Event-triggered control of multi-agent systems: pinning control, cloud coordination, and sensor coverage

ANTONIO ADALDO

Licentiate Thesis
Stockholm, Sweden 2016

KTH Royal Institute of Technology
School of Electrical Engineering
Department of Automatic Control
SE-100 44 Stockholm
Sweden

TRITA-EE 2016:129
ISSN 1653-5146
ISBN 978-91-7729-081-0

Akademisk avhandling som med tillstånd av Kungliga Tekniska högskolan framlägges till offentlig granskning för avläggande av Teknologie licentiatexamen i elektro- och systemteknik den 23 september 2016 klockan 10:00 i sal E3, Kungliga Tekniska högskolan, Osquars backe 14, Stockholm.

Tryck: Universitetsservice US AB

Abstract

A multi-agent system is composed of interconnected subsystems, or agents. In control of multi-agent systems, the aim is to obtain a coordinated behavior of the overall system through local interactions among the agents. Communication among the agents often occurs over a wireless medium with finite capacity. In this thesis, we investigate multi-agent control systems where inter-agent communication is modelled by discrete events triggered by state conditions.

In the first part, we consider event-triggered pinning control for a network of agents with nonlinear dynamics and time-varying topologies. Pinning control is a strategy to steer the behavior of a multi-agent system in a desired manner by controlling only a small fraction of the agents. We express the controllability of the network in terms of an average value of the network connectivity over time, and we show that all the agents can be driven to a desired reference trajectory.

In the second part, we propose a control algorithm for multi-agent systems where inter-agent communication is substituted with a shared remote repository hosted on a cloud. Communication between each agent and the cloud is modelled as a sequence of events scheduled recursively by the agent. We quantify the connectivity of the network and we show that it is possible to synchronize the multi-agent system to the same state trajectory, while guaranteeing that two consecutive cloud accesses by the same agent are separated by a finite time interval.

In the third part, we propose a family of distributed algorithms for coverage and inspection tasks for a network of mobile sensors with asymmetric footprints. We develop an abstract model of the environment under inspection and define a measure of the coverage attained by the sensor network. We show that the sensor network attains nondecreasing coverage, and we characterize the equilibrium configurations.

The results presented in the thesis are corroborated by simulations or experiments.

Contents

Acknowledgments

I thank my supervisor Karl Henrik Johansson, for being always friendly supportive, and for providing me with insightful guidance and contagious enthusiasm.

I am grateful to my co-supervisor Dimos V. Dimarogonas, for his enthusiasm and careful attention put in all levels of our work. I am grateful to my former advisor Mario di Bernardo, for gently introducing me to the wonders of networked system, but even more for always keeping his door open. I am grateful to my advisor and collaborator Davide Liuzza, for giving me confidence, reserveless support, and hearthfelt advice. I am grateful to my former advisor and collaborator Guodong Shi, for his insightful guidance and careful attention put in our work.

Hearthfelt thanks go to my former collaborator Francesco Alderisio, for taking with me the very first steps in the research world, for coping with my literally uninterrupted presence throughout the whole extent of our exchange studies at KTH, for giving me courage and motivation, and for cheering me up when I was sad.

I thank all my colleagues (current and former) at the Automatic Control department of KTH, for creating a nice working atmosphere, and for always keeping the door open to inspiring conversations. Special thanks go to my office roommates (current and former), to the NetCon group, and to the reading group on Hybrid Control, for all the nice and inspiring conversations and activities that we had together. Special thanks go also to Henrik Sandberg, Riccardo Risuleo, and Sebastian Van de Hoef, each of whom kindly agreed to review some chapters of this thesis.

I am grateful to the Knut och Alice Wallenberg Foundation, the European Union through the project AEROWORKS, the school of Electrical Engineering of KTH through the Program of Excellence, and the Swedish Foundation for Strategic Research, for the financial support that made this work possible.

I am grateful to my family and friends for their unconditional love and support. Last, but not least, I am grateful to Frank for making sure that I always had a sufficient supply of provole.

Chapter 1

Introduction

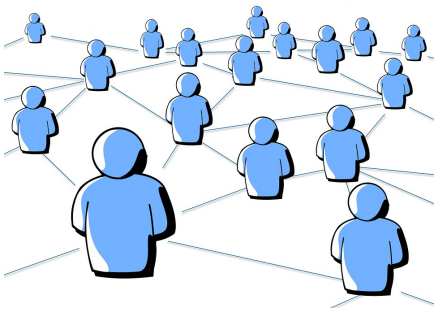
A multi-agent system is a system composed of interconnected subsystems, or *agents*. Each agent behaves according to its own dynamics, but it exchanges some form of interaction with a subset of the other agents in the system. Multi-agent systems have been the subject of an enormous body of research in the past few decades. The reason for so much research attention is that multi-agent systems provide an abstract model for a large number of phenomena of scientific interest, spanning physics, biology, engineering, computer science, and social sciences [1–4]. Figure 1.1 illustrates some examples of entities that can be modelled as multi-agent systems.

The rest of this chapter is organized as follows. In Section 1.1, we discuss some applications of multi-agent systems that have motivated the work presented in this thesis. In Section 1.2, we review some related literature. In Section 1.3, we present the outline of the thesis and the related contributions by the author.

1.1 Motivating examples

Coordination of autonomous underwater vehicles

A wide range of underwater missions, such as wreck inspection, sea floor mapping, or water sample collection, can be carried out by autonomous or semiautonomous vehicles, see Figure 1.2. Such unmanned vehicles are usually known as autonomous underwater vehicles (AUVs). The described underwater missions can be performed more efficiently if more than one AUV is used. However, the use of a fleet of AUVs inevitably brings the problem of coordinating the fleet. This problem can be addressed by modelling the fleet as a multi-agent system, where each AUV is an agent in the system. However, the system model needs to take into account the particular limitations and



(a) A social network. A connection between two individuals represents a social relation. Connected individuals influence each others' behavior and opinions. *Source:* `https://pixabay.com`, Public Domain.

(b) A flock of birds exhibiting swarm behavior. The motion of each bird is influenced by the neighboring birds. *Source:* C. A. Rasmussen, own work, `https://commons.wikimedia.org`, Public Domain.

(c) A power distribution network. The frequency of the current on each line of the network is influenced by the frequency of the current in the neighboring lines. *Source:* `https://pixabay.com`, Public Domain.

(d) A platoon of heavy-duty vehicles. Each vehicle's motion is planned in coordination with the closest vehicles, in order to maintain the platoon and avoid collisions. *Source:* courtesy of Scania.

Figure 1.1. Examples of entities that can be modelled as a multi-agent system.

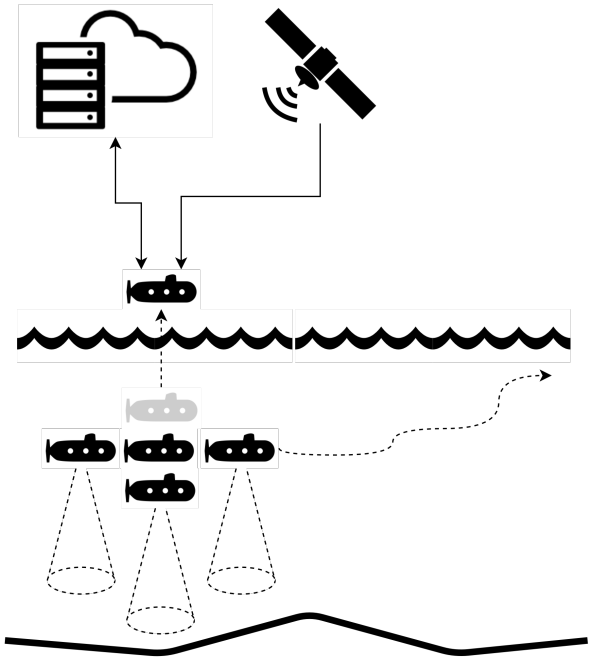


Figure 1.2. Schematic representation of a sea floor mapping mission with a fleet of AUVs. In order to perform a cooperative task, such as mapping the sea floor, the vehicles have to move in a coordinated way. However, underwater communication is severely limited. Moreover, GPS is not available underwater, and the vehicles have to surface whenever they need a GPS position measurement. On the water surface, the vehicles have access to GPS and may also communicate with a base station to deposit and retrieve data.

challenges involved in underwater missions. To achieve coordination, each AUV needs to receive position information about itself and about a subset of the other AUV. However, the common technologies for wireless communication cannot be used in the underwater domain. Underwater communication may be realized by means of acoustic modems, but such devices are short-ranged and power-hungry. Alternatively, two or more AUV may surface to exchange data, but this strategy requires to interrupt the navigation and synchronize the surfacing times. For these reasons, coordination algorithms that involve continuous communication cannot be implemented in realistic underwater setups.

Coverage and inspection control with unmanned vehicles

Deployment and inspection are very common tasks for robotic networks. Challenging missions such as search, recovery, manipulation and environmental monitoring in hazardous environments are desirably delegated to a team of unmanned vehicles. For example, consider the task of inspecting a wind turbine with a team of unmanned aerial vehicles (UAVs), as illustrated in Figure 1.3. The UAVs need to inspect the whole surface of the turbine. The aerial robots are battery-powered, and the duration of the battery imposes an upper bound on the mission time. The robots also need to avoid collisions and counteract possible air currents. Different robots may be equipped with different sensing hardware, which makes each vehicle more apt to inspect certain parts of the turbine rather than others. To coordinate the inspection mission—that is, to agree on which part of the turbine each robot should inspect—the robots communicate over a wireless medium, which is a shared resource with limited capacity. Therefore, communication among the robots should be modelled as intermittent rather than continuous. The robot team can still be seen as a multi-agent system, but the system model needs to take into account the particular limitations and challenges of the mission.

Swarm control

The motion of large group of animals—such as flocks of birds, swarms of insects, or schools of fish—exhibits remarkable coordination behaviors [5]. For example, a flock is capable of following a flight direction, maintaining cohesion, and at the same time avoiding collisions, obstacles, and predators. This coordination emerges as a result of simple interaction rules applied by each individual with respect to its physical neighbors in the group. By studying the flocks as multi-agent systems, and imitating these interaction rules, it is possible to obtain similar coordinated behaviors on a network of mobile robots. Moreover, the motion of a real flock (or swarm, or school) can be influenced by injecting a limited number of artificial agents—also known as *leaders* in the literature—in the group: a few artificial agents may successfully steer the whole flock towards a desired flight direction. The success of the steering depends on the dynamics of the movement of the individual birds, on the number and the position in the group of the artificial agents, and on the topology of the interactions among the birds in the flock.

Thermal regulation in smart buildings

Adjacent rooms in a building influence each other's temperature by means of heat conduction through the building's walls. In thermal regulation problems, we have direct control over the temperature of one or a few rooms,

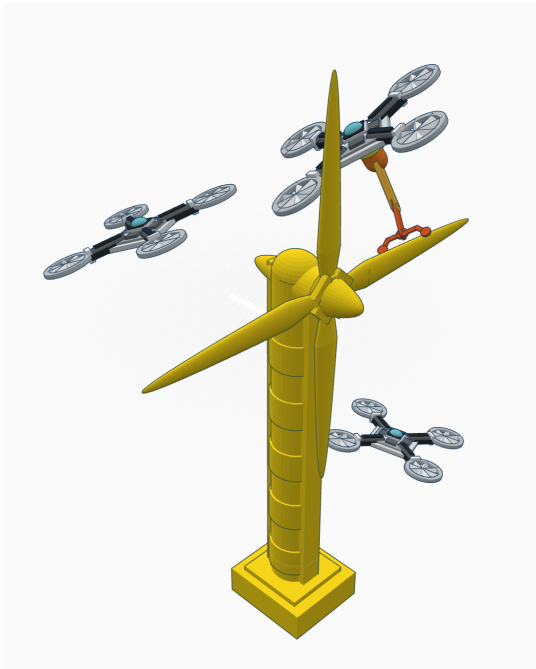


Figure 1.3. Schematic representation of the inspection of a wind turbine with a team of UAVs. The aerial robots need to inspect the whole surface of the turbine. A possible approach is to have each UAV take up one part of the surface of the turbine. To coordinate the mission—that is, to decide which part of the turbine should be assigned to each robot—the UAVs communicate over a wireless medium, which is a shared resource with limited capacity. The robots also need to avoid collisions and counteract possible air currents. Different robots may be equipped with different sensing hardware.

where heating and/or cooling devices are installed, but we want to steer the overall temperature of the building to a desired value. This problem can be addressed by modelling the building as a multi-agent system, where each room represents an agent in the system [6]. The dynamics of the room temperatures depend, other than on the thermal conductivity on the material and other physical factors, on the topology of the wall adjacency within the building, and also on the topology of the heat dispersion to the external environment.

Frequency regulation in power networks

A power network is a network for delivering electricity from suppliers to consumers. A power network consists of generating stations and individual costumers, which are connected by transmission lines that carry the electricity. One of the most important problems in power systems is to maintain the frequency of the current close to a nominal operational frequency [6]. In fact, if the frequency deviates too much from the operational frequency, the generation and utilization equipment may cease to function properly. Adjacent lines influence each other's frequency according to a nonlinear feedback action. Hence, a power network can be modelled as a multi-agent system, where each agent represents a generating station or a costumer. The frequency of the network may be regulated by directly controlling the frequency on a small fraction of the stations, and letting the inter-line feedback action propagate the control action through the network. The topology of the network plays a major role in determining whether it is possible to synchronize all the agents on the operational frequency.

1.2 Related work

Pinning control

Pinning control is a feedback control strategy for synchronization of networks of dynamical systems. In a pinning control task, a common reference trajectory for all the systems in the network is assigned, but it is possible to exert a feedback control action on only a small fraction of the systems. The systems that can receive direct feedback from the reference trajectory are called the *pinned nodes* of the network. However, feedback links also exist among the different systems in the network. Synchronization of the whole network to the reference trajectory is obtained through these inter-system feedback links: information on the reference trajectory propagates in the network through the feedback links and eventually reaches all the systems. The success of the synchronization depends on the dynamics of the system in the network, on the nature of the feedback action among the systems, and on the topology of the interconnections.

