One–input Decision Functions). Consider first identifying the decision system $y = f(u) = 1$. If the observed input sequence is $u(0) = 0, u(1) = 1$, the estimator's state evolves as follows

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 0 \end{pmatrix},$$

while, if the observed input sequence $u(0) = 1, u(1) = 0$, we have

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} 0 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 0 \end{pmatrix},$$

In both cases, the identified logical decision function is $\hat{y} = 1 \oplus 0\, u = 1 = f(u)$.

As a second example, consider the negation function $y = f(u) = \neg u$. If the observed input sequence is $u(0) = 0, u(1) = 1$, the estimators' state evolves as follows

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 1 \end{pmatrix},$$

while, if the observed input sequence $u(0) = 1, u(1) = 0$, we have

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} 0 \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 1 \end{pmatrix},$$

which both give $\hat{y} = 1 \oplus 1\, u = \neg u = f(u)$. ⧫

6.3.3 Application to Cooperative Robotic Systems

Consider the distributed system introduced in Section 3.3 where a new car $\mathcal{A}_h$ entering the system willing to identify the interaction structure between the vehicles that are in its neighborhood. We show how the new car $\mathcal{A}_h$ can identify the decision functions of one of its neighbors $\mathcal{A}_i$, by using the iterative estimator of Theorem 6.5. While we assume that $\mathcal{A}_i$ is a standard vehicle, the new car $\mathcal{A}_h$ has no information on this. $\mathcal{A}_h$ can measure the event input vector $\bar{u}$ by using the $\mathcal{A}_i$'s detection map and the corresponding decisions namely $\bar{y}_i$ by using the dynamics and automaton. The observation period is long enough to ensure measurement of all possible pair of inputs and outputs. By running the above estimator, $\mathcal{A}_i$ can reconstruct all decision functions. Referring to the Table 3.2, denoting $u_{i,1} = s_{i,1}, \cdots, u_{i,4} = s_{i,4}, u_{i,5} = \lambda_{i,1}, \cdots, u_{i,8} = \lambda_{i,4}$, the automaton for the right–hand species may be rewritten as

$$\delta_i : \Sigma_i \times \mathbb{B}^9 \to \Sigma_i$$

$$(\mathsf{FAST}, y_{i,1}) \mapsto \mathsf{FAST}\,, (\mathsf{FAST}, y_{i,2}) \mapsto \mathsf{SLOW}\,,$$

$$(\mathsf{FAST}, y_{i,3}) \mapsto \mathsf{LEFT}\,, (\mathsf{FAST}, y_{i,4}) \mapsto \mathsf{RIGHT}\,,$$

$$(\mathsf{SLOW}, y_{i,5}) \mapsto \mathsf{FAST}\,, (\mathsf{SLOW}, y_{i,2}) \mapsto \mathsf{SLOW}\,,$$

$$(\mathsf{SLOW}, y_{i,3}) \mapsto \mathsf{LEFT}\,,$$

$$(\mathsf{LEFT}, y_{i,6}) \mapsto \mathsf{FAST}\,, (\mathsf{LEFT}, y_{i,7}) \mapsto \mathsf{LEFT}\,,$$

$$(\mathsf{RIGHT}, y_{i,8}) \mapsto \mathsf{FAST}\,, (\mathsf{RIGHT}, y_{i,9}) \mapsto \mathsf{RIGHT}\,.$$

and the decision vector results:

$$y_{i,1} = \neg u_{i,1}\,(u_{i,3} + u_{i,6})\,, \;\; y_{i,2} = u_{i,1}\,(u_{i,2} + u_{i,4} + u_{i,5})\;,$$

$$y_{i,3} = u_{i,1} \neg u_{i,2} \neg u_{i,4}\, \neg u_{i,5}, \;\; y_{i,4} = \neg u_{i,4}\, \neg u_{i,3}\, \neg u_{i,6}\,,$$

$$y_{i,5} = \neg u_{i,1}\,, \;\; y_{i,6} = \neg u_{i,1} + u_{i,7}\,, \;\; y_{i,7} = u_{i,1}\, \neg u_{i,7}\,,$$

$$y_{i,8} = u_{i,1} + u_{i,8}\,, \;\; y_{i,9} = \neg u_{i,1}\, \neg u_{i,8}\,.$$

As an example, Fig. 6.7 reports a complete, randomly generated input vector sequence U and the corresponding output for the decision function $y_{i,1}$. As expected from theory, the estimated function is

$$\hat{y}_{i,1} = u_{i,3} \oplus u_{i,6} \oplus u_{i,1}u_{i,3} \oplus u_{i,1}u_{i,6} \oplus$$

$$\oplus\, u_{i,3}u_{i,6} \oplus u_{i,1}u_{i,3}u_{i,6} \qquad = y_{i,1}\,.$$

Finally, Fig. 6.8 shows the evolution of the estimation errors, i.e. the number of input combinations for which the value of each of the nine estimated decision functions differ from the real ones. As expected from theory this number goes to zero and all network interaction structures are correctly identified.

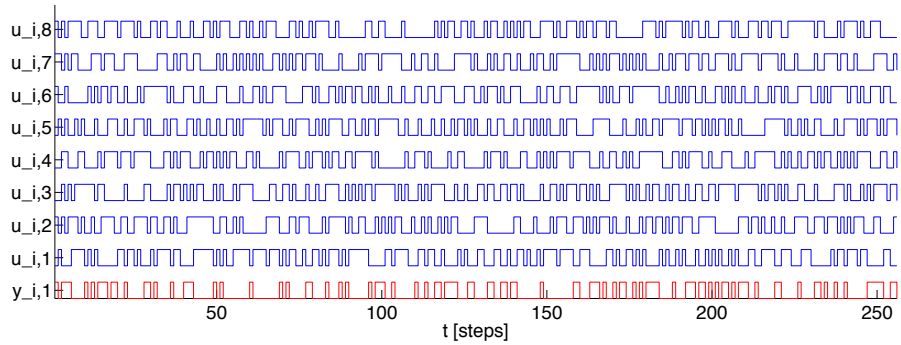


Fig. 6.7 A complete, randomly generated input vector sequence U and the corresponding output for the decision function $y_{i,1}$.