A remarkable amount of research work related to pinning control has appeared around the turn of the century; here, we only recall a small selection.

In [7], the authors apply pinning control to networks of chaotic oscillators. In [8], the authors study the problem of selecting the pinned nodes in a special class of networks, called scale-free networks. In [9], the authors compare several pin selection strategies on different network classes. In [10, 11], the authors introduce the concept of *pinning controllability* which quantifies how easy it is to control the network with pinning. Pinning controllability is

defined in terms of the spectral properties of the network and of the intensity of the feedback coupling. In [12], the authors address pinning control of networks of systems with second-order dynamics. In [13], the authors apply pinning control to the problem of cluster synchronization of a network of dynamical systems. In [14], the authors select the pinned nodes via a technique that they call *edge snapping*. In [15], the authors study how the minimum number of pinned nodes that is necessary to control the network varies according to the network topology.

Event-triggered control for multi-agent coordination

In most of the applications of multi-agent systems related to engineering and robotics, the feedback link existing between different agents is not the result of mechanical coupling, but it is realized by means of wireless communication between the controllers of the agents. In this case, it is unrealistic to assume that the agents can exchange feedback continuously. In reality, each agent can send messages to another agent with a certain frequency, which is upper-bounded by the throughput capacity of the wireless medium.

For these reasons, *triggered control* is often considered in the control design for multi-agent applications. In triggered control of multi-agent system, the agents exchange feedback only intermittently. Different flavors of triggered control can be applied to multi-agent systems. With *time-triggered control*, the agents exchange information periodically. With *event-triggered control*, communication is triggered by a condition that is continuously monitored by the agents. When an agent meets the specified condition, it sends a new information packet to the other agents. With *self-triggered control*, each agent schedules its communication instances recursively. Namely, when sending an information packet, an agent schedules the time instant when to send the following packet too. Note that, in the literature, the expression "event-triggered control" is often used to denote triggered control in general.

Event-triggered and self-triggered control [16] have been introduced to reduce the information flow among the different parts of a networked control system (for example, the number of packets sent by the sensors to the controller, and by the controller to the actuators) with respect to the existing time-triggered control strategies [17, 18]. In the context of multi-agent systems, event-triggered control is used to reduce inter-agent communication—that is, the number of packets that the agents have to send to each other to achieve coordination.

Event-triggered control for multi-agent system is introduced in [19]. In [20, 21], the authors introduce the idea of event-based information broadcasting for multi-agent coordination. Event-based broadcasting for stabilization in networked control systems is also studied in [22]. In [23, 24], the authors address self-triggered control for multi-agent systems. In [25], event-

triggered coordination is studied as an application of robust consensus. In [26], the authors study event-based agreement protocols for switching networks. In [27, 28], the authors study event-based consensus for multi-agent systems with general linear agent dynamics. In [29], the authors study event-triggered leader-following for second-order multi-agent systems. In [30], the authors study self-triggered multi-agent coordination with ternary controllers. In [31], the authors consider event-triggered control for discrete-time multi-agent systems. Event-based model predictive control for multi-agent systems has been proposed in [32, 33]. In [34], the authors introduce event-triggered control for synchronization of nonlinear systems.

Triggered control strategies have been proposed for AUV coordination. In [35], the authors employ periodic broadcasting for a formation control task in a network of AUVs. In [36, 37], the authors study event-based motion coordination of AUVs under disturbances. In [38], the authors consider communication scheduling in platoons of underwater vehicles. In [39], the authors use adaptive sampling for multi-AUV control.

Coverage and inspection control

A wide variety of industrial and humanitarian applications involve collecting information in hazardous environments, which makes it desirable to delegate such missions to a team of autonomous robots with sensing capabilities. Typically, the goal is to design a distributed algorithm that drives the robots to a spatial configuration such that the team's collective perception of the environment is optimized according to some criterion. This problem is commonly known as the *coverage* problem in the literature on multi-agent systems and multi-agent robotics. A different but related problem is to inspect the internal or external surface of a building or other structure, for example to detect rusty spots, fractures, or other structural problems. Control strategies addressing coverage and inspection problems are known as *coverage control* and *inspection control* respectively. Coverage and inspection control involve path planning for the single robotic agents, as well as the design of the interaction protocols among the different robots for coordinating the mission operations.

In the last few decades, a lot of research interest has been devoted to the coverage problem, as a way to design the autonomous deployment of robotic sensor teams in an assigned space—see [40–43] to name just a few references.

The vast majority of the existing papers considers mobile sensors with omnidirectional footprints—that is, whose perception of the surrounding environment only depends on the distance between the sensor and the points under observation. Typically, the analysis of such sensor networks relies on *Voronoi tessellations* [44] as a way to partition the environment under inspection into parts and to assign each part to one of the sensors. Once a Voronoi

tessellation is computed, the Lloyd algorithm [44] suggests a natural path planning strategy for each one of the sensors.

Recently, agents with anisotropic [45–49] as well as vision-based [50–52] sensing patterns have been considered. Additional challenges arise if non-convex environments are considered, as studied in [51–54].

Dynamic versions of the coverage problem have also been studied, where the robots do not converge to fixed positions but keep navigating the environment in order to maintain a satisfactory coverage over time. This is commonly known as *effective* or *dynamic coverage* [55, 56]. A vision-based version of effective coverage is studied in [57, 58].

In the real-world implementation of coverage algorithms, the design of the communication protocols among the robots involved in the task constitutes one of the major challenges. To address this challenge, a gossip-based communication strategy for coverage is studied in [59, 60].

In some coverage missions, it is convenient to abstract the environment into a finite set of points (which may either correspond to a sparse set of points of major interest within the environment or provide a discretized approximation of the environment itself). This strategy is explored in [61–63].

1.3 Thesis outline and contributions

The rest of the thesis is organized as follows.

Chapter 2: Technical preliminaries

In Chapter 2, we introduce some technical definitions and results that are used in the thesis.

Chapter 3: Event-triggered pinning control of switching networks

In Chapter 3, we consider the problem of synchronizing a network of nonlinear systems by using event-triggered control updates. This chapter is based on the following contributions.

- A. Adaldo, F. Alderisio, D. Liuzza, G. Shi, D. V. Dimarogonas, M. di Bernardo, and K. H. Johansson, "Event-triggered pinning control of complex networks with switching topologies," in *IEEE Conference on Decision and Control*, pp. 2783–2788, 2014.

- A. Adaldo, F. Alderisio, D. Liuzza, G. Shi, D. V. Dimarogonas, M. di Bernardo, and K. H. Johansson, "Event-triggered pinning control of switching networks," *IEEE Transactions on Control of Network Systems*, vol. 2, no. 2, pp. 204–213, 2015.

Chapter 4: Cloud-supported multi-agent coordination

In Chapter 4, we consider the problem of coordinating a team second-order dynamical systems through the use of a remote information repository, which substitutes inter-agent communication. This chapter is based on the following contributions.

- A. Adaldo, D. Liuzza, D. V. Dimarogonas, and K. H. Johansson, "Control of multi-agent systems with event-triggered cloud access," in *European Control Conference*, 2015.
- A. Adaldo, D. Liuzza, D. V. Dimarogonas, and K. H. Johansson, "Multi-agent trajectory tracking with self-triggered cloud access," *Accepted for publication in the IEEE Conference on Decision and Control*, 2016.
- A. Adaldo, D. Liuzza, D. V. Dimarogonas, and K. H. Johansson, "Cloud-supported coordination of second-order multi-agent systems," *Submitted to the IEEE Transactions on Control of Network Systems*.

Chapter 5: Coverage control of anisotropic sensor networks

In Chapter 5, we consider a coverage problem for a network of mobile sensors. This chapter is based on the following contribution.

- A. Adaldo, D. V. Dimarogonas, and K. H. Johansson, "Discrete partitioning and intermittent communication for anisotropic coverage and inspection missions," *To be submitted to the 2017 World Congress of the International Federation of Automatic Control (IFAC)*.

Chapter 6: Conclusions and future research

In Chapter 6, we present a summary of the results, and discuss directions for future research.

Chapter 2

Technical preliminaries

THE study of multi-agent coordination relies on several results from algebraic graph theory and from linear and nonlinear control theory, while the study of event-triggered control strategies requires some concepts related to hybrid systems. The aim of this chapter is to provide the technical concepts in the mentioned areas that are used to derive the main results presented in this thesis. We also cover some general notational and mathematical preliminaries.

2.1 Notation

The set of the positive integers is denoted $\mathbb{N}$, while $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$. The set of the nonnegative real numbers is denoted $\mathbb{R}_{\geq 0}$, while the set of the positive real numbers is denoted $\mathbb{R}_{>0}$.

For $N \in \mathbb{N}$, the vector made up of N unitary entries is denoted 1_N, the vector made up of N null entries is denoted 0_N, and the N-by-N identity matrix is denoted I_N. For $N, M \in \mathbb{N}$, the N-by-M matrix whose entries are all zero is denoted $0_{N\times M}$.

For a matrix $A \in \mathbb{R}^{N\times M}$, we let $A_{i_1:i_2,j_1:j_2}$ denote the submatrix of A spanning the rows from i_1 to i_2, including i_1 and i_2, and the columns from j_1 to j_2, including j_1 and j_2. We let $A_{i,j_1:j_2} = A_{i:i,j_1:j_2}$ and similarly $A_{i_1:i_2,j} = A_{i_1:i_2,j:j}$. Also, we let $A_{:,j_1,j_2} = A_{1:N,j_1,j_2}$ and similarly $A_{i_1:i_2,:} = A_{i_1:i_2,1:M}$.

The set of the symmetric matrices in $\mathbb{R}^{N\times N}$ is denoted as $\mathbf{S}^N$. For $A \in \mathbf{S}^N$, $A \geq 0$ means that A is positive semidefinite, and $A > 0$ means that A is positive definite. The set of the positive semidefinite matrices in $\mathbf{S}^N$ is denoted as $\mathbf{S}^N_{\geq 0}$, and the set of the positive definite matrices in $\mathbf{S}^N$ is denoted as $\mathbf{S}^N_{>0}$.

The operator $\|\cdot\|$ denotes the Euclidean norm of a vector and the corresponding induced norm of a matrix.

The operator $\otimes$ denotes the Kronecker product. For the properties of this operator, the interested reader is referred to [70].

For $f : \mathbb{R} \to \mathbb{R}^n$, with $n \in \mathbb{N}$, and $t \in \mathbb{R}$, we let $f(t^+) = \lim_{\tau \to t^+} f(\tau)$ and $f(t^-) = \lim_{\tau \to t^-} f(\tau)$.

In pseudocode segments, $=$ denotes logical equality, while $\leftarrow$ denotes assignment.

2.2 Elements of graph theory

In this section, we review some concepts related to graph theory that are commonly used in the study of multi-agent systems. Here we shall take a different cut than the majority of the literature on multi-agent coordination, and focus our attention to what is known as the *edge space* of a graph. Most of the results mentioned in this section can be related to either [2] or [71], but are further elaborated and expanded.

In this thesis, a multi-agent system is often associated to a graph, which represents the topology of the connections among the agents in the multi-agent system. For the purposes of the thesis, a *graph* is defined as a tuple $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$, where $\mathcal{V} = \{1, \dots, N\}$, with $N \in \mathbb{N}$, $\mathcal{E} \subset \mathcal{V}^2$, with the constraint that $(i, i) \notin \mathcal{E}$ for all $i \in \mathcal{V}$, and $w : \mathcal{E} \to \mathbb{R}_{>0}$. The elements of $\mathcal{V}$ are called the *vertexes* of the graph, while the elements of $\mathcal{E}$ are called the *edges* of the graph. For each edge $e = (j, i)$, we let $\mathrm{head}(e) = i$ and $\mathrm{tail}(e) = j$. The number of edges is denoted as M, and the edges are numbered as $e_1, \dots, e_M$. For each edge $e = (j, i)$, the value $w(e) \in \mathbb{R}_{>0}$ is called the *weight* of that edge. With abuse of notation, we denote $w_{i,j} = w((j, i))$.

A graph is often represented by drawing the vertexes as circles and the edges as arrows connecting the circles. Namely, if (j, i) is an edge, then an arrow is drawn from the circle representing vertex j to the circle representing vertex i. The vertexes are labelled with their indexes, while the edges are labelled as $j(w)$, where j is the index and w is the weight. This representation is adopted systematically in this thesis, and it is exemplified in Figure 2.1, for a graph with $N = 4$ vertexes and $M = 5$ edges.

A *path* is a sequence of distinct vertexes $i_1, \dots, i_{P+1}$, with $P \in \mathbb{N}$, such that $(i_k, i_{k+1}) \in \mathcal{E}$ for each $k \in \{0, 1, \dots, P\}$. A *spanning tree* is a subset $\mathcal{T} \subseteq \mathcal{E}$ of the edges with the following properties: (i) there is a vertex $r \in \mathcal{V}$ such that there is a path from r to any other vertex in the graph made up of edges in $\mathcal{T}$; (ii) property (i) does not hold for any proper subset of $\mathcal{T}$. The vertex r is called the *root* of the spanning tree. A spanning tree contains exactly $N - 1$ edges. For example, for the graph represented in Figure 2.1, one spanning tree is given by

$$\mathcal{T} = \{e_1, e_2, e_3\},$$

which has vertex 1 as its root.

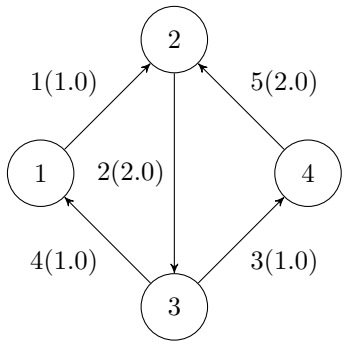


Figure 2.1. Graphical representation of a graph with $N = 4$ nodes and $M = 5$ edges. Each node is labelled with its index, and each edge is labelled with its index and its weight.