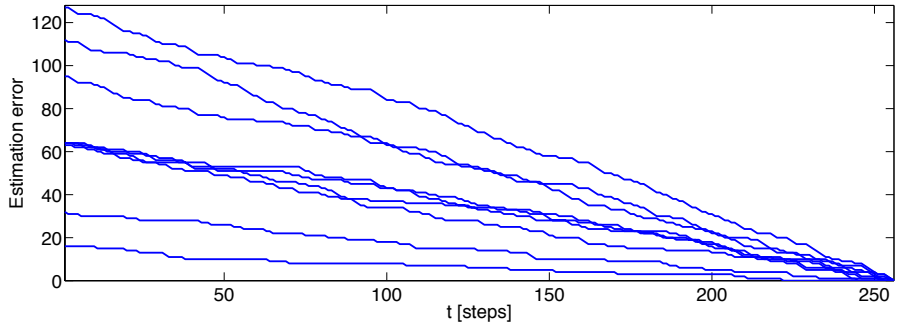


Fig. 6.8 Evolution of the estimation errors, $\sum_u (f_i(u) \oplus \hat{f}_i(u))$, i.e. the number of input combinations for which the value of each estimated decision functions differ from the real ones.

Chapter 7

An Industrial Society for Fast and Reliable Factory Automation

Present manufacturing and production industries are facing challenging complexity specifications that require very high levels of productivity and of quality to be matched by an unprecedented level of flexibility and sustainability along with a strong reduction in maintenance and reconfiguration costs. This not only requires a paradigm shift in the organization of the production sites and of the logistic areas, but also a much further exploitation of industrial robotic systems. In this context, the traditional operational scenario where robots are segregated in specific areas of the plants, carry out repetitive and elementary tasks and are controlled by a centralized intelligence is doomed to a quick obsolescence for its unexpected inability to guarantee a sufficient level of scalability, robustness, and reconfigurability.

Multi–vehicle robotic systems are largely used in industrial transportation and logistics systems as they provide competitive advantage versus the single-agent solutions in terms of task speedup, robustness and scalability. For instance, a typical function of a multi Laser Guided Vehicle (LGV) system consists in transporting raw or semi–finished material from a factory's warehouse to its production lines. However, their adoption raises management and coordination problems, such as collision avoidance, conflict resolution requiring fast and reliable negotiation of shared resources.

Conflict resolution in the use of these resources is critical for safety and robustness of the operations of material handling. Avoidance/resolution of stall situations as well as fluent navigation of the robotic agents must be guaranteed for system efficiency. Stall situations occur when agents are unable to move from the particular configuration (i.e. deadlock) or constrained to move along a finite number of paths without reaching the final destination (i.e. livelock).

In this topic, the academic literature is divided into two categories: centralized and decentralized, see e.g. [14], [35], [36] and [37]. A method using the notion of composite robot is presented in [96]. Other centralized approach, using e.g. the *master-slave* control, are proposed in [97] and using the so called coordination diagram in [98] and [99]. A distributed route planning method for multiple mobile robots that uses so-called Lagrangian decomposition technique is presented in [100]; in [101] authors presents a coordination algorithm, which can be considered in between centralized and decoupled planning. In [102] a framework for decentralized and parallel coordination system, based on dynamic assignment of robot motion priorities, is developed, but only the collision avoidance problem has been addressed, while in [103, 104], the workspace is decomposed into discrete spatial resources and robots move on preplanned paths applying the concept of distributed mutual exclusion [105] to coordinate their motions.

Many other works focus also on the important aspects of collision avoidance and deadlock avoidance (see e.g. [38, 39, 40, 41]). In [41] a novel paradigm for conflict resolution in multi vehicle traffic systems, where a number of mobile agents move freely in a finite area following a pre–defined motion profile is presented. The key idea is the strictly related with the tessellation of the underlying motion area in a finite number of cells. This cells are considered as resources that have to be acquired by the mobile agents for the execution of their motion profile, according with an appropriate resource allocation protocol. The developed protocol is based upon the real–time management of sequential resource allocation systems (RASs) and it is able to formally guarantee the safety

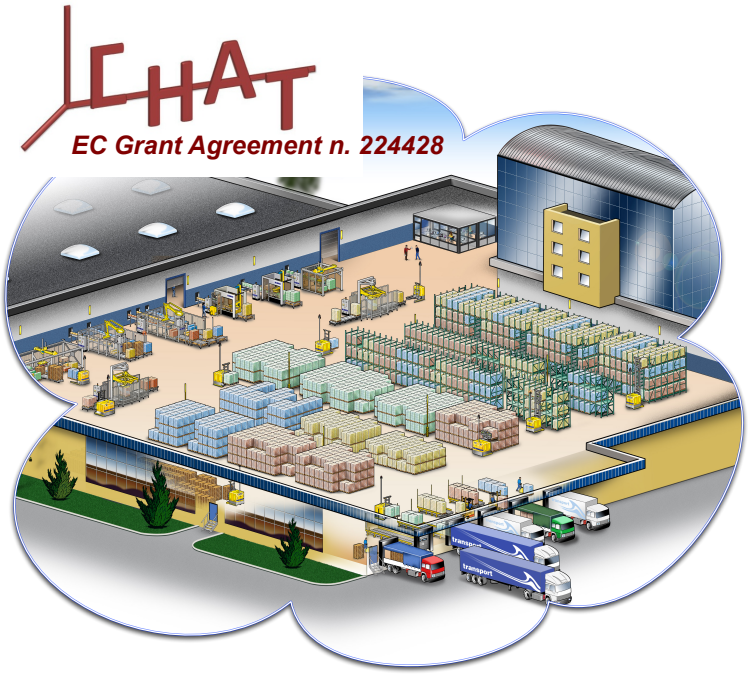


Fig. 7.1 CHAT project objective: a production plant where the decision skills are increased by enhancing the awareness on the status of the factory floor.

operation of the underlying traffic system, while remaining scalable w.r.t. the number of the involved agents. It is worth noting that this approach is applicable even to those traffic systems where all vehicles have to be in perpetual motion until their retirement. Furthermore, in [106] and [107] a technique, based on a Petri net, that avoids deadlocks through re-routing is presented. Other strategies, e.g. [108], [109], [110] and [111], avoid deadlocks trying to detect a cyclic-waiting situation, using graph theory for planning paths such that deadlock is a-priori avoided or using a matrix- based deadlock detection algorithm.

In [112] are described some interesting aspects related to the Kiva system which creates a new paradigm for pick-pack-and-ship warehouses that significantly improves worker productivity, using movable storage shelves that can be lifted by small, autonomous robots. Although the overall system is cooperative, Kiva robots are essentially independent. No robot depends upon any other robot to accomplish its task, but the system requires them all to successfully complete a customer order. Each robot and station are represented in the system by a drive unit agent (DUA) and by an inventory station agent (ISA) respectively. Robots receive requests and act basing on them. At the same time, the system embodies a massive, real-time, resource allocation problem together with resource allocation, task, path, and motion planning by using a control stack with standard abstraction layers which are well known in literature.