For a given graph, several matrices can be defined that describe, partially or completely, its structure. The *incidence matrix* of a graph is defined as $B \in \mathbb{R}^{N\times M}$ such that

$$B_{i,j} = \begin{cases} 1 & \text{if } \mathrm{head}(e_j) = i, \\ -1 & \text{if } \mathrm{tail}(e_j) = i, \\ 0 & \text{otherwise.} \end{cases}$$

Each column of the incidence matrix corresponds to an edge in the graph. If a graph contains a spanning tree, the columns of the incidence matrix can be split into those corresponding to edges in the tree, and those corresponding to edges that are not in the tree. Without loss of generality, suppose that the first $N-1$ columns correspond to the spanning tree, so that we can write

$$B = [B_{\mathcal{T}},\ B_{\mathcal{C}}], \tag{2.1}$$

with $B_{\mathcal{T}} \in \mathbb{R}^{N\times N-1}$ and $B_{\mathcal{C}} \in \mathbb{R}^{N\times M-(N-1)}$. It can be shown that $B_{\mathcal{T}}$ is full column rank, which means that the columns corresponding edges in the spanning tree are linearly independent. On the other hand, the columns corresponding to edges that are not in the tree can be written as linear combinations of the columns corresponding to the edges that are in the tree. Namely, there exists $T \in \mathbb{R}^{M-(N-1)\times(N-1)}$ such that

$$B_{\mathcal{C}} = B_{\mathcal{T}}T. \tag{2.2}$$

Since $B_{\mathcal{T}}$ is full column rank, its left pseudo-inverse $B_{\mathcal{T}}^{\dagger}$ is unique, and T can be computed as

$$T = B_{\mathcal{T}}^{\dagger}B_{\mathcal{C}}.$$

The *weight matrix* is defined as $W \in \mathbb{R}^{N\times M}$ such that

$$W_{i,j} = \begin{cases} w_{ij} & \text{if } \mathrm{head}(e_j) = i, \\ 0 & \text{otherwise.} \end{cases}$$

If the graph contains a spanning tree, the columns of the weight matrix can be split in a similar way as done for the incidence matrix. Assuming without loss of generality that the spanning tree is made up of the first $N-1$ edges, we let

$$W = [W_{\mathcal{T}},\ W_{\mathcal{C}}]. \tag{2.3}$$

The *Laplacian matrix* is defined as

$$L = WB^\top \in \mathbb{R}^{N\times N}. \tag{2.4}$$

From (2.4), it follows that

$$L_{i,j} = \begin{cases} \sum_{j:(j,i)\in\mathcal{E}} w_{i,j} & \text{if } i = j, \\ -w_{i,j} & (j,i) \in \mathcal{E}, \\ 0 & \text{otherwise.} \end{cases} \tag{2.5}$$

A widely known result in graph theory relates the structural properties of a graph with the eigenvalues of its Laplacian matrix. This result can be formalized as follows.

Lemma 2.1. *The eigenvalues of the Laplacian matrix of a graph have nonnegative real parts. Zero is always an eigenvalue and 1_N is always an eigenvector with eigenvalue zero. All the nonzero eigenvalues have positive real parts. The algebraic multiplicity of the eigenvalue zero is one if and only if the graph contains a spanning tree.*

For example, for the graph represented in Figure 2.1, we have

$$\mathrm{eig}(L) = (0, 1, 3 \pm \sqrt{2}i).$$

The *edge Laplacian matrix* is defined as

$$E = B^\top W \in \mathbb{R}^{M\times M}. \tag{2.6}$$

As a consequence of Sylvester's determinant identity [72], the Laplacian and the edge Laplacian have the same nonzero eigenvalues with the same algebraic multiplicities. In particular, if the graph contains a spanning tree, then, by Lemma 2.1, the algebraic multiplicities of the nonzero eigenvalues of the Laplacian sum to $N-1$, and therefore also the algebraic multiplicities of nonzero eigenvalues of the edge Laplacian sum to $N-1$. This implies that,

in the edge Laplacian, the eigenvalue zero has multiplicity $M-(N-1)$. For example, for the graph represented in Figure 2.1, we have

$$\mathrm{eig}(E) = (0, 0, 1, 3 \pm \sqrt{2}i).$$

If the graph contains a spanning tree, we can substitute (2.1)–(2.3) into (2.6), which yields

$$E = \begin{bmatrix} B_{\mathcal{T}}^{\top} W_{\mathcal{T}} & B_{\mathcal{T}}^{\top} W_{\mathcal{C}} \\ T^{\top} B_{\mathcal{T}}^{\top} W_{\mathcal{T}} & T^{\top} B_{\mathcal{T}}^{\top} W_{\mathcal{C}} \end{bmatrix}.$$

Applying the similarity transformation

$$S = \begin{bmatrix} I_{N-1} & 0_{(N-1)\times(M-(N-1))} \\ -T^{\top} & I_{M-(N-1)} \end{bmatrix},$$

we have

$$SES^{-1} = \begin{bmatrix} B_{\mathcal{T}}(W_{\mathcal{T}} + W_{\mathcal{C}} T^{\top}) & B_{\mathcal{T}} W_{\mathcal{C}} \\ 0_{(M-(N-1))\times(N-1)} & 0_{M-(N-1)} \end{bmatrix}.$$

The upper-left block of SES^{-1} is called the *reduced edge Laplcian* of the graph, and it is denoted as $R \in \mathbb{R}^{(N-1)\times(N-1)}$. Namely, we let

$$R = B_{\mathcal{T}}(W_{\mathcal{T}} + W_{\mathcal{C}} T^{\top}). \tag{2.7}$$

Lemma 2.2. *In a graph that contains a spanning tree, the eigenvalues of the reduced edge Laplacian coincide with the nonzero eigenvalues of the Laplacian. Consequently, $-R$ is Hurwitz.*

Proof. Since SES^{-1} is block-triangular, its eigenvalues are the eigenvalues of the two blocks on the main diagonal—that is, R and $0_{M-(N-1)}$. Since E and SES^{-1} are similar, these eigenvalues are also the eigenvalues of the edge Laplacian, with the same algebraic multiplicities. In particular, in the edge Laplacian the eigenvalue zero has multiplicity $M-(N-1)$, which corresponds to all the eigenvalues in the block $0_{M-(N-1)}$ in SES^{-1}. It follows that the eigenvalues of R, which are the remaining $N-1$ eigenvalues of SES^{-1}, coincide with the nonzero eigenvalues of E, which are also the nonzero eigenvalues of L. By Lemma 2.1, these eigenvalues have positive real parts, which implies that $-R$ is Hurwitz. □

For example, for the graph represented in Figure 2.1, we have

$$\mathrm{eig}(R) = (1, 3 \pm \sqrt{2}i)$$

Remark 2.1. *Note that the reduced edge Laplacian is defined only if the graph has a spanning tree. Moreover, if the graph has more than one spanning tree, selecting different spanning trees leads to different reduced edge Laplacians. However, all the possible reduced edge Laplacians have the same eigenvalues, because they coincide with the nonzero eigenvalues of the Laplacian.*

Remark 2.2. *If $\mathcal{T} = \mathcal{E}$, the reduced edge Laplacian is conventionally defined as the edge Laplacian itself, $R = E$. Lemma 2.2 also holds in this case, and has a similar proof.*

A graph is said to be *undirected* if $(j,i) \in \mathcal{E} \iff (i,j) \in \mathcal{E}$ and $w_{i,j} = w_{j,i}$ for all $(j,i) \in \mathcal{E}$. In the graphical representation of an undirected graph, both the edges in a pair (j,i) and (i,j) are usually represented as a single line connecting nodes i and j. Clearly, if an undirected graph contains a spanning tree, then there exists a path from any vertex to any other vertex. For this reason, if an undirected graph contains a spanning tree, we say that it is *connected*. It follows from (2.5) that, for an undirected graph, the Laplacian matrix is symmetric, and consequently it has real eigenvalues. Hence, Lemma 2.1 can be specialized to undirected graphs as follows.

Lemma 2.3. *For an undirected graph, the Laplacian matrix has real nonnegative eigenvalues. Zero is always an eigenvalue, and 1_N is always an eigenvector of eigenvalue zero. The multiplicity of the eigenvalue zero is one if and only if the graph is connected.*

If an undirected graph is not connected, then its vertex set $\mathcal{V}$ can be partitioned into subsets $\mathcal{V}_1, \ldots, \mathcal{V}_{N_c}$ with the property that, within each subset, there exists a path from every node to every other node. Denoting as $\mathcal{E}_i$ the restriction of $\mathcal{E}$ to the vertexes in $\mathcal{V}_i$, we have that $(\mathcal{V}_i, \mathcal{E}_i)$ is a connected graph. The graphs $(\mathcal{V}_i, \mathcal{E}_i)$ are called the *(connected) components* of the original graph. A connected graph can be seen as having a single connected component that coincides with the graph itself. Given an undirected graph, suppose without loss of generality that the vertexes are indexed in such a way that the first n_1 vertexes belong to one component, the following n_2 vertexes belong to another component, etc. Then, it follows from (2.5) that the Laplacian matrix is block-diagonal, with each block being the Laplacian matrix of the corresponding component of the graph. Consequently, Lemma 2.3 generalizes as follows.

Lemma 2.4. *In the Laplacian of an undirected graph, the multiplicity of the eigenvalue zero is equal to the number N_c of the connected components in the graph. Supposing without loss of generality that the vertexes are labelled in such a way that the first n_1 vertexes belong to one component, the following n_2 vertexes belong to another component, etc., the eigenvectors with eigenvalue zero are in the form $[\alpha_1 1_{n_1}^\top,\ \alpha_2 1_{n_2}^\top,\ \ldots,\ \alpha_{N_c} 1_{n_{N_c}}^\top]^\top$, with $\alpha_i \in \mathbb{R}$ for all $i \in \{1, \ldots, N_c\}$.*

2.3 Hybrid time trajectories and Zeno behavior

In this section, we define the concepts of hybrid time trajectory and Zeno behavior. These concepts have been given different formal definitions in the literature. The definitions that are given in this section follow [73], but they are modified to best suit the problems considered in this thesis.

Definition 2.1 (hybrid time trajectory)**.** *A* hybrid time trajectory (HTT) $\tau = \{I_i\}_{i=0}^{N}$ *is a finite or infinite sequence of intervals of* $\mathbb{R}$ *such that:*

- *for all* $i < N$, $I_i = [\tau_i, \tau_i']$ *with* $\tau_i \leq \tau_i' = \tau_{i+1}$;
- *if* $N < \infty$, *then either* $I_N = [\tau_N, \tau_N']$ *with* $\tau_N \leq \tau_N' < \infty$, *or* $I_N = [\tau_N, +\infty)$.

The time instants τ_i, *with* $i \in \{0, 1, \ldots, N_\tau\}$, *are called the* events *of the* HTT. *For an infinite* HTT, *the (finite or infinite) time instant* $\tau_\infty = \sum_{i=0}^{\infty}(\tau_i' - \tau_i)$ *is called the* Zeno time *of the* HTT. *The concept of Zeno time is extended to finite* HTTs *by setting* $\tau_\infty = \infty$.

The interpretation of a HTT is that some form of event or transition occurs at the time instants τ_i. A HTT extends to infinity if it is an infinite sequence, but also if it is a finite sequence ending with an interval of the form $[\tau_{N_\tau}, \infty)$. Note that a HTT is uniquely defined by the sequence of its events, $\{\tau_i\}_{i=0}^{N}$. For this reason, sometimes we refer to a HTT by the sequence of its events.

Definition 2.2 (Zeno behavior)**.** *A* HTT τ *is said to exhibit* Zeno behavior *if it is infinite and* $\tau_\infty < \infty$.

If a HTT exhibits Zeno behavior, we also say, for brevity, that it is *Zeno*. The interpretation of a Zeno HTT is that the events have a finite accumulation point, namely $\tau_\infty < \infty$.

In this thesis, HTTs are associated to control signals, and the events in a trajectory correspond to the updates of the control signal. Therefore, Zeno behavior corresponds to an accumulation of control updates, and results into the impossibility to implement the control law in a physical control system. Therefore, in this thesis we regard Zeno behavior as an undesired phenomenon, and for every proposed control law, we show that it does not induce Zeno behavior in the closed-loop system. A sufficient condition to exclude Zeno behavior is given in the following lemma.

Lemma 2.5. *If, for an infinite* HTT τ, *there is a positive lower bound on the inter-event times* $\tau_i' - \tau_i$ *for all* i, *then* τ *is not Zeno.*

Proof. Let $\tau_i' - \tau_i \geq \delta$ for all i. Then $\sum_{i=0}^{\infty}(\tau_i' - \tau_i) \geq \sum_{i=0}^{\infty}\delta = \infty$, which by Definition 2.2 means that τ is not Zeno. □

Lemma 2.6. *An infinite* HTT *is not Zeno if and only if for any $T > 0$, there is an event larger than T.*

Proof. By definition, we have that τ is not Zeno if and only if $\sum_{i=0}^{\infty}(\tau_i' - \tau_i) = \infty$, which, by the definition of limit, means that for any $T \geq 0$ there is a $\nu \in \mathbb{N}_0$ such that for any $n > \nu$ we have $\sum_{i=0}^{n}(\tau_i' - \tau_i) > T$. But the left-hand side of the last inequality is τ_{n+1}, therefore τ is not Zeno if and only if, for any $T \geq 0$, there exists an event larger than T. □

For the purposes of this thesis, we need to elaborate further on the concept of HTT than what is directly available in [73]. Namely, we introduce here the concept of union of two HTTs.