In [113] the application of a formal hybrid control approach to design semi-autonomous multi vehicle systems that are guaranteed to be safe is illustrated. It is proved that, in a structured task, such as driving, simple human-decision models can be effectively learned and employed in a feedback control system, allowing the control to guarantee safety specifications. Deterministic models are here considered even if human decision models are more naturally captured by stochastic frameworks, in which uncertainty due to variability in both subjects and realizations of the same decision is probabilistic.

Starting from this point and looking a little bit further ahead we may think to robots not limited to the traditional stereotype of large heavy industrial manipulators or mobile platforms. Probably, we may envision a future where there will be no fixed stereotype at all, since robots of different type, coming from different makers, will possibly enter the production at different times. Large systems of fast and heterogeneous units will co-exist in the factory, sharing the same units and communicating with each other through wireless connections. Some robots will pursue a common goal in a coordinated effort. In other cases, they could have different and sometimes conflicting goals (e.g., reaching

a common location in minimum time through a narrow corridor). The society will have to simplify cooperation and enable each agent to achieve its goals without jeopardizing the chances of others.

The application realized for a demonstration of a EU project CHAT (Chapter 7), showed that by using the tools described in this work, it is possible to improve the factory's decision skills by enhancing its awareness on the current situation of the factory floor.

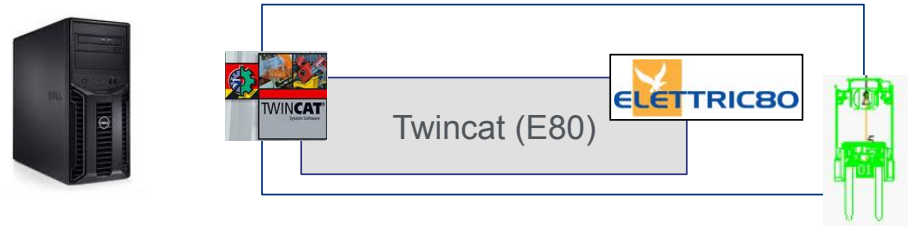


Fig. 7.2 Centralized processing system based on Beckoff's TwinCAT.

7.1 The EU Project CHAT

As stated above, over the last few years many academic achievements have evidenced a spectacular growth in robots' abilities to operate autonomously and in coordinated teams, thus disclosing unsuspected opportunities as e.g. in disaster management, planetary explorations, surveillance and control on a geographic scale. However, most of these results remain concealed into their labs and do not contribute to the development of competitive technologies for the industry. The CHAT project aims at diminishing the gap between these two worlds in the very important domain of industrial mobility systems. This objective is intended to be pursued through a concrete knowledge transfer and implantation of innovative ideas from modern cognitive robotics into the realm

of factory automation in a way that is compatible with its specific needs. The idea underpinning the project is that a competitive automated factory can only be realized if the intelligence of each robotic component (involving their sensing, computation, communication, and actuation skills) is exploited for its own benefit and for that of the other robots.

7.1.1 Motivation and Goals

The project effort strongly advocate for the implementation of a full–fledged *"society" of robots*, where the different and heterogeneous agents operate and interact using a blend of autonomous skills, *social rules* and an eventual central coordination. Robots in this society communicate with each other through wireless connections for a common goal in a coordinated effort, or to solve conflicting goals (e.g., reaching a common location in minimum time through a narrow corridor). The CHAT project pursues three concrete and relevant objectives to provide *practical solutions* exposing the limitation of the current factories and *implanting* the scientific and technological foundations for a future factory with high levels of integration and interaction. The first goal is to improve the current factory's decision skills by enhancing its awareness on the status of the factory floor. This is intended to be achieved through an effective distributed sensing scheme, where every "member" of the society acquires local information of the system state for its own use and for sharing it with the neighbors and eventually with a central supervisor. Members of the factory society must be able to reconstruct/estimate the status of every other member and even that of the global system, through cooperative, scalable, reliable, and fast interaction schemes and communication protocols .

The second objective is that of enhancing the current coordination capabilities of the mobile robotic systems (mostly represented by Laser Guided Vehicles) through the introduction of rule–based coordination and collaboration

motion protocols . These rules provide for a correct–by–construction behavior, combining safety (collision avoidance with robots and humans) and performance (ensuring that each robot eventually gets a chance to accomplish its mission). In addition, such rules must be able to solve conflicts, ensuring the absence of deadlock and livelock situations and a fluent motion. In the view of the factory society, these protocols must allow new robots to get in or to leave, at any time (*openness* property), irrespective of model, type, size or weight of the robots (*heterogeneity* property) – provided only that they abide the society's rules. Moreover, the architecture of every robotic agents must be modular composable (*adaptability* property), enabling e.g. an autonomous vehicle moving from hangar to hangar to dynamically download a new localization service while traveling in the outdoor.

Strongly related to the two first objectives is the essential requirement of detecting, classifying, and recovery from unexpected faults in the factory — including misbehavior of some of its robotic "peers" — that may be spontaneous or caused by malicious intervention or tampering. As a final third goal, the CHAT project aims at endowing the factory robotic society with built–in functionalities enabling quick, cooperative detection of misbehaving components and effective recovery and reconfiguration of the system .

7.1.2 The Case Study

The considered industrial case refers to the logistic scenario of industrial plants with operations typical of process industry, and stores, where the transport of material takes place: away from the endpoint of the processing lines that feeds the store, into a temporary storage location, and then away from this and into the start point of the next processing segment. Thus, the material movement takes place at stores, which is usually (but not necessarily) organized with asynchronous operation under performance—based algorithms that pur-

Fig. 7.3 LGVs produced by Elettric80 at the beginning of the demo.

sue optimization based on queue capacity. Briefly, from a conceptual point of view, several options of control distribution may be implemented (and reported here) in this plant scenario at increasing level of decentralization, and with a look also at the implications in terms of hierarchical structure of the control problem:

1 A decision maker (or a controller, agent or node of the network) is responsible for a single task;
2 Any agent is able to manage the task but has only partial information on the task: in this case collaboration must be achieved among agents in order to complete the task (e.g. by using "consensus" algorithms);

Centralized control policies are certainly better in terms of trajectory optimality, but they tend to be much more conservative than needed, i.e. robots are assigned with paths that temporally minimize their intersection. Moreover, they are hardly limited by the computational time request that increases with the number of robots that are involved. Another disadvantage of the centralized control policies is that if the central control unit fails, the whole system is out

of control. The major benefits of the decentralized distributed approach can be identified as follows:

- Scalability: ability to handle growing amount of work without compromising efficiency or without the need of reconfiguring the already active agents (and the technology associated);
- Modularity: ability to handle complex processes through the cooperation of simpler agents, allowing faster development from planning and design to production;
- Resiliency and Fault Tolerance: ability of the system to continue to operate correctly even though one or more of its components are malfunctioning or corrupted;
- Maintenance and programming: each distributed agent is easier to maintain, program and reconfigure, since interaction with the rest of the plant is minimized;
- Hardware reduction also from a hardware point of view, the capability of a common network to manage the communication from and to all the layers will reduce network complexity and hardware requirements in terms of cables, gateway, switches, etc.