Definition 2.3 (union of two HTTs)**.** *Let $\tau^{(1)}$ and $\tau^{(2)}$ be two* HTT*s with N_1 and N_2 events respectively. The union of $\tau^{(1)}$ and $\tau^{(2)}$ is denoted as $\tau^{(1)} \cup \tau^{(2)}$, and it is defined as the* HTT *whose events are $\{t \in \{\tau^{(1)}\}_{i=0}^{N_1} \cup \{\tau^{(2)}\}_{i=0}^{N_2} : \ t < \tau_\infty^{(1)}, \tau_\infty^{(2)}\}$.*

Lemma 2.7. *If $\tau^{(1)}$ and $\tau^{(2)}$ are not Zeno, then $\tau^{(1)} \cup \tau^{(2)}$ is not Zeno.*

Proof. We need to distinguish two cases. If $\tau^{(1)}$ and $\tau^{(2)}$ are both finite, then $\tau^{(1)} \cup \tau^{(2)}$ is finite, and therefore it is not Zeno. Suppose now that one of the sequences is infinite. Without loss of generality, suppose that $\tau^{(1)}$ is infinite. Note that, since both the trajectories are not Zeno, we have $\tau_\infty^{(1)} = \tau_\infty^{(2)} = \infty$. Therefore, by Definition 2.3, the events of $\tau^{(1)} \cup \tau^{(2)}$ are the union of the events of $\tau^{(1)}$ and the events of $\tau^{(2)}$. Since $\tau^{(1)}$ is not Zeno, by Lemma 2.6, for any $T > 0$ there is an event of $\tau^{(1)}$ larger than T. But since any event of $\tau^{(1)}$ is also an event of $\tau^{(1)} \cup \tau^{(2)}$, we have that for any $T > 0$ there is an event of $\tau^{(1)} \cup \tau^{(2)}$ larger than T, which by Lemma 2.6 means that $\tau^{(1)} \cup \tau^{(2)}$ is not Zeno. □

Chapter 3

Event-triggered pinning control of switching networks

In this chapter, we consider a problem of event-triggered pinning control of a multi-agent system with switching topology.

Pinning control is a strategy to steer the collective behavior of a networked multi-agent system by directly controlling only a small fraction of the agents. The goal is for the states of the agents to converge onto a given reference trajectory, which corresponds to a control objective. The agents that receive direct feedback control from the reference trajectory are called *pins*, or are said to be *pinned*.

In many application scenarios for pinning control, an assumption that the topology of the network is constant over time is unrealistic. Topology variations result from imperfect communications links among the agents or simply from the existence of a proximity range beyond which communication is not possible. In this chapter we show that the proposed pinning control strategy is robust with respect to a class of uncontrolled topology variations.

Pinning control algorithm have been traditionally designed under the hypothesis of continuous-time communication. However, in many realistic networked systems, the information flow among the agents has some limitations, due, for example, to the finite capacity of the communication medium or to communication costs. To address such limitations, the control strategy that is proposed in this chapter employs event-triggered communication.

The rest of the chapter is organized as follows. In Section 3.1, we give a mathematical formulation of the pinning control problem under investigation, and we outline the proposed control algorithm to address such problem. In Section 3.2, we give a graph-theoretical interpretation of the proposed algorithm. In Section 3.3, we discuss a distributed and model-based implementation of the proposed algorithm that aims at reducing the necessary amount

of communication among the agents. In Section 3.4, we state our main convergence result, whose proof occupies Sections 3.5–3.7. In Section 3.8, we specialize the general results to the case of networks with fixed topologies. In Section 3.9, we present a simulated network of nonlinear systems under the proposed algorithm, and we show that the simulation corroborates the theoretical results. Section 3.10 concludes the chapter by summarizing the results and outlining possible future developments.

3.1 Problem statement

Consider a multi-agent system with agents indexed as $\mathcal{V} = \{1, \dots, N\}$. Let each agent have state $x_i(t) \in \mathbb{R}^n$ that evolves according to

$$\begin{cases} \dot{x}_i(t) = f(t, x_i(t)) + u_i(t), \\ x_i(0) = x_{i,0}, \end{cases} \tag{3.1}$$

where $f : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ is a time-varying field, $x_{i,0} \in \mathbb{R}^n$ is an initial condition, and $u_i(t) \in \mathbb{R}^n$ is a control input. We introduce the assumption that f is a globally Lipschitz function of the state, with uniform Lipschitz constant with respect to the time. This assumption is formalized as follows.

Assumption 3.1. *For each $t \geq 0$, the function $f(t, \cdot)$ is globally Lipschitz with Lipschitz constant λ_f. Namely, there exists $\lambda_f > 0$ such that, for each $t \geq 0$ and each $x_1, x_2 \in \mathbb{R}^n$, we have*

$$\|f(t, x_1) - f(t, x_2)\| \leq \lambda_f \|x_1 - x_2\|.$$

A reference trajectory $r(t) \in \mathbb{R}^n$ is assigned, whose dynamics is compatible with the dynamics of the agents. Namely, we have

$$\begin{cases} \dot{r}(t) = f(t, r(t)), \\ r(0) = r_0. \end{cases} \tag{3.2}$$

The control objective is that the states of all the agents asymptotically converge to the reference trajectory, and it is formalized as

$$\lim_{t \to \infty} \|r(t) - x_i(t)\| = 0 \; \forall i \in \mathcal{V}. \tag{3.3}$$

To reach the control objective, we employ piecewise constant control signals,

$$u_i(t) \equiv u_{i,k} \; \forall t \in [t_{i,k}, t_{i,k+1}), \tag{3.4}$$

with $u_{i,k} \in \mathbb{R}^n$. The sequence of the time instants $\{t_{i,k}\}_{k\in\mathbb{N}_0}$ defines a HTT, and corresponds to the times when the control signal $u_i(t)$ is updated to a new value. We let $t_{i,0} = 0$ for all $i \in \mathcal{V}$, so that $u_{i,0}$ is the initial control input for each agent. The control values are computed as

$$\begin{aligned} u_{i,k} = &\sum_{j=1}^{N} w_{i,j}(t_{i,k}^+)C(x_j(t_{i,k}) - x_i(t_{i,k})) \\ &+ p_i(t_{i,k}^+)K(r(t_{i,k}) - x_i(t_{i,k})), \end{aligned} \tag{3.5}$$

where $w_{i,j}(t) \in \mathbb{R}$ and $p_i(t) \in \mathbb{R}$ for all $t \geq 0$, and $C, K \in \mathbf{S}^n_{>0}$. The interpretation is that each agent receives feedback from the other agents and from the reference trajectory to align its state with the states of the other agents and with the reference. The matrices C and K can be interpreted as a control protocol that translates a mismatch in the state space into a control action. The scalar $w_{i,j}(t)$ is the weight of the feedback from agent j to agent i at time t, while $p_i(t)$ is the weight of the feedback from the reference trajectory to agent i at time t. Hence, the scalars $w_{i,j}(t)$ and $p_i(t)$ with $i, j \in \mathcal{V}$ define the topology of the networked multi-agent system at each time instant $t \geq 0$. We make the assumption that the feedback between two agents is symmetric, which is formalized as follows.

Assumption 3.2. *For each $i, j \in \mathcal{V}$, we have $w_{i,j}(t) = w_{j,i}(t)$ for each $t \geq 0$.*

We also make the assumption that the signals $w_{i,j}(t)$ and $p_i(t)$ are piecewise constant and bounded. This agrees with the interpretation that changes in the values of $w_{i,j}(t), p_i(t)$ correspond to changes in the topology of the agents' network, due, for example, to communication failures. This assumption is formalized as follows.

Assumption 3.3. *The signals $w_{i,j}(t)$ and $p_i(t)$ that appear in* (3.5) *are piecewise constant, and they are lower-bounded and upper-bounded. Namely, there exist $\underline{w}_{i,j}, \bar{w}_{i,j}$ and $\underline{p}_i, \bar{p}_i$ such that $\underline{w}_{i,j} \leq w_{i,j}(t) \leq \bar{w}_{i,j}$ and $\underline{p}_i \leq p_i(t) \leq \bar{p}_i$ for all $t \geq 0$. Moreover, the* HTT *defined by the instants when a change of value occurs for some $w_{i,j}(t)$ or $p_i(t)$ is not Zeno.*

In practice, $w_{i,j}(t) \neq 0$ means that agents i and j are connected, and can exchange information, while $p_i(t) \neq 0$ means that agent i is connected to the reference. In most applications, we have that, at each time $t \geq 0$, $w_{i,j}(t) \neq 0$ only for a small fraction of the possible pairs of agents, and $p_i(t) \neq 0$ only for a small fractions of the agents. Note that, in order to compute $u_{i,k}$ as by (3.5), agent i only needs to receive the state of agent j if $w_{i,j}(t_{i,k}^+) \neq 0$, and similarly, it only needs to receive the value $r(t_{i,k})$ of the reference if $p_i(t_{i,k}^+) \neq 0$. For these reasons, (3.5) can be considered a pinning control law.

In order to completely define our control strategy, we also need to specify a rule for scheduling the control updates $t_{i,k}$ for each agent. To this aim, consider the following signals:

$$\begin{aligned} z_i(t) = &\sum_{j=1}^{N} w_{i,j}(t) C(x_j(t) - x_i(t)) \\ &+ p_i(t) K(r(t) - x_i(t)). \end{aligned} \tag{3.6}$$

Note that $z_i(t)$ is similar to the control signals (3.5), but the update time $t_{i,k}$ is substituted with the current time. In other words, $z_i(t)$ would correspond to the control input $u_i(t)$ if this were to be continuously updated. An update for agent i is scheduled for each time instant when the difference between $z_i(t)$ and $u_i(t)$ has overcome an assigned threshold. The threshold is defined by the function

$$\varsigma(t) = \varsigma_0 e^{-\lambda_\varsigma t}, \tag{3.7}$$

where ς_0 is a positive constant and $\lambda_\varsigma > 0$ is a positive convergence rate. We refer to $\varsigma(t)$ as the *threshold function*. The threshold function is part of the control design, and it is known by all the agents. An update for agent i is also scheduled for the instants when $w_{i,j}(t)$ for some j or $p_i(t)$ has changed its value. The rule for scheduling the updates can therefore be formalized as follows:

$$\begin{aligned} t_{i,k+1} = \inf\{t \geq t_{i,k} : \\ & w_{i,j}(t) \neq w_{i,j}(t_{i,k}^+) \text{ for some } j, \text{ or} \\ & p_i(t) \neq p_i(t_{i,k}^+), \text{ or} \\ & \|\tilde{u}_i(t)\| \geq \varsigma(t)\}, \end{aligned} \tag{3.8}$$

where

$$\tilde{u}_i(t) = u_i(t) - z_i(t). \tag{3.9}$$

Our goal is to show that the control algorithm defined by (3.2) and (3.4)–(3.9) makes the closed-loop system well-posed and attains the control objective (3.3). Well-posedness of the closed-loop system means that the sequences $\{t_{i,k}\}_{k \in \mathbb{N}_0}$ of the control updates do not exhibit Zeno behavior.

3.2 Representation as a graph

The topology of the multi-agent system (3.1) can be loosely represented as a time-varying graph $\mathcal{G}(t)$. Each agent in the system corresponds to a node in the graph, while each couple (i, j) such that $w_{i,j}(t) \neq 0$ constitutes an edge in the graph, with weight equal to $w_{i,j}(t)$. Under Assumption 3.2, such graph is undirected. The reason why this interpretation is not precise is that,

according to the definition given in Chapter 2, the weights in a graph are positive scalars, while here we just need to weights to be lower-bounded and upper-bounded. Nevertheless, interpreting the topology of the multi-agent system (3.1) as a graph allows to relate the convergence properties of the multi-agent system to the structural properties of the graph, as we will discuss in Section 3.8.

3.3 Implementation

In order to implement scheduling rule (3.8), agent i needs to know the value of the signals $w_{i,j}(t)$, $p_i(t)$ and $\tilde{u}_i(t)$ at every time instant. The signals $w_{i,j}(t)$ and $p_i(t)$ represent the topology of the information sources of agent i, therefore it is reasonable that agent i is aware of the value of these signals at any time instant. On the other hand, to compute $\tilde{u}_i(t)$ as by (3.6) and (3.9), agent i needs to know its own state $x_i(t)$, the reference $r(t)$, and the states $x_j(t)$ of the other agents. However, since all the agents and the reference have the same known dynamics, these signals can be predicted simply by integrating said dynamics. Namely, for the reference trajectory, (3.2) holds for all $t \geq 0$. Therefore, in order to compute $r(t)$ at all $t \in [t_{i,k}, t_{i,k+1})$, agent i only needs to know the initial value $r(t_{i,k})$. Moreover, agent i needs to compute $r(t)$ for $t \in [t_{i,k}, t_{i,k+1})$ only if $p_i(t_{i,k}^+) \neq 0$. In fact, if $p_i(t_{i,k}^+) = 0$, $r(t)$ does not affect $\tilde{u}_i(t)$ for $t \in [t_{i,k}, t_{i,k+1})$, see (3.4)–(3.6) and (3.9). Similarly, for the states $x_j(t)$ of the other agents, (3.1) holds for all $t \geq 0$. Therefore, in order to compute $x_j(t)$ for all $t \in [t_{i,k}, t_{i,k+1})$, agent i only needs to know the initial value $x_j(t_{i,k})$ and the values, say u_{j,h_j}, of the control signal that agent j uses within the interval $[t_{i,k}, t_{i,k+1})$. Moreover, agent i needs to compute $x_j(t)$ for $t \in [t_{i,k}, t_{i,k+1})$ only if $w_{i,j}(t_{i,k}^+) \neq 0$. In fact, if $w_{i,j}(t_{i,k}^+) = 0$, $x_j(t)$ does not affect $\tilde{u}_i(t)$ for $t \in [t_{i,k}, t_{i,k+1})$, see (3.4)–(3.6) and (3.9). This implies that when an agent j updates its control input, it has to broadcast the newly computed control input, say u_{j,h_j}, to all the other agents i such that $w_{i,j}(t_{j,h_j}^+) \neq 0$. These considerations lead us to propose the following Algorithm 3.1 as an implementation of the control algorithm (3.4)–(3.9). From Algorithm 3.1, it is clear that the proposed control algorithm requires inter-agent communication only when one of the agents updates its control input, and not at every time instant.

Algorithm 3.1. Operations executed by each agent i at a generic time instant $t \geq 0$.

1: compute $x_i(t)$ by prediction
2: if $p_i(t_{i,k}) \neq 0$, compute $r(t)$ by prediction
3: compute $x_j(t)$ by prediction for each j such that $w_{i,j}(t_{i,k}) \neq 0$

compute $\tilde{u}_i(t)$ as by (3.9)
compute $\varsigma(t)$ as by (3.7)
if $w_{i,j}(t) \neq w_{i,j}(t_{i,k})$ for some j **or** $p_i(t) \neq p_i(t_{i,k})$ **or** $\|\tilde{u}_i(t)\| \geq \varsigma(t)$ **then**
    for $j \in \mathcal{V} \setminus \{i\}$ **do**
        if $w_{i,j}(t) \neq 0$ **and** $w_{i,j}(t_{i,k}) = 0$ **then**
            acquire $x_j(t)$ from agent j
        end if
    end for
    if $p_i(t) \neq 0$ **and** $p_i(t_{i,k}) = 0$ **then**
        acquire $r(t)$
    end if
    $k \leftarrow k + 1$
    $t_{i,k} \leftarrow t$
    compute $u_{i,k}$ as by (3.5) and set it as the control input
    broadcast $u_{i,k}$ to each agent j such that $w_{j,i}(t_{i,k}) \neq 0$
end if

3.4 Main result

In order to state our main result, we need to introduce some further notation. Let

$$x(t) = [x_1(t)^\top, \dots, x_N(t)^\top]^\top, \tag{3.10}$$

$$F(t, x(t)) = [f(t, x_1(t))^\top, \dots, f(t, x_N(t))^\top]^\top, \tag{3.11}$$

$$u(t) = [u_1(t)^\top, \dots, u_N(t)^\top]^\top, \tag{3.12}$$

$$x_0 = [x_{1,0}^\top, \dots, x_{N,0}^\top]^\top. \tag{3.13}$$

With (3.10)–(3.13), the dynamics of the open-loop system (3.1) can be rewritten compactly as

$$\begin{cases} \dot{x}(t) = f(t, x(t)) + u(t), \\ x(0) = x_0. \end{cases} \tag{3.14}$$

Consider now the error signals

$$e_i(t) = r(t) - x_i(t), \tag{3.15}$$

and let

$$\begin{aligned} e(t) &= [e_1(t)^\top, \dots, e_N(t)^\top]^\top \\ &= 1_N \otimes r(t) - x(t). \end{aligned} \tag{3.16}$$

Note that the control objective (3.3) can be rewritten in terms of the error vector $e(t)$ as

$$\lim_{t \to \infty} e(t) = 0_{Nn}.$$

Let

$$L(t)_{i,j} = \begin{cases} \sum_{j=1}^{N} w_{i,j}(t) & \text{if } i = j, \\ -w_{i,j}(t) & \text{otherwise,} \end{cases} \tag{3.17}$$

$$P(t) = \operatorname{diag}(p_1(t)^\top, \dots, p_N(t)^\top), \tag{3.18}$$

$$A(t) = L(t) \otimes C + P(t) \otimes K, \tag{3.19}$$

$$\lambda(t) = \min \operatorname{eig}(A(t)). \tag{3.20}$$

Note that, under Assumption 3.2, and with $C, K \in \mathbf{S}^N_{\succ 0}$, the matrix $A(t)$ is symmetric for any $t \geq 0$, and, therefore, its minimum eigenvalue $\lambda(t)$ is well defined.

Remark 3.1. *Comparing* (3.17) *with* (2.5)*, we see that $L(t)$ can be loosely interpreted as the Laplacian of the undirected graph $\mathcal{G}(t)$ that represents the topology of the multi-agent system* (3.1)*. Recall that this interpretation is not precise, since we are not requiring that $w_{i,j}(t) \geq 0$.*

Our main result can now be formalized as the following theorem.

Theorem 3.1. *Consider the multi-agent system* (3.1)*, under the control algorithm defined by* (3.2) *and* (3.4)–(3.9)*. Let Assumptions 3.1–3.3 hold. If there exist $T > 0$ and $\varphi > \lambda_f + \lambda_\varsigma$ such that, for any $t \geq 0$,*

$$\frac{1}{T} \int_t^{t+T} \lambda(\tau) \, \mathrm{d}\tau \geq \varphi, \tag{3.21}$$

then the closed-loop system is well posed and achieves the control objective (3.3)*. In particular, the error stack vector $e(t)$ defined by* (3.16) *converges to zero exponentially with a convergence rate that is lower-bounded by the convergence rate λ_ς of the threshold function; namely, there exists $\bar{\eta} > 0$ such that*

$$\|e(t)\| \leq \bar{\eta} \exp(-\lambda_\varsigma t) \quad \forall t \geq 0.$$

Remark 3.2. *Condition* (3.21) *essentially requires that the connectivity between the reference trajectory and the agents in the network, parametrized by the minimum eigenvalue $\lambda(t)$ of $A(t)$, has an average over time that is large enough, compared to the Lipschitz constant of the agents' dynamics and to the convergence rate of the threshold function. However, condition* (3.21) *does not require $\lambda(t)$ to be large at any specific time instant.*

The proof of Theorem 3.1 is given in the next three sections of the chapter. Namely, in Section 3.5, we prove that the closed-loop system achieves

exponential convergence of the error vector $e(t)$, and in Section 3.6, we prove that the closed-loop system is well posed, in the sense that the sequence of the control updates of each agent does not exhibit Zeno behavior. Finally, in Section 3.7, we use the results obtained in the previous two sections two formalize the proof of Theorem 3.1.

3.5 Convergence proof

In order to analyze the convergence properties of the closed-loop system (3.1), (3.2) and (3.4)–(3.9), we write the dynamics of the open-loop system in terms of the error vector $e(t)$. Taking the time derivative of both sides in (3.16), and using (3.2) and (3.14), we can write the open-loop dynamics of the error signals as

$$\begin{cases} \dot{e}(t) = 1_N \otimes f(t, r(t)) - f(t, x(t)) - u(t), \\ e(0) = 1_N \otimes r_0 - x_0, \end{cases} \tag{3.22}$$

where $e_0 = e(0)$. Note that (3.6) can be rewritten in terms of the error signals (3.15) as

$$z_i(t) = \sum_{j=1}^{N} w_{i,j}(t) C(e_i(t) - e_j(t)) + p_i(t) K e_i(t). \tag{3.23}$$

Moreover, letting

$$z(t) = [z_1(t)^\top, \dots, z_N(t)^\top]^\top,$$

we can rewrite (3.23) compactly as

$$z(t) = A(t)e(t), \tag{3.24}$$

where $A(t)$ is defined in (3.19).

Substituting (3.9) and (3.24) into (3.22), we have

$$\dot{e}(t) = 1_N \otimes f(t, r(t)) - f(t, x(t)) - A(t)e(t) - \tilde{u}(t). \tag{3.25}$$

From (3.25), it is clear that convergence of the error vector $e(t)$ can be related to $f(\cdot, \cdot)$ being Lipschitz, to the eigenvalues of $A(t)$, and to the boundedness of $\tilde{u}(t)$. This is formalized in the following lemma.

Lemma 3.1. *If $\|\tilde{u}_i(t)\| \leq \varsigma(t)$ for all $t \in [0, T]$ for all $i \in \mathcal{V}$, where $T > 0$, then, under Assumption 3.1, we have $\|e(t)\| \leq \eta(t)$ for all $t \in [0, T]$, where $\eta(t)$ satisfies*

$$\begin{cases} \dot{\eta}(t) = (\lambda_f - \lambda(t))\eta(t) + \sqrt{N}\varsigma(t), \\ \eta(0) = \eta_0, \end{cases} \tag{3.26}$$

where $\eta_0 = \|e_0\|$ and $\lambda(t)$ is defined by (3.20)

Proof. Consider the function

$$V(t) = \frac{1}{2} e(t)^\top e(t). \tag{3.27}$$

Note that we shall not refer to $V(t)$ as to a candidate Lyapunov function, since we are not going to use any Lyapunov theorem. Taking the time derivative of both sides, and using (3.25), we have

$$\begin{aligned} \dot{V}(t) =& e(t)^\top \dot{e}(t) \\ =& e(t)^\top (1_N \otimes f(t, r(t)) - f(t, x(t)) - A(t)e(t) - \tilde{u}(t)) \\ =& \sum_{i=1}^{N} e_i(t)^\top (f(t, r(t)) - f(t, x_i(t))) \\ & - e(t)^\top A(t) e(t) - e(t)^\top \tilde{u}(t). \end{aligned} \tag{3.28}$$

The terms on the right-hand side of (3.28) can be bounded as follows. By Assumption 3.1, we have

$$e_i(t)^\top (f(t, r(t)) - f(t, x_i(t)) \leq \lambda_f \|e_i(t)\|^2. \tag{3.29}$$

Since $A(t)$ is symmetric, we have

$$- e(t)^\top A(t) e(t) \leq -\lambda(t) \|e(t)\|^2, \tag{3.30}$$

where $\lambda(t)$ is the smallest eigenvalue of $A(t)$. Finally, if $t \in [0, T]$, by hypothesis we have $\|\tilde{u}_i(t)\| \leq \varsigma(t)$, implying

$$e(t)^\top \tilde{u}(t) \leq \|e(t)\| \sqrt{N} \varsigma(t). \tag{3.31}$$

Substituting (3.29)–(3.31) in (3.28), we have

$$\dot{V}(t) \leq (\lambda_f - \lambda(t)) \|e(t)\|^2 + \|e(t)\| \sqrt{N} \varsigma(t). \tag{3.32}$$

Now note that (3.27) can be written equivalently as $V(t) = \frac{1}{2} \|e(t)\|^2$, which taking the time derivative of both sides yields $\dot{V}(t) = \|e(t)\| \frac{\mathrm{d}\|e(t)\|}{\mathrm{d}t}$, which, in turn, compared with (3.32) yields

$$\|e(t)\| \frac{\mathrm{d}\|e(t)\|}{\mathrm{d}t} \leq (\lambda_f - \lambda(t)) \|e(t)\|^2 + \|e(t)\| \sqrt{N} \varsigma(t). \tag{3.33}$$

For any t such that $\|e(t)\| \neq 0$, (3.33) reduces to

$$\frac{\mathrm{d}\|e(t)\|}{\mathrm{d}t} \leq (\lambda_f - \lambda(t)) \|e(t)\| + \sqrt{N} \varsigma(t). \tag{3.34}$$

On the other hand, if $e(t) = 0$, we can write, for $t \in [0, T)$,

$$\frac{\mathrm{d}\|e(t)\|}{\mathrm{d}t} = \lim_{\delta t \to 0} \frac{\|e(t+\delta t)\| - \|e(t)\|}{\delta t}. \tag{3.35}$$

where $\|e(t)\| = 0$, and

$$e(t+\delta t) = \int_t^{t+\delta t} \dot{e}(\tau)\,\mathrm{d}\tau\,. \tag{3.36}$$

Substituting (3.25) into (3.36), taking norms of both sides, using the triangular inequality and Assumption 3.1, and observing that $\tilde{u}_i(t) \leq \varsigma(t)$ for all $t \in [0, T)$, we have

$$\|e(t+\delta t)\| \leq \int_t^{t+\delta t} ((\lambda_f - \lambda(\tau))\|e(\tau)\| + \sqrt{N}\varsigma(\tau))\,\mathrm{d}\tau$$

Dividing both sides by δt, taking the limit for $\delta t \to 0$, using the mean value theorem, and comparing with (3.35), we have again (3.34), which therefore applies for all $t \in [0, T)$. From (3.34), and using Gronwall's lemma [74], we have (3.26). □

Under the hypotheses of Lemma 3.1, we have $e(t) \to 0_{Nn}$ if $\eta(t) \to 0$. Therefore, we only need to prove that the closed-loop system is well posed and achieves $\eta(t) \to 0$ to prove Theorem 3.1. The following lemma gives a sufficient condition for convergence of $\eta(t)$.

Lemma 3.2. *Let $\eta(t)$ be defined by* (3.26)*, and let Assumption 3.3 hold. If* (3.21) *holds, then there exists $\bar{\eta} > 0$ such that*

$$\eta(t) \leq \bar{\eta}\exp(-\lambda_\varsigma t), \tag{3.37}$$

In particular, $\eta(t) \to 0$.