Distributed control implies that a given mission has to be accomplished coordinating the efforts of multiple LGVs, either all equivalent or specialized, with none of them mastering all the other agents. Decisions as how to accomplish the mission are taken not by a single, but by several, co–operating LGVs. Each of these robots may be aware of the whole mission, of the set of resources available, of the other cooperating robots: in this case the cooperation is provided consciously by each of them, i.e. it is explicitly coded in their internal software. Alternatively, each robots may be not aware of the mission as a whole and not aware of the other robots: in this case cooperation is provided by the agents not consciously, but compulsorily, i.e. it is implicit in their algorithms. To give an example, when a shrink wrapper has a pallet ready and calls an LGV to take it away, conscious cooperation implies that all LGVs be aware of

this new task, of the other pending calls and their priorities, of their own placement, status, and capabilities, negotiate and agree between them who shall go and serve this call, all being aware that the agreed decision is the best possible; on the contrary, a fixed, hard coded decision scheme as "the closest idle LGV will always go" would implement unconscious cooperation. Within the logistics area, LGVs deliver material from the output queue of the production lines to the input queue of stretch wrapper machines, and are connected to the communication network through a wireless network.

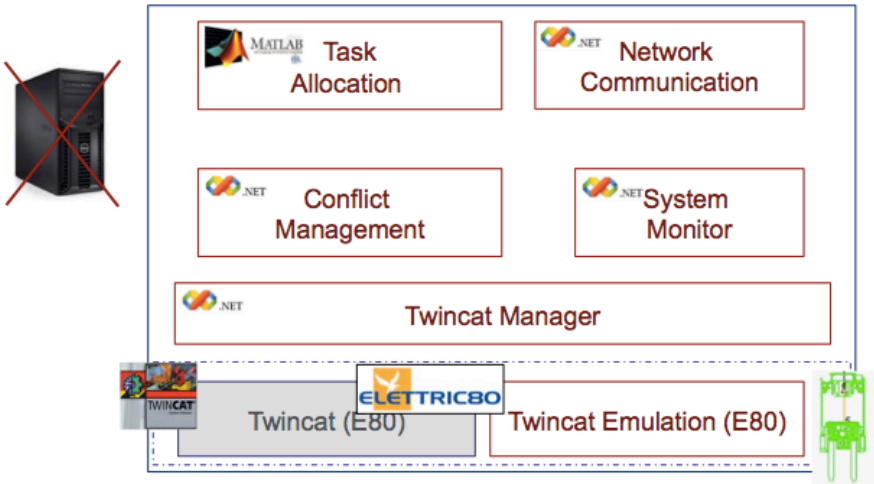


Fig. 7.4 Modular organization of each LGV's distributed controller.

In industrial plants LGVs can move along fixed routes consisting of intersections and passage segments. Such areas usually have finite capacity and can be considered as resources to be shared among the LGVs. As the industrial layout may change (also temporally due to obstacles in the environment), the centralized planning must be recomputed with high computational costs while the system should be shut–off. Decentralized control policies partially resolve the issues of high computational costs, considering the problem divided in two

phases: first, for each agent an optimum path can be defined according to some cost index; then each agent and its neighbors can resolve locally and by themselves conflicts that could arise, according to a shared coordination policy.

Notwithstanding this, fully decentralized approaches tend to disregard information that is made available by the infrastructure of the factory, and may turn out not to be the most efficient solutions. These are typically organized into two phases: during a first planning phase robots' paths are computed by using independent objectives for each robot; during a second coordination phase, robots cooperatively manage their motion based on their local neighborhood situation.

7.2 The Proposed Architecture

The industrial partner in the CHAT project, *Elettric 80 Spa* [114], which is the worldwide market leader of AGV production, has defined control and architectural requirements for the case–study, based on the specifications of Delipapier plant, one of the tissue production sites of one of its major customers, Sofidel S.p.A. [115]. Within this plant, paper pallets have to be moved from production lines, to temporary storage locations near stretch-wrappers, and finally to the warehouse. For products movement, a team of LGVs are used. Before the project, coordination of these LGVs was controlled by a centralized processing system with onboard controllers based on Beckoff's TwinCAT [116] system (Fig. 7). No forms of behavior observation and recovery from critical situations were present, thus failure of a unique LGV could cause deadlock, or unbounded time delay in pallets storing and delivery. Inter–robot communication is possible through a wireless network but not used.

Within the above explained context, work aims at seeking the correct trade–off between decentralization and centralization for industrial robot coordination. To achieve this, it also applies and transfer available rule–based, open

control policies [14], ensuring conflict avoidance by design, to the factory domain. Moreover robotic agents share a set of rules, that specify what actions their are allowed to perform in the pursuit of their individual goals: rules are distributed, i.e. they can be evaluated based only on the state of the individual robot, and on information that can be sensed directly or through communication with immediate neighbors. The definition of all the possible LG tasks as behavior according to the rule–based hybrid model defined in Chapter 3, following e.g. [117], allows us to provide LGV with the ability to investigate the status of entire system by monitoring the behavior of the robots according to the procedure described in Chapter 6.

A robot that do not follow the correct rules due to spontaneous failure or malicious tampering can be considered as threat for the entire system and an alarm is triggered. Once an alarm is launched, if possible, the other LGVs undertakes an adequate countermeasure. The monitor has been developed and integrated in the pre-existent control system (TwinCAT), by realizing a software package that is based on the C# programming code and TwinCAT ADS Communication Library (Fig. 7.1.2).

Furthermore, the monitor is based only on locally observed information, thus cooperation with neighbors that can observe the congruence of their neighbors' behavior with the rules is necessary. In this view a crucial aspect is represented by reliable and secure dissemination of information obtained with local and partial observations of the system's state. If dissemination is unreliable, neighboring LGVs may achieve inconsistent local views of the system and consequently take inconsistent actions; if dissemination is not secure, an adversary may modify or inject fake messages so causing AGVs to achieve wrong and/or inconsistent views. In both cases the coordination task may fail, which gives raise to safety issues. Therefore a reliable and secure state information exchange among neighbors is necessary whose component is a scalable mechanism global state reconstruction.

Fig. 7.5 LGVs are moving to the loading points on the left.

Fig. 7.6 A problem with LGV 3 occurs and the system is in deadlock.