Proof. Condition (3.21) can be rewritten as

$$\int_t^{t+T} (\lambda_f - \lambda(\tau))\,\mathrm{d}\tau \leq -T\phi \quad \forall t \geq 0, \tag{3.38}$$

where $\phi = \varphi - \lambda_f > \lambda_\varsigma$. For any $t' > t$ we can write $t' = t + \nu T + \delta t$, with $\nu \in \mathbb{N}_0$ and $0 \leq \delta t < T$. Therefore, using (3.38) repeatedly, we have

$$\begin{aligned}\int_t^{t'} (\lambda_f - \lambda(\tau))\,\mathrm{d}\tau &\leq -\nu T\phi + \int_{t+\nu T}^{t'} (\lambda_f - \lambda(\tau))\,\mathrm{d}\tau \\ &= -\phi(t'-t) + \int_{t+\nu T}^{t'} (\lambda_f - \lambda(\tau))\,\mathrm{d}\tau\,.\end{aligned} \tag{3.39}$$

Under Assumption 3.3, $\lambda(\tau)$ is bounded, and therefore, the last integral in (3.39) is bounded. Hence, we can rewrite (3.39) as

$$\int_t^{t'} (\lambda_f - \lambda(\tau))\,\mathrm{d}\tau \leq -\phi(t'-t) + \xi \tag{3.40}$$

for some $\xi > 0$. The Laplace solution of (3.26) in $[0, t)$ reads, using also (3.7),

$$\begin{aligned}\eta(t) = \Phi(t,0)\eta_0 \\ + \sqrt{N}\varsigma_0 \int_0^t \Phi(t,\tau)\exp(-\lambda_\varsigma \tau)\,\mathrm{d}\tau\,,\end{aligned} \tag{3.41}$$

where

$$\Phi(t',t) = \exp\left(\int_t^{t'} (\lambda_f - \lambda(\tau))\,\mathrm{d}\tau\right). \tag{3.42}$$

Using (3.40) in (3.42), we have

$$\Phi(t',t) \leq \exp(-\phi(t'-t))\exp(\xi) \tag{3.43}$$

Using (3.43) in (3.41), we have

$$\begin{aligned}\eta(t) \leq \exp(-\phi t)\exp(\xi)\eta_0 \\ + \sqrt{N}\varsigma_0 \exp(\xi) \int_0^t \exp(-\phi(t-\tau))\exp(-\lambda_\varsigma \tau)\,\mathrm{d}\tau\,,\end{aligned} \tag{3.44}$$

Since $\phi > \lambda_\varsigma$, we have

$$\int_0^t \exp((\phi - \lambda_\varsigma)\tau)\,\mathrm{d}\tau = \frac{\exp((\phi-\lambda_\varsigma)t) - 1}{\phi - \lambda_\varsigma},$$

which substituted into (3.44) yields

$$\eta(t) \leq k'\left(\eta_0 + \sqrt{N}\varsigma_0 \exp(\xi)\frac{\exp((\phi-\lambda_\varsigma)t) - 1}{\phi - \lambda_\varsigma}\right)\mathrm{e}^{-\phi t}\,. \tag{3.45}$$

Using again $\phi > \lambda_\varsigma$, we can further bound (3.45) as (3.37), with

$$\bar{\eta} = k'\left(\eta_0 + \frac{\sqrt{N}\varsigma_0}{\phi - \lambda_\varsigma}\right).$$

Thanks to Lemmas 3.1 and 3.2, proving that the proposed control algorithm attains the objective (3.3) reduces to proving that the algorithm makes the closed-loop system well-posed, and attains $\|\tilde{u}_i(t)\| \leq \varsigma(t)$ for all $t \geq 0$ as well as (3.40). This will be the subject of the following Section 3.6.

3.6 Well-posedness proof

Well-posedness of the closed-loop systems means that the HTT generated by the control updates $\{t_{i,k}\}$ of each agent i do not exhibit Zeno behavior. In order to study this property, first observe that $\|\tilde{u}_i(t)\| \leq \varsigma(t)$ is automatically guaranteed by the scheduling rule (3.8). In fact, for each $k \in \mathbb{N}_0$ and each $i \in \mathcal{V}$, we have from (3.5) and (3.6) that $z_i(t_{i,k}) = u_{i,k}$, which by (3.9) implies

$$\tilde{u}_i(t_{i,k}) = 0.$$

Since a new update $t_{i,k+1}$ is triggered whenever $\|\tilde{u}_i(t)\| \geq \varsigma(t)$, it is not possible that $\|\tilde{u}_i(t)\| > \varsigma(t)$ for some $t \geq 0$, $i \in \mathcal{V}$. Well-posedness of the closed-loop system is formalized in the following lemma.

Lemma 3.3. *Consider the multi-agent system* (3.1)*, under the control algorithm defined by* (3.2) *and* (3.4)–(3.9)*. Under Assumptions 3.1–3.3, and* (3.40)*, the closed-loop system is well posed. In particular, the sequences $\{t_{i,k}\}_{k\in\mathbb{N}}$ of the control updates for $i \in \mathcal{V}$ do not exhibit Zeno behavior.*

Proof. Let us consider a generic agent $i \in \mathcal{V}$ within the generic time interval $[t_{i,k}, t_{i,k+1})$. By (3.5), taking the time derivative of both sides in (3.9), we have

$$\dot{\tilde{u}}_i(t) = -\dot{z}_i(t). \tag{3.46}$$

Note now that, from (3.24), we have $z_i(t) = A(t)_{n(i-1)+1:in,:}e(t)$, and moreover,

$$A(t)_{n(i-1)+1:in,:} = A(t_{i,k})_{n(i-1)+1:in,:},$$

because $w_{i,j}(t)$ for all $j \in \mathcal{V} \setminus \{i\}$ and $p_i(t)$ are constant for $t \in [t_{i,k}, t_{i,k+1})$. Therefore, $\dot{z}_i(t) = A(t_{i,k})_{n(i-1)+1:in,:}\dot{e}(t)$, which substituted in (3.46) yields

$$\dot{\tilde{u}}_i(t) = -A(t_{i,k})_{n(i-1)+1:in,:}\dot{e}(t). \tag{3.47}$$

Substituting (3.25) into (3.47), we have

$$\begin{aligned}\dot{\tilde{u}}_i(t) = &- A(t_{i,k})_{n(i-1)+1:in,:}(\\ &1_N \otimes f(t, r(t)) - f(t, x(t)) - A(t)e(t) - \tilde{u}(t)).\end{aligned} \tag{3.48}$$

Note now that, by Assumption 3.3, we have $\|A(t)\| \leq \alpha$ for some $\alpha > 0$, since all the entries of $A(t)$ are bounded. Therefore, taking norms of both sides in (3.48), using the triangular inequality, $\|\tilde{u}_j(t)\| \leq \varsigma(t)$ for all $j \in \mathcal{V}$, and Assumption 3.1, we have

$$\|\dot{\tilde{u}}_i(t)\| \leq \alpha((\lambda_f + \alpha)e(t) + \sqrt{N}\varsigma(t)). \tag{3.49}$$

Since Lemmas 3.1 and 3.2 apply, we have $\|e(t)\| \leq \bar{\eta}\,\mathrm{e}^{-\lambda_\varsigma t}$, which compared with (3.49), together with (3.7), yields

$$\|\dot{\tilde{u}}_i(t)\| \leq \alpha((\lambda_f+\alpha)\bar{\eta}+\sqrt{N}\varsigma_0)\,\mathrm{e}^{-\lambda_\varsigma t}\,. \tag{3.50}$$

Since $\tilde{u}_i(t_{i,k}) = 0$, we have $\tilde{u}_i(t) = \int_{t_{i,k}}^{t} \dot{\tilde{u}}_i(\tau)\,\mathrm{d}\tau$, which by taking norms of both sides, and using the triangular inequality yields

$$\|\tilde{u}_i(t)\| \leq \int_{t_{i,k}}^{t} \|\dot{\tilde{u}}_i(\tau)\|\,\mathrm{d}\tau\,. \tag{3.51}$$

Substituting (3.50) into (3.51), we have

$$\|\tilde{u}_i(t)\| \leq \alpha((\lambda_f+\alpha)\bar{\eta}+\sqrt{N}\varsigma_0)\frac{1-\mathrm{e}^{-\lambda_\varsigma(t-t_{i,k})}}{\lambda_\varsigma}\,\mathrm{e}^{-\lambda_\varsigma t_{i,k}}\,. \tag{3.52}$$

Note now that (3.7) can be written as

$$\varsigma(t) = \varsigma_0\,\mathrm{e}^{-\lambda_\varsigma t_{i,k}}\,\mathrm{e}^{-\lambda_\varsigma(t-t_{i,k})}\,. \tag{3.53}$$

Comparing (3.52) and (3.53), it is clear that a necessary condition for having $\|\tilde{u}_i(t)\| \geq \varsigma(t)$ is

$$\alpha((\lambda_f+\alpha)\bar{\eta}+\sqrt{N}\varsigma_0)\frac{1-\mathrm{e}^{-\lambda_\varsigma(t-t_{i,k})}}{\lambda_\varsigma} \geq \varsigma_0\,\mathrm{e}^{-\lambda_\varsigma(t-t_{i,k})},$$

which is attained if and only if $t - t_{i,k} \geq \delta > 0$, where δ satisfies

$$\alpha((\lambda_f+\alpha)\bar{\eta}+\sqrt{N}\varsigma_0)\frac{1-\mathrm{e}^{-\lambda_\varsigma\delta}}{\lambda_\varsigma} = \varsigma_0\,\mathrm{e}^{-\lambda_\varsigma\delta},$$

or equivalently

$$\delta = \ln\left(\frac{\lambda_\varsigma+\alpha((\lambda_f+\alpha)\bar{\eta}+\sqrt{N}\varsigma_0)}{\alpha((\lambda_f+\alpha)\bar{\eta}+\sqrt{N}\varsigma_0)}\right) > 0. \tag{3.54}$$

From (3.54), it is clear that two consecutive control updates due to $\|\tilde{u}_i(t)\| \geq \varsigma(t)$ are separated by a positively lower-bounded inter-event time. Then, using Lemma 2.3, we can conclude that the HTT generated by the control updates due to $\|\tilde{u}_i(t)\| \geq \varsigma(t)$ is not Zeno. From Assumption 3.3, we know that the sequence of the control updates due to $w_{i,j}(t) \neq w_{i,j}(t_{i,k}^+)$ for some $j \in \mathcal{V}$ or $p_i(t) \neq p_i(t_{i,k}^+)$ is not Zeno either. From the scheduling law (3.8), we know that the sequence $\{t_{i,k}\}_{k\in\mathbb{N}_0}$ of the control updates of agent i is the union of the sequence of the control updates due to $\|\tilde{u}_i(t)\| \geq \varsigma(t)$ and the sequence of the control updates due to $w_{i,j}(t) \neq w_{i,j}(t_{i,k}^+)$ for some $j \in \mathcal{V}$ or $p_i(t) \neq p_i(t_{i,k}^+)$. Therefore, by Lemma 2.7, the sequence $\{t_{i,k}\}_{k\in\mathbb{N}_0}$ is not Zeno. Since this is valid for all the agents $i \in \mathcal{V}$, the closed-loop system is well posed. □

Remark 3.3. *Lemma 3.3 does not guarantee that two consecutive control updates $t_{i,k}$ and $t_{i,k+1}$ are separated by a finite inter-event time. In fact, two events of the type $w_{i,j}(t) \neq w_{i,j}(t_{i,k}^+)$ or $p_i(t) \neq p_i(t_{i,k}^+)$ may occur infinitely close to each other, and also infinitely close to the events of the type $\|\tilde{u}_i(t)\| \geq \varsigma(t)$. However, a finite inter-event time is guaranteed in the particular case that the network topology is constant—that is, that the scalars $w_{i,j}(t)$ and $p_i(t)$ are constant for all $i, j \in \mathcal{V}$. This is further discussed in the following Section 3.8, which examines the particular case of networks with fixed topology.*

3.7 Proof of the main result

Using Lemma 3.3, we have that, under the scheduling rule (3.8), $\|\tilde{u}_i(t)\| \leq \varsigma(t)$ for all $t \geq 0$ and all $i \in \mathcal{V}$. Hence, using Lemmas 3.1 and 3.2, and taking $t \to \infty$, we can conclude that $\|e(t)\| \leq \eta(t) \leq \bar{\eta} \exp(-\lambda_\varsigma t) \to 0$. Therefore, the control objective (3.3) is achieved, and, in particular, $e(t)$ converges to zero exponentially.

3.8 Fixed network topologies

In this section, we consider the particular case that the topology of the networked multi-agent system (3.1) is constant, i.e., that the scalars $w_{i,j}(t) \equiv w_{i,j}$ and $p_i(t) \equiv p_i$ are constant for all $i, j \in \mathcal{V}$. In this case, condition (3.40) in Lemma 3.2 is equivalent to

$$\lambda > \lambda_f + \lambda_\varsigma, \tag{3.55}$$

where λ is the minimum eigenvalue of the (now constant) matrix A defined by (3.19). Since the eigenvalues of A scale linearly with the matrices C and K (when C and K are scaled simoultaneously), (3.55) can be satisfied by making A positive definite, and then by scaling it opportunely by scaling the matrices C and K. The following Lemma relates the positive definiteness of A to the positive definiteness of $L + P$.

Lemma 3.4. *Let $A, B \in \mathbf{S}_{\geq 0}^N$ and $C, D \in \mathbf{S}_{>0}^n$. Then $A \otimes C + B \otimes D \in \mathbf{S}_{\geq 0}^{Nn}$, and $A \otimes C + B \otimes D \in \mathbf{S}_{>0}^{Nn}$ if and only if $A + B \in \mathbf{S}_{>0}^N$.*

Proof. Since $A, B \in \mathbf{S}_{+}^N$, if $A + B \in \mathbf{S}_{++}^N$, then either $A \in \mathbf{S}_{++}^N$ or $B \in \mathbf{S}_{++}^N$ (possibly both). Consequently, $A \otimes C, B \otimes D \in \mathbf{S}_{+}^{Nn}$, and either $A \otimes C \in \mathbf{S}_{++}^{Nn}$ or $B \otimes D \in \mathbf{S}_{++}^{Nn}$. Hence, $A \otimes C + B \otimes D \in \mathbf{S}_{++}^{Nn}$. Similarly, since $A \otimes C, B \otimes D \in \mathbf{S}_{+}^{Nn}$, if $A \otimes C + B \otimes D \in \mathbf{S}_{++}^{Nn}$ then either $A \otimes C \in \mathbf{S}_{++}^{Nn}$ or $B \otimes D \in \mathbf{S}_{++}^{Nn}$ (possibly both). Therefore, either $A \in \mathbf{S}_{++}^N$ or $B \in \mathbf{S}_{++}^N$, which implies $A + B \in \mathbf{S}_{++}^N$. □

By Lemma 3.4, A can be made positive definite by making L and P positive semidefinite and $L+P$ positive definite. A sufficient condition for making L positive semidefinite is that $w_{i,j} \geq 0$ for all $i, j \in \mathcal{V}$. In fact, for $w_{i,j} \geq 0$ for all $i, j \in \mathcal{V}$, L is the Laplacian matrix of a graph, which is positive semidefinite, cfr. Lemma 2.3. On the other hand, a sufficient condition for making P positive semidefinite is that $p_i \geq 0$ for all $i \in \mathcal{V}$. The hypotheses $w_{i,j} \geq 0$ and $p_i \geq 0$ correspond to the feedback between any two agents and from the reference to each agent being either positive ($w_{i,j} > 0$ or $p_i > 0$) or absent ($w_{i,j} = 0$ or $p_i = 0$). Such hypotheses are verified in most realistic settings, while negative feedback occurs in applications featuring adversarial connections between two or more agents. Given that L and P are positive semidefinite, a sufficient condition for making $L+P$ positive definite is given by the following lemma.

Lemma 3.5. *Let $w_{i,j} = w_{j,i} \geq 0$ and $p_i \geq 0$ for all $i, j \in \mathcal{V}$. Let $\mathcal{G}$ be the undirected graph defined by the nodes $\mathcal{V}$ and the edges $\mathcal{E} = \{(j,i) \in \mathcal{V} \times \mathcal{V} : \ w_{i,j} > 0\}$, where $w_{i,j} > 0$ is also the weight of the edge (j,i). The nodes $i \in \mathcal{V}$ such that $p_i > 0$ are said to be pinned. Let L and P be defined by* (3.17) *and* (3.18)*, so that L is the Laplacian matrix of $\mathcal{G}$. Then $L+P$ is positive definite if and only if there is at least one pinned node in each connected component of $\mathcal{G}$.*

Proof. Without loss of generality, suppose that the nodes of $\mathcal{G}$ are ordered in such a way that the first n_1 nodes are in a first component, the following n_2 nodes are a the second component, etc. Then L and P are block-diagonal, with each block corresponding to one of the components. We divide the proof in two parts.

Part 1 (If $L+P$ is positive definite, then there is at least one pinned node in each component of $\mathcal{G}$.)**.** Suppose that $L+P$ is positive definite, and suppose by contradiction that there is a connected component that does not contain any pinned node. Without loss of generality, suppose that this component is the first component. Then, consider the vector $v = [1_{n_1}^\top, 0_{n_2}^\top, \dots, 0_{n_{N_c}}^\top]^\top$, where N_c is the number of components in $\mathcal{G}$. Then we have

$$v^\top (L+P) v = 1_{n_1}^\top (L_1 + P_1) 1_{n_1}, \tag{3.56}$$

where L_1 and P_1 are the blocks of L and P respectively corresponding to the first component. Since the first component does not contain any pinned node, we have $P_1 = 0_{n_1 \times n_1}$, and since L_1 is a Laplacian matrix, we have $L_1 1_{n_c} = 0$. Substituting the last two equations into (3.56), we have $v^\top (L + P) v = 0$, with $v \neq 0_N$, which is a contradiction.

Part 2 (If there is at least one pinned node in each component of $\mathcal{G}$, then $L+P$ is positive definite.)**.** Viceversa, suppose that there is at least one pinned node in each connected component of $\mathcal{G}$, and suppose by contradiction that

$L+P$ is not positive definite, i.e., that there exists a nonzero $v \in \mathbb{R}^N$ such that $v^\top(L+P)v = 0$. Since both L and P are block-diagonal with each block of L being the same size of the corresponding block of P, the previous equation implies that

$$\sum_{i=1}^{N_c} v_{(i)}^\top (L_i + P_i) v_{(i)} = 0 \tag{3.57}$$

for each $i \in \{1, \dots, N_c\}$, where N_c is the number of components in $\mathcal{G}$, $v_{(i)} \in \mathbb{R}^{n_i}$ is the restriction of v to the entries corresponding to the i-th component, and L_i and P_i are the i-th diagonal block of L and P respectively. Note that L_i and P_i are both positive semidefinite, since L_i is a Laplacian matrix and P_i is diagonal with nonnegative diagonal entries. Therefore, (3.57) implies $v_{(i)}^\top (L_i + P_i) v_{(i)}$ for all $i \in \{1, \dots, N_c\}$, which, in turn, implies

$$v_{(i)}^\top L_i v_{(i)} = 0, \tag{3.58}$$

$$v_{(i)}^\top P_i v_{(i)} = 0. \tag{3.59}$$

for all $i \in \{1, \dots, N_c\}$. Since L_i is a Laplacian matrix, (3.58) implies $v_{(i)} = \alpha 1_{n_i}$ for some $\alpha \in \mathbb{R}$, which substituted in (3.59) gives $\alpha \operatorname{tr}(P_i) = 0$. Since the entries of P_i are nonnegative, and since there is at least one pinned node in each component, we have $\operatorname{tr}(P_i) > 0$, which means that $\alpha = 0$. Hence, $v_{(i)} = \alpha 1_{n_i} = 0_{n_i}$. Since this reasoning applies to all components $i \in \{1, \dots, N_c\}$, we conclude that $v = 0_N$, which is a contradiction. □

The intuition behind Lemma 3.5 is very simple: for the multi-agent system to converge to the reference trajectory, each agent needs to have access to information originating from the reference trajectory, either by directly receiving feedback from the reference trajectory, or by receiving feedback from other agents that are influenced by the reference trajectory. As a particular case of Lemma 3.5, we have the following corollary.

Corollary 3.1. *In Lemma 3.5, if $\mathcal{G}$ is connected, then $L + P$ is positive definite if and only if there is at least one pinned node.*

Remark 3.4. *If in Lemma 3.5 we relax the assumption that the scalars $w_{i,j}$ and p_i are nonnegative, we can still write Part 1 of the proof to show that a necessary (but, in this case, not sufficient) condition for $L + P$ to be positive definite is that there is at least one node i such that $p_i \neq 0$ in each component of the graph.*

When the topology of the networked system is fixed, the proposed control algorithm comes with a guaranteed constant lower bound for the inter-event times between two consecutive control updates $t_{i,k}$ and $t_{i,k+1}$ of the same agent. This property is immediately deduced by the proof of Lemma 3.3, observing that in this case the control updates can only be triggered by events

of the kind $\|\tilde{u}_i(t)\| \geq \varsigma(t)$. In particular, the lower bound for the inter-event times is given by $\delta > 0$ defined by (3.54).

3.9 Numerical simulations

To illustrate the effectiveness of the proposed control algorithm, we apply it to a simulated network of $N = 5$ identical Chua oscillators [75]. The individual dynamics of each oscillator is described by

$$f(x) = \begin{bmatrix} a(x_2 - x_1 - \gamma(x_1)) \\ x_1 - x_2 + x_3 \\ -bx_2 \end{bmatrix},$$

with $x = [x_1, x_2, x_3]^\top \in \mathbb{R}^3$, where $a, b, m_0, m_1 \in \mathbb{R}$ and $\gamma : \mathbb{R} \to \mathbb{R}$, namely,

$$\gamma(y) = m_1 y + \frac{1}{2}(m_0 - m_1)(|y + 1| - |y - 1|).$$

Choosing $a = b = 0.9$, $m_0 = -1.34$, and $m_1 = -0.73$, the oscillators are globally Lipschitz with Lipschitz constant $\lambda_f = 3.54$ (see [34] for further details). Let $C = 5I_3$ and $K = 30I_3$. All the agents are connected to each other with $w_{i,j}(t) \equiv 1$. Our simulation is set on the time interval $[0, 30]$. At the beginning of the simulation, we set $p_1(0) = p_2(0) = 1$, while $p_i(0) = 0$ for $i \in \{3, 4, 5\}$, which yields $\lambda(0) = 6.14$. At $t = 0.75$, we set $p_1(t) = 0$, so that $\lambda(t) = 2.88$. At $t = 0.90s$, we set $p_2(t) = 0$, which yields $\lambda(t) = 0$. At $t = 1.0$ the original values of the signals $p_i(t)$ are restored, and the cycle is repeated for every time unit. With this setting, it is clear that Assumptions 3.2 and 3.3 are satisfied. Also, we can verify that condition (3.40) is satisifed with $\gamma = 1.5$ and $k = 0$. Figure 3.1 provides an illustration of the graph underlying the simulated network.

For the threshold function (3.7), we choose $\varsigma_0 = 1$ and $\lambda_\varsigma = 0.3$, so that the hypotheses of Lemma 3.2 are satisifed. For each agent, the initial conditions are chosen within the domain of attraction of a Chua oscillator with the chosen parameters a, b, m_0, m_1.

Some results of the simulation are illustrated in Figures 3.2–3.4 and Table 3.1. Namely, Figure 3.2 illustrates the evolution of the second state variable $x_i^{(2)}$ for each agent $i \in \{1, \ldots, 5\}$ over the whole simulation. This result confirms that the control objective (3.3) is achieved, i.e., the state of each agent converges asymptotically to the reference trajectory. As a term of comparison, Figure 3.3 illustrates the evolution of the same state variables, with the same initial conditions, when no control input is applied ($u_i(t) \equiv 0_3$). Figure 3.4 illustrates the time instants when each agent updates its control input within the interval $[0, 1]$, and Table 3.1 illustrates the average inter-event

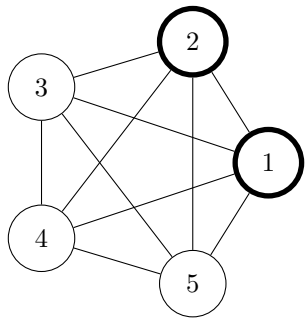


Figure 3.1. An illustration of the graph underlying the simulated network. Each node in the graph represents a Chua oscillator. The nodes with thicker contour represent the oscillators that receive feedback from the reference trajectory during part of the simulation.

Table 3.1. Average inter-event time for each Chua oscillator over the time interval $[0.0, 30.0]$, with the proposed control algorithm applied.

NODE	AVERAGE $t^i_{i,k+1} - t_{i,k}$
1	0.061
2	0.054
3	0.115
4	0.123
5	0.115

time for each agent over the whole simulation. These results confirm that the closed-loop system is not Zeno.

3.10 Summary

In this chapter, we have proposed an algorithm for event-triggered pinning synchronization of complex networks of nonlinear agents with switching topologies. We have found sufficient conditions under which Zeno behavior of the closed-loop system is excluded, and the synchronization objective is achieved. We have also shown that the error stack vector that represents the global distance of the system from the synchronization vanishes exponentially. A constant lower bound on the inter-event times has been provided for the case of networks with time-invariant topologies. Numerical simulations have been presented to validate the theoretical results.

Some viable extensions of this work include the application of the pro-

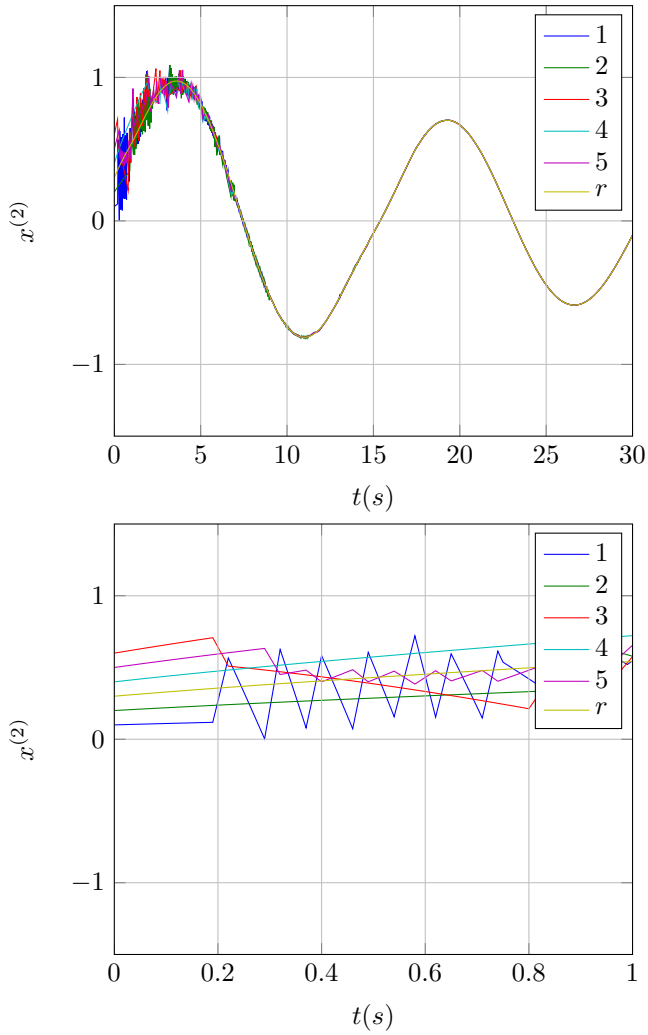


Figure 3.2. Evolution of the state variable $x_i^{(2)}(t)$ for each Chua oscillator $i \in \{1, \ldots, 5\}$ and of $r^{(2)}(t)$ for the reference trajectory, over $t \in [0, 30]$ (above) and $t \in [0, 1]$ (below), with the proposed control algorithm applied. As predicted by Theorem 3.1, the state of each agent converges to the reference trajectory.

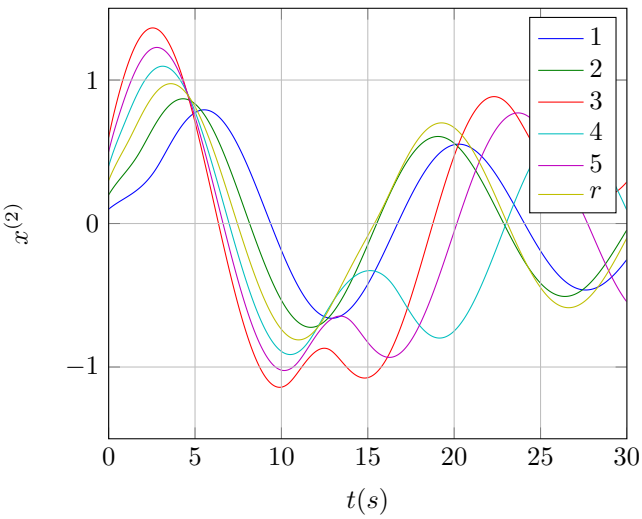


Figure 3.3. Evolution of the state variable $x_i^{(2)}$ for each Chua oscillator $i \in \{1, \ldots, 5\}$ and of $r^{(2)}$ for the reference trajectory, over $t \in [0, 30]$, with no control input applied ($u_i(t) \equiv 0_3$). The states of the agents do not converge to the reference trajectory.