Intuitively, neighborhood monitoring is fundamental for reliable state dissemination because, when an AGV broadcasts its state, an accurate and timely

notion of its neighborhood allows it to track which neighbors have received such state and which have not and thus need a re-transmission. Furthermore, the type of information to be elaborated can be very diverse, ranging from scalar values representing e.g. proximity measures to complex sets, representing e.g. the segments or areas that are currently occupied by unexpected obstacles. To effectively disseminate such types of information off–the–shelf consensus algorithms and protocols [16, 17, 18] are inadequate as they are practically able to effectively deal with variables that are real numbers or vectors. To this aim, the innovative approach based on consensus algorithms on different representations of the state of information (Chapters 4,5) is also successfully applied to the factory–specific scenario.

In particular, LGV's position, velocity, pallet status (load/unload), lights (red/yellow/green) (Fig. 7.1.1) and information coming from resource negotiation among the LGVs has been achieved through implementation of a distributed *signboard* where each LGVs can leave messages to its neighbors.

Finally, the validation and comparison of the distributed controller with respect to the centralized TwinCAT has finally shown that the distributed solution developed within CHAT gives better performance and allows solution of unexpected failure of some LGVs.

7.3 The Demonstration

In the demonstration (the reader may refer to `http://www.youtube.com/watch?v=7yOkdydCX3k` for the complete simulation run), 3 LGVs, produced by Elettric 80, are performing logistic operations such as loading pallets at the end of production lines on the left and moving them to the storage points on the right for unloading (Fig. 7.2). Each LGV is given a priority and the paths are composed of segments, which are assigned to LGVs to be executed according to their priority in order to avoid collisions.

Fig. 7.7 The LGVs are now provided with the ability to investigate the status of entire system: when LGV 3 fails, LGVs number 1 and 2 detect that there is a problem in the system.

In the first part of the demonstration, the system is running with the Centralized processing system based on Beckoff's TwinCAT. The demo starts with the LGVs moving to the loading points on the left (Fig. 7.2) along a path assigned by the centralized warehouse manager system.

Unfortunately a problem with LGV 3 occurs and is not able to leave the loading point. Since the priority of this LGV is higher than that of the other LGV, they stop waiting until LGV 3 completes its path. The system is now in deadlock and the only possibility for exiting this condition is for the user to solve the problem with LGV3 so that it can complete its task (Fig. 7.2).

In the second part of the demo, this situation can be avoided by using the proposed distributed control architecture (Fig. 7.1.2). The LGVs are now provided with the ability to investigate the status of entire system by combining the information coming from the partial observations of each LGV.

As before, the LGVs are moving to the loading points on the left and when LGV number 3 fails, as indicated by the red lights, LGVs number 1 and 2 de-

Fig. 7.8 After having detected that the problem is relate with LGV 3, LGV 1 and 2 are able to react specifically to this problem overcoming the deadlock.

tect that there is a problem in the system and that this problem is related to the LGV 3 (Fig. 7.3). Thus, they are able to react specifically to the problem by dynamically changing the assigned priority, leaving LGV 3 out of consideration and thereby overcoming the deadlock (Fig. 7.3). As soon as the problem with LGV 3 has been solved so that it can continue with its task, as indicated by the green leds, LGVs 1 and 2 realize that LGV 3 is proper functioning, and the system can be restored to the initial condition (Fig. 7.3).

Fig. 7.9 As soon as the problem with LGV 3 has been solved, as indicated by the green leds, LGVs 1 and 2 realize that the problem has been solved, and the system can be restored to the initial condition.

Chapter 8

Conclusions

This thesis have focused on how very large numbers of heterogenous robots, differing in their bodies, sensing and intelligence, may be made to coexist, communicate, and compete fairly towards achieving their individual goals, i.e. to build a "society of robots". More specifically, "robotic societies" are distributed multi–agent systems where each agent is assigned with a possibly different local goal, but needs to coordinate its actions with other neighboring agents. "Behavior–based" society of robots can be built by giving a set of rules that each robot has to follow and that are based only on information that is locally available for each robot via communication with neighboring robots and proprioceptive sensing. This work have presented a formalism that allows a large variety of possible cooperative systems to be uniformly modeled. The complexity need to represent such behaviors can be successfully captured by hybrid models, in which a continuous–time dynamics describes the physical motion of each agent, while an event–based one describes the sequence of interactions with its neighbors. Moreover, this thesis have focused on distributed algorithms that members of the society can use to reach a consensus on the environment and on the integrity of the peers, so as to improve the overall security of the society of robots. In particular, the work have focused on the convergence of information that is not represented by real numbers, as often in the literature, rather by sets. The dynamics of the evolution of informa-

tion across the network is accordingly described by set–valued iterative maps. While the study of convergence of set–valued iterative maps is highly complex in general, this thesis have focused on Boolean maps, which are comprised of arbitrary combinations of unions, intersections, and complements of sets. For these important class of systems the work have provided results on convergence and on synthesis.

Furthermore, the work have addressed the the problem of classifying a set of robotic agents, based on their dynamics or the interaction protocols they obeys, as belonging to different “species". The proposed procedure allows a distributed classification systems to be built, that are based on a decentralized identification mechanism, by which every agent classifies its neighbors using only locally available information. Finally, this thesis have proved that all the tools developed in the thesis can be successfully applied to a real industrial society of robots. By extending the above concept to the factory structured environment, it is possible to set the basis for a full-fledged factory of the future, where the different and heterogeneous agents operate and interact using a blend of autonomous skills, social rules and central coordination.

References

Author's

[A1] S. Martini, A. Fagiolini, G. Dini, and A. Bicchi, "Safety and security in networked robotic systems via logical consensus," *International Workshop on Networked embedded and control system technologies: European and Russian RD cooperation (NESTER'09*, 2009.

[A2] A. Fagiolini, S. Martini, and A. Bicchi, "Set–valued consensus for distributed clock synchronization," *IEEE Conf. on Automation Science and Engineering*, 2009.

[A3] A. Fagiolini, S. Martini, N. Dubbini, and A. Bicchi, "Distributed consensus on boolean information," in *1st IFAC Workshop on Estimation and Control of Networked Systems (NecSys'09)*, Venice, Italy, September, 24 - 26 2009, pp. 72 – 77.

[A4] A. Fagiolini, S. Martini, D. Di Baccio, and A. Bicchi, "A self-routing protocol for distributed consensus on logical information," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2010. IROS 2010.*, 2010.

[A5] S. Martini, A. Fagiolini, G. Zichittella, M. Egerstedt, and A. Bicchi, "Decentralized classification in societies of autonomous and heteroge-

nous robots," in *ICRA. Proceedings of the IEEE International Conference on Robotics and Automation*, 2011.