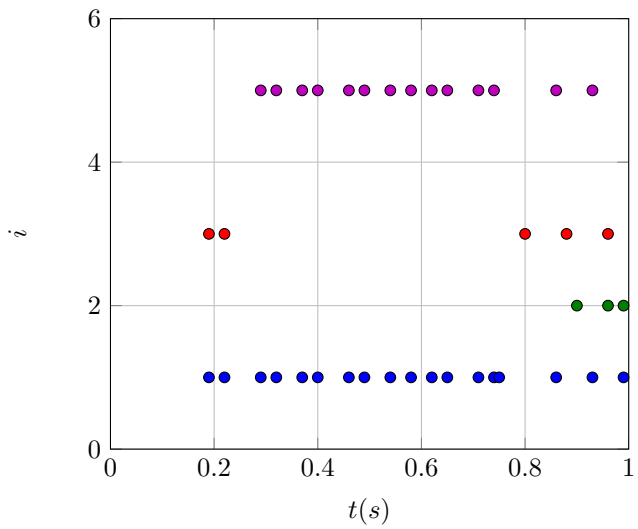


Figure 3.4. Control updates for each Chua oscillator $i \in \{1, \ldots, 5\}$ over the time interval $[0, 1]$, with the proposed control algorithm applied.

posed algorithm to more general classes of networks, such as networks with asymmetric couplings among the agents, and networks where errors in the communication can occur, such as delays and packet drops.

Chapter 4

Cloud-supported multi-agent coordination

In some realistic applications of multi-agent systems, inter-agent communication is completely or almost completely interdicted. This challenge arises, for example, in the coordination of a fleet of autonomous underwater vehicles (AUVs) [36, 37, 66]. Because of their severely limited communication, sensing, and localization capabilities, underwater vehicles are virtually isolated systems. Underwater communication and positioning may be implemented by means of battery-powered acoustic modems, but such devices are expensive, limited in range, and power-hungry. Inertial sensor for underwater positioning are prohibitively expensive in most practical scenarios. Moreover, GPS is not available underwater, and a vehicle needs to surface whenever it needs to get a position fix [76].

In the control architecture described in this chapter, inter-agent communication is substituted by the use of a shared information repository hosted on a cloud. Each agent schedules its own accesses independently, and does not need to be alert for information broadcast by other agents. When an agent accesses the repository, it uploads some data packets, and downloads other packets that were previously deposited by other agents. Therefore, each agent receives outdated information about the state of the other agents. The control law and the rule for scheduling the cloud accesses are designed to guarantee that the closed-loop system is well-posed and achieves a given coordination objective, even if each agent receives only outdated information about the state of the other agents. Our motivating example is a waypoint generation algorithm for AUVs, which, as described above, represents a challenging application, since underwater communication is interdicted.

The use of a shared information repository in multi-agent control tasks is subject to recent, but growing, research attention. In [63], the authors employ

asynchronous communication with a base station to address a multi-agent coverage control problem. In [77], the authors present a cloud-supported approach to multi-agent optimization. In our previous work [66], we presented a cloud-supported control strategy for the rendezvous of single-integrator agents. In [78], the authors employ a similar approach to achieve asymptotic rendezvous in a disturbance-free scenario. In this chapter, we introduce cloud support for multi-agent systems with second-order dynamics. We consider both persistent and vanishing disturbances, which lead to approximate and perfect coordination, respectively. In both cases, we show that the closed-loop system is well posed (meaning that the sequence of the cloud accesses does not exhibit Zeno behavior [73]), and achieves the control objective. Our analysis extends the use of the edge Laplacian [71, 79] to second-order multi-agent systems on directed graphs, which allows us to consider control tasks with asymmetric information flow among the agents, such as leader-following tasks.

The rest of this chapter is organized as follows. In Sections 4.1 and 4.2, we present the system model and outline the control strategy. In Section 4.3, we state our main result, whose proof is given in Sections 4.4–4.6. Section 4.7 corroborates the theoretical results by presenting two numerical simulations of the proposed control strategy. Finally, in Section 4.8, we present our conclusions and some directions for future research.

4.1 System model

In this section, we describe our control architecture, by defining the agents' model, the communication between the agents and the cloud repository, the agents' control inputs, and the control objective.

4.1.1 Agent Model

We consider a set $\mathcal{V} = \{1, \dots, N\}$ of N agents. The position and velocity of agent i are denoted respectively as $p_i, v_i \in \mathbb{R}^n$. For the sake of generality, we consider the generic agent dimension $n \in \mathbb{N}$. However, in our motivating example of planar AUV coordination, we have $n = 2$. The agents move according to the following equations:

$$\dot{p}_i(t) = v_i(t), \tag{4.1a}$$
$$\dot{v}_i(t) = u_i(t) + d_i(t), \tag{4.1b}$$

for $i = 1, \dots, N$, where $u_i(t)$ is a control input and $d_i(t)$ is a disturbance signal. We denote $p(t) = [p_1(t)^\top, \dots, p_N(t)^\top]^\top$, and similarly for $v(t)$, $u(t)$

and $d(t)$, so that (4.1) can be rewritten as

$$\dot{p}(t) = v(t), \tag{4.2a}$$
$$\dot{v}(t) = u(t) + d(t). \tag{4.2b}$$

The control objective is for all the agents to converge to the same positions and velocities within a given tolerance. Such objective is formalized mathematically later in this section.

Assumption 4.1. *The disturbance signals $d_i(t)$ in* (4.1b) *satisfy $\|d_i(t)\| \leq \delta(t)$, where*

$$\delta(t) = (\delta_0 - \delta_\infty)e^{-\lambda_\delta t} + \delta_\infty, \tag{4.3}$$

for some $0 \leq \delta_\infty \leq \delta_0$ and $\lambda_\delta > 0$.

Assumption 4.1 allows to consider both scenarios where only a constant upper bound is known ($\delta_0 = \delta_\infty$) and scenarios where the disturbances slowly vanish ($\delta_\infty = 0$), which makes it possible to reach asymptotic convergence.

4.1.2 Cloud Repository

The agents cannot exchange any information directly, but can only upload and download information on a shared repository hosted on a cloud. The cloud is accessed intermittently by each agent and asynchronously by different agents. A motivating application is a group of AUVs that can only communicate with a remote repository when they are on the water surface, while they are isolated when they are underwater. When an agent accesses the cloud, it also has access to a sampled measurement of its own position and velocity. In our motivating application, this corresponds to the underwater vehicles being able to access GPS while they are on the water surface. The time instants when agent i accesses the cloud are denoted $t_{i,k}$, $k \in \mathbb{N}$, and by convention $t_{i,0} = 0$ for all the agents. For convenience, we denote $l_i(t)$ the index of the most recent access time of agent i before time t—that is,

$$l_i(t) = \max\{k \in \mathbb{N} : \ t_{i,k} \leq t\}. \tag{4.4}$$

The position and velocity measurement obtained by agent i upon the time instant $t_{i,k}$ are denoted $p_{i,k}$ and $v_{i,k}$ respectively. The control signals $u_i(t)$ are held constant between two consecutive cloud accesses:

$$u_i(t) = u_{i,k} \quad \forall t \in [t_{i,k}, t_{i,k+1}). \tag{4.5}$$

When an agent accesses the cloud, it uploads data that other agents may download later, when they, in turn, access the cloud. Namely, when agent i

Table 4.1. Data contained in the cloud at a generic time instant $t \geq 0$. The i-th column corresponds to the latest packet uploaded by agent i.

agent	1	2	...	N
last access	$t_{1,l_1(t)}$	$t_{2,l_2(t)}$	...	$t_{N,l_N(t)}$
position	$p_{1,l_1(t)}$	$p_{2,l_2(t)}$	...	$p_{N,l_N(t)}$
velocity	$v_{1,l_1(t)}$	$v_{2,l_2(t)}$	...	$v_{N,l_N(t)}$
control	$u_{1,l_1(t)}$	$u_{2,l_2(t)}$	...	$u_{N,l_N(t)}$
next access	$t_{1,l_1(t)+1}$	$t_{2,l_2(t)+1}$	...	$t_{N,l_N(t)+1}$

accesses the cloud at time $t_{i,k}$, it uploads a packet containing the following information: the current time $t_{i,k}$, the position and velocity measurements $p_{i,k}$ and $v_{i,k}$, the value $u_{i,k}$ of the control input that is going to be applied in the time interval $[t_{i,k}, t_{i,k+1})$, and the time $t_{i,k+1}$ of the next access. The data packet may overwrite the packet that was uploaded on the previous access, avoiding that the amount of data contained in the cloud grow over time, since, at each time instant, the cloud only contains the data that each agent has uploaded upon its latest access. The data contained in the cloud at a generic time instant is represented in Table 4.1.

To achieve inter-agent coordination, each agent needs to download information about a subset of the other agents. For each agent i, we denote as $\mathcal{N}_i \subseteq \mathcal{V} \setminus \{i\}$ the subset of the agents whose information is downloaded by agent i.[1] Namely, when agent i accesses the cloud at time $t_{i,k}$, it downloads and stores the latest packet uploaded by each agent $j \in \mathcal{N}_i$. This information, together with the measurements $p_{i,k}$ and $v_{i,k}$, is used by agent i to compute its control input $u_{i,k}$ for the upcoming time interval $[t_{i,k}, t_{i,k+1})$, and to schedule the next cloud access $t_{i,k+1}$. In order to better illustrate the access sequence and the corresponding notation, Figure 4.1 illustrates a possible sequence of cloud accesses on the time line. Note that, in the scenario depicted in Figure 4.1, within the interval $[t_{i,k}, t_{i,k+1})$, while agent i is underwater and, therefore, isolated, agent j surfaces and changes its control input more than one time. Agent i does not know the control input that agent j will apply after t_{j,h_j+1}, nor it knows whether agent j will surface more times after t_{j,h_j+1}. The scheduling algorithm is able to guarantee the overall system's convergence in spite of these limitations.

The cloud uses the packets that it is storing to compute information about the global state of the system. Such information can be downloaded by the

[1] The number $\mathcal{N}_i$ of other agents whose information is downloaded by agent i may be chosen according to the available bandwidth or to agent i's computational capabilities.

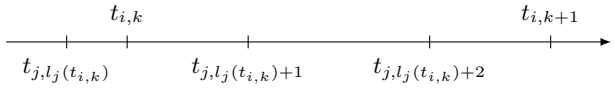


Figure 4.1. Excerpt of a possible sequence of cloud accesses on the time line. Recall that $t_{j,l_j(t)}$ denotes the most recent cloud access of agent j with respect to the time t. Note that there can be more than one access of agent j between two consecutive accesses of agent i.

agents when they access the cloud, and used to improve the coordination performance. Here, we consider the following case: when agent i accesses the cloud at time $t_{i,k}$, it also receives the positive scalar $\hat{\eta}(t_{i,k})$ representing an estimate of how far the system is from reaching the control objective. This estimate is formally defined in Section 4.4. The operations that each agent i performs upon each cloud access $t_{i,k}$ are summarized in the following Algorithm 4.2.

Algorithm 4.2. Operations executed by agent i at $t_{i,k}$.

measure position $p_{i,k}$
measure velocity $v_{i,k}$
for $j \in \mathcal{N}_i$ **do**
  download packet $\{t_{j,l_j}, p_{j,l_j}, v_{j,l_j}, u_{j,l_j}, t_{j,l_j+1}\}$
end for
receive $\hat{\eta}(t_{i,k})$ from the cloud
compute control input $u_{i,k}$
schedule next access $t_{i,k+1}$
upload packet $\{t_{i,k}, p_{i,k}, v_{i,k}, u_{i,k}, t_{i,k+1}\}$

Remark 4.1. *In most existing self-triggered control protocols for multi-agent coordination, when one agent updates its control input, such information is broadcast immediately to that agent's neighbors, which requires the neighbors to always be alert for possibly coming information. This requirement is relaxed in the proposed cloud-supported framework.*

4.1.3 Controller

The controls $u_{i,k}$, with $i \in \mathcal{V}$, are computed as follows:

$$u_{i,k} = \sum_{j \in \mathcal{N}_i} w_{ij}(k_p(\hat{p}_j(t_{i,k}) - p_{i,k}) + k_v(\hat{v}_j(t_{i,k}) - v_{i,k})), \tag{4.6}$$

$$\hat{v}_j(t) = v_{j,l_j(t)} + u_{j,l_j(t)}(t - t_{j,l_j(t)}), \tag{4.7}$$

$$\hat{p}_j(t) = p_{j,l_j(t)} + v_{j,l_j(t)}(t - t_{j,l_j(t)}) + \frac{1}{2}u_{j,l_j(t)}(t - t_{j,l_j(t)})^2, \tag{4.8}$$

where $k_p, k_v > 0$ are control gains and $w_{ij} > 0$ represents the strength of the influence of agent j on agent i. The values $\hat{p}_j(t_{(i,k)})$ and $\hat{v}_j(t_{(i,k)})$ represent the estimates of, respectively, the position and the velocity of agent j at time $t_{i,k}$. Note that, in order to compute such estimates, agent i only needs the data downloaded from the cloud at time $t_{i,k}$, and it is not necessary to communicate directly with agent j.

The sets $\mathcal{N}_1, \ldots, \mathcal{N}_N$ and the scalars w_{ij} induce a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$ over the set $\mathcal{V}$ of the agents, where $\mathcal{N}_i$ is the set of the neighbors of i and w_{ij} is the weight of edge (j, i). We are going to refer to this graph as the *network graph*. Throughout the chapter, we assume that the network graph contains a spanning tree.

Assumption 4.2. *The network graph contains a spanning tree.*

4.1.4 Control Objective

As we have anticipated, the control objective is that the agents synchronize their positions and velocities. This objective can be formalized in a convenient way if we exploit Assumption 4.2. Denote the spanning tree of the network graph as $\mathcal