[A6] S. Martini, D. Di Baccio, A. Fagiolini, and A. Bicchi, "Robust network agreement on logical information," in *18th IFAC World Congress, IFAC 2011*, 2011.

[A7] M. Franceschelli, S. Martini, M. Egerstedt, A. Bicchi, and A. Giua, "Observability and controllability verification in multi-agent systems through decentralized laplacian spectrum estimation," in *Decision and Control (CDC), 2010 49th IEEE Conference on*. IEEE, 2010, pp. 5775–5780.

Miscellaneous

[8] C. Fall, E. Marland, J. Wagner, and J. Tyson, *Computational Cell Biology*. Springer, 2005.

[9] T. Balch and R. Arkin, "Behavior-based formation control for multirobot systems," vol. 14, no. 6, pp. 926–939, 1998.

[10] B. Bamieh, F. Paganini, and M. Dahleh, "Distributed control of spatially-invariant systems," vol. 47, no. 7, pp. 1091–1107, 2002.

[11] M. Mesbahi and F. Y. Hadaegh, "Formation flying control of multiple spacecraft via graphs, matrix inequalities, and switching," vol. 24, no. 2, pp. 369–377, 2001.

[12] S. Valverde, G. Theraulaz, J. Gautrais, V. Fourcassie, and R. Sole, "Self-organization patterns in wasp and open source communities," *IEEE Intelligent Systems*, vol. 21, no. 2, pp. 36–40, 2006.

[13] C. Kube and E. Bonabeau, "Cooperative transport by ants and robots," *Robotics and autonomous systems*, vol. 30, no. 1-2, pp. 85–102, 2000.

[14] L. Pallottino, V. G. Scordio, E. Frazzoli, and A. Bicchi, "Decentralized cooperative policy for conflict resolution in multi-vehicle systems," *IEEE Transaction on Robotics*, vol. 23, no. 6, pp. 1170–1183, 2007.

[15] F. Bullo, J. Cortés, and S. Martinez, *Distributed control of robotic networks*. Princeton University Press, 2007.

[16] R. Olfati-Saber, J. Fax, and R. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, p. 215, 2007.

[17] A. Jadbabaie, J. Lin, and A. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, 2003.

[18] J. Fax and R. Murray, "Information flow and cooperative control of vehicle formations," vol. 49, no. 9, pp. 1465–1476, Sept. 2004.

[19] S. Oh, L. Schenato, P. Chen, and S. Sastry, "Tracking and coordination of multiple agents using sensor networks: system design, algorithms and experiments," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 234–254, 2007.

[20] M. Egerstedt and X. Hu, "Formation constrained multi-agent control," Seoul, Korea, May 2001, pp. 3961–3967.

[21] L. Parker, "Distributed intelligence: Overview of the field and its application in multi-robot systems," *Journal of Physical Agents*, vol. 2, no. 1, p. 5, 2008.

[22] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," vol. 48, no. 6, pp. 988–1001, 2003.

[23] M. Ji and M. Egerstedt, "Distributed coordination control of multi-agent systems while preserving connectedness,," vol. 23, no. 4, pp. 693–703, Aug. 2007.

[24] Z. Lin, M. Broucke, and B. Francis, "Local control strategies for groups of mobile autonomous agents," vol. 49, no. 4, pp. 622–629, 2004.

[25] S. Martínez, J. Cortés, and F. Bullo, "Motion coordination with distributed information," vol. 27, no. 4, pp. 75–88, 2007.

[26] M. Mesbahi, "On state-dependent dynamic graphs and their controllability properties," vol. 50, no. 3, pp. 387–392, 2005.

[27] H. Tanner, A. Jadbabaie, and G. J. Pappas, "Flocking in fixed and switching networks," vol. 52, no. 5, pp. 863–868, 2007.

[28] J. Cortés, S. Martínez, and F. Bullo, "Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions," vol. 51, no. 8, pp. 1289–1298, 2006.

[29] J. Fax and R. Murray, "Information flow and cooperative control of vehicle formations," vol. 49, no. 9, pp. 1465–1476, Sept. 2004.

[30] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Trans. on Programming Languages and Systems (TOPLAS)*, vol. 4, no. 3, pp. 382–401, 1982.

[31] S. Calinon, F. Guenter, and A. Billard, "Goal-directed imitation in a humanoid robot," in *ICRA. Proceedings of the IEEE International Conference on Robotics and Automation*, 2005, pp. 299 – 304.

[32] S.Calinon, F. Guenter, and A. Billard, "On learning the statistical representation of a task and generalizing it to various contexts," in *ICRA. Proceedings of the IEEE International Conference on Robotics and Automation*, May 2006, pp. 2978 –2983.

[33] Y. Nakamura, T. Inamura, and H. Tanie, "A statistic model of embodied symbol emergence," in *Robotics Research*, ser. Springer Tracts in Advanced Robotics, P. Dario and R. Chatila, Eds. Springer Berlin / Heidelberg, 2005, vol. 15, pp. 573–584.

[34] M. H. Stone, "The theory of represent. for boolean algebras," *Trans. of American Mathematical Society*, vol. 40, no. 1, pp. 37–111, 1936.

[35] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006.

[36] R. Alami, S. Fleury, M. Herrb, F. Ingrand, and F. Robert, "Multi-robot cooperation in the martha project," *Robotics Automation Magazine, IEEE*, vol. 5, no. 1, pp. 36–47, Mar 1998.

[37] J. Lygeros, D. Godbole, and S. Sastry, "Verified hybrid controllers for automated vehicles," *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 522–539, Apr 1998.

[38] E. Roszkowska and S. Reveliotis, "On the liveness of guidepath-based, zoned-controlled, dynamically routed, closed traffic systems," *Automatic Control, IEEE Transactions on*, vol. 53, pp. 1689–1695, 2008.

[39] M. Fanti, "Event-based controller to avoid deadlock and collisions in zone-control agvs," *Int. J. of Production Res.*, vol. 40, no. 6, pp. 1453–1478, 2002.

[40] S. Reveliotis and P. Ferreira, "Deadlock avoidance policies for automated manufacturing cells," *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 6, pp. 845–857, 2002.

[41] S. A. Reveliotis, "Conflict resolution in agv systems," *IIE Transactions*, vol. 32, pp. 647–659, 2000.

[42] "Chat - control of heterogeneus automation systems," http://http://www.ict-chat.eu/ A project supported by the European Commission under the 7th Framework Programme.

[43] J. Cortés, S. Martínez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Trans. Robot. Autom.*, vol. 20, no. 2, pp. 243â±–255, 2004.

[44] J. Lin, A. S. Morse, and B. D. O. Anderson, "The multi-agent rendezvous problem," Maui, HI, Dec. 2003, pp. 1508–1513.

[45] M. Franceschelli, M. Egerstedt, and A. Giua, "Motion Probes for Fault Detection and Recovery in Networked Control Systems," *American Control Conference, Seattle, WA, June*, 2008.

[46] R. Olfati-Saber and R. M. Murray, "Consensus protocols for networks of dynamic agents," Denver, CO, Jun. 2003, pp. 951–956.

[47] A. Olshevsky and J. N. Tsitsiklis, "Convergence rates in distributed consensus and averaging," San Diego, CA, Dec. 2006, pp. 3387–3392.

[48] H. Asama, K. Ozaki, H. Itakura, A. Matsumoto, Y. Ishida, and I. Endo, "Collision avoidance among multiple mobile robots based on rules and communication," in *Intelligent Robots and Systems' 91.'Intelligence for Mechanical Systems, Proceedings IROS'91. IEEE/RSJ International Workshop on*. IEEE, 1991, pp. 1215–1220.

[49] S. Kato, S. Nishiyama, and J. Takeno, "Coordinating mobile robots by applying traffic rules," in *Proc. of the lEEE/RSJ International Conference onIntelligent Robots and Systems*, vol. 3, 1992.

[50] C. Candea, H. Hu, L. Iocchi, D. Nardi, and M. Piaggio, "Coordination in multi-agent robocup teams," *Robotics and Autonomous Systems*, vol. 36, no. 2, pp. 67–86, 2001.

[51] M. Benjamin, J. Leonard, J. Curcio, and P. Newman, "A method for protocol-based collision avoidance between autonomous marine surface craft," *Journal of Field Robotics*, vol. 23, no. 5, pp. 333–346, 2006.

[52] J. Lygeros, "Lecture notes on hybrid systems," in *Notes for an ENSIETA workshop*. Citeseer, 2004.

[53] B. Hölldobler and E. Wilson, "The multiple recruitment systems of the African weaver ant Oecophylla longinoda (Latreille)(Hymenoptera: Formicidae)," *Behavioral Ecology and Sociobiology*, vol. 3, no. 1, pp. 19–60, 1978.

[54] W. Gronenberg, "Structure and function of ant (Hymenoptera: Formicidae) brains: Strength in numbers," *Myrmecological News*, vol. 11, pp. 25–36, 2008.

[55] Y. Madi and K. Jaffe, "On foraging behavior of the polymorphic tree dwelling ant Daceton armigerum (Hymenoptera: Formicidae)," *Entomotropica*, vol. 21, no. 2, 2006.

[56] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, 2002.

[57] J. Cortés, S. Martínez, T. Karataş, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[58] N. Lynch, *Distributed Algorithms*. Morgan Kaufmann Publishers.

[59] P. Frasca, R. Carli, F. Fagnani, and S. Zampieri, "Average consensus on networks with quantized communication," *Intl. Journal of Robust and Nonlinear Control*, 2008.

[60] A. Fagiolini, E. Visibelli, and A. Bicchi, "Logical Consensus for Distributed Network Agreement," in *IEEE Conf. on Decision and Control*, 2008, pp. 5250–5255.

[61] S. Sundaram and C. Hadjicostis, "Information dissemination in networks via linear iterative strategies over finite fields," in *IEEE Conf. on Decision and Control and Chineese Control Conference*, 2009, pp. 3781–3786.

[62] K. Marzullo, "Maintaining the time in a distributed system: An example of a loosely-coupled distributed service." *Dissertation Abstracts International Part B: Science and Engineering,*, vol. 46, no. 1, 1985.

[63] D. Mills, "Internet time synchronization: the network time protocol," *IEEE Trans. on Comm.*, vol. 39, no. 10, Oct 1991.

[64] M. Di Marco, A. Garulli, A. Giannitrapani, and A. Vicino, "Simultaneous localization and map building for a team of cooperating robots: a set membership approach," vol. 19, no. 2, pp. 238–249, 2003.

[65] G. Hardy and E. Wright, *An introduction to the theory of numbers*. Oxford University Press, 2008.

[66] F. Robert, "Théorèmes de perron–frobenius et stein–rosenberg booleens," *Linear Algebra and its applications*, vol. 19, pp. 237–250, 1978.

[67] ——, "Itérations sur des ensembles finis convergence d'automates cellulaires contractants," *Linear Algebra and its applications*, vol. 29, pp. 393–412, 1980.

[68] S. Sundaram and C. Hadjicostis, "Control of quantized multi-agent systems with linear nearest neighbor rules: A finite field approach," in *American Control Conference (ACC), 2010*. IEEE, 2010, pp. 1003–1008.

[69] M. Nosarzewska, “Evaluation de la différence entre l’aire d’une région plane convexe et le nombre des points aux coordonnées entiéres couverts par elle,” *Colloq. Math.*, vol. 1, pp. pp. 305–311, 1948.

[70] K. Römer, “Time synchronization in ad hoc networks,” *In Mobi Hoc’01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, 2001.

[71] M. Sichitiu and C. Veerarittiphan, “Simple, accurate time synchronization for wireless sensor networks,” *In WCNC’03: Proceedings of the IEEE Wireless Communications and Networking Conference*, 2003.

[72] H. Dai and R. Han, “Tsync: a lightweight bidirectional time synchronization service for wireless sensor networks,” *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 8, no. 1, 2004.

[73] J. van Greunen and J. Rabaey, “Lightweight timesynchronization for sensor networks,” *In WSNA’03: Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, 2003.

[74] M. Maròti, B. Kusy, G. Simon, and A. Lèdeczi, “The flooding time synchronization protocol,” *In SenSys’04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, 2004.

[75] R. Solis, V. Borkar, and P. Kumar, “A new distributed time synchronization protocol for multi hop wireless networks,” *45th IEEE Conference on Decision and Control*, 2006.

[76] J. Elson, L. Girod, and D. Estrin, “Fine-grained network time synchronization using reference broadcasts,” *SIGOPSOper. Syst. Rev.*, vol. 36, 2002.

[77] S. Ganeriwal, R. Kumar, and M. Srivastava, “Timing-sync protocol for sensor networks,” *In SenSys’03: Proceedings of the 1st international conference on Embedded networked sensor systems*, 2003.

[78] J. Eidson and K. Lee, “Ieee 1588 standard for a precision clock synchronization protocol for networked measurement and control systems,”

Sensors for Industry Conference, 2002. 2nd ISA/IEEE, pp. 98– 105, Nov. 2002.

[79] L. Schenato and G. Gamba, "A distributed consensus protocol for clock synchronization inwireless sensor network," *46th IEEE Conference on Decision and Control*, Dec 2007.

[80] M. Fischer, N. Lynch, and M. S. Paterson, "Impossibility of distributed consensus with one faulty process," *ACM Journal of the Association for Computing Machinery*, vol. 32, no. 2, pp. 374–382, April 1985.

[81] C. D. Godsil and G. F. Royle, *Algebraic Graph Theory*, ser. Graduate Texts in Mathematics, 2001, vol. 207.

[82] A. Fagiolini, M. Pellinacci, G. Valenti, G. Dini, and A. Bicchi, "Consensus–based Distributed Intrusion Detection for Secure Multi-Robot Systems," *IEEE Int. Conf. on Robotics and Automation*, 2008.

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

[84] S. Martínez, F. Bullo, J. Cortés, and E. Frazzoli, "On synchronous robotic networks – Part I: Models, tasks, and complexity," Apr. 2005, submitted. Electronic version available at `http://motion.mee.ucsb.edu`.

[85] J. O'Rourke, *Art Gallery Theorems and Algorithms*, 1987.

[86] MOTEIV, "Tmotesky datasheet: Ultra low power ieee 802.15.4 compliant wireless sensor module," 2006.

[87] A. Dunkels, B. Grönvall, and T. Voigt, "Contiki a lightweight and flexible operating system for tiny networked sensor," in *Proceeding of IEEE Workshop on Embedded Networked Sensor*. IEEE, 2004.

[88] A. Dunkels, T. Voigt, J. Alonso, H. Ritter, and J. Schiller, "Connecting Wireless Sensornets with TCP/IP Networks," in *Proceedings of the Second International Conference on Wired/Wireless Internet Communications (WWIC2004)*, Frankfurt (Oder), Germany, FEB 2004.

[89] A. Balluchi, L. Benvenuti, M. Di Benedetto, and A. Sangiovanni-Vincentelli, "Design of observers for hybrid systems," *Hybrid Systems: Computation and Control*, vol. 2289, pp. 76–89, 2002.

[90] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[91] L. Rabiner and B. Juang, "An introduction to hidden Markov models," *IEEE ASSp Magazine*, vol. 3, no. 1 Part 1, pp. 4–16, 1986.

[92] L. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains," *The annals of mathematical statistics*, vol. 41, no. 1, pp. 164–171, 1970.

[93] M. Karnaugh, "The map method for synthesis of combinational logic circuits," *Trans. AIEE. pt. I*, vol. 72, no. 9, pp. 593–599, 1953.

[94] E. J. McCluskey, "Minimization of boolean functions," *Bell Syst. Tech.*, vol. 35, no. 5, pp. 1417–1444, 1956.

[95] F. Robert, "Dérivée discrète et comportement local d'une itération discrète," *Linear algebra and its applications*, vol. 52, pp. 547–589, 1983.

[96] P. Svestka and M. Overmars, "Coordinated motion planning for multiple car-like robots using probabilistic roadmaps," *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1631–1636, May 1995.

[97] S. Yuta and S. Premvuti, "Coordinating autonomous and centralized decision making to achieve cooperative behaviors between multiple mobile robots," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, pp. 1566–1574, Jul 1992.

[98] P. O'Donnell and T. Lozano-Periz, "Deadlock-free and collision-free coordination of two robot manipulators," *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 1, pp. 484–489, May 1989.

[99] R. Olmi, C. Secchi, and C. Fantuzzi, "Coordination of multiple agvs in an industrial application," *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1916–1921, May 2008.

[100] Y. Guo and L. Parker, "A distributed and optimal motion planning approach for multiple mobile robots," *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 3, pp. 2612–2619, 2002.

[101] S. LaValle and S. Hutchinson, "Optimal motion planning for multiple robots having independent goals," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 6, pp. 912–925, Dec 1998.

[102] K. Azarm and G. Schmidt, "Conflict-free motion of multiple mobile robots based on decentralized motion planning and negotiation," *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 4, pp. 3526–3533, Apr 1997.

[103] J. Wang, "Operating primitives supporting traffic regulation and control of mobile robots under distributed robotic systems," *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1613–1618, May 1995.

[104] J. Wang and S. Premvuti, "Distributed traffic regulation and control for multiple autonomous mobile robots operating in discrete space," *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1619–1624, May 1995.

[105] L. Lamport, "The mutual exclusion problem: part i—a theory of interprocess communication and partii—statement and solutions," *J. ACM*, vol. 33, no. 2, pp. 313–348, 1986 I and II.

[106] N. Wu and M. Zhou, "Shortest routing of bidirectional automated guided vehicles avoiding deadlock and blocking," *IEEE/ASME Transactions on Mechatronics*, vol. 12, no. 1, pp. 63–72, Feb. 2007.

[107] M. Fanti, "A deadlock avoidance strategy for agv systems modelled by coloured petri nets," *Proc. of 6th International Workshop on Discrete Event Systems*, pp. 61–66, 2002.

[108] R. Lochana Moorthy, W. Hock-Guan, N. Wing-Cheong, and T. Chung-Piaw, "Cyclic deadlock prediction and avoidance for zone-controlled agv system," *International Journal of Production Economics*, vol. 83, no. 3, pp. 309–324, 2003.

[109] M. Singhal, "Deadlock detection in distributed systems," *Computer*, vol. 22, no. 11, pp. 37–48, 1989.

[110] J. Yoo, E. Sim, C. Cao, and J. Park, "An algorithm for deadlock avoidance in an agv system," *The International Journal of Advanced Manufacturing Technology*, vol. 26, no. 5, pp. 659–668, 2005.

[111] M. Lehmann, M. Grunow, and H. Günther, "Deadlock handling for real-time control of agvs at automated container terminals," *OR Spectrum*, vol. 28, no. 4, pp. 631–657, 2006.

[112] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," in *Proceedings of the 19th national conference on Innovative applications of artificial intelligence - Volume 2*. AAAI Press, 2007, pp. 1752–1759. [Online]. Available: http://dl.acm.org/citation.cfm?id=1620113.1620125

[113] R. Verma and D. Vecchio, "Semiautonomous multivehicle safety," *Robotics Automation Magazine, IEEE*, vol. 18, no. 3, pp. 44 –54, sept. 2011.

[114] "Elettric 80 s.p.a." http://www.elettric80.it.

[115] "Sofidel s.p.a." http://www.sofidel.it.

[116] "The beckhoff twincat software system." http://www.beckhoff.ae.

[117] S. Manca, A. Fagiolini, and L. Pallottino, "Decentralized coordination system for multiple agvs in a structured environment," in *2011 Congress of the International Federation of Automatic Control*, Milano, italy, August 28 - September 2 2011, pp. 6005 – 6010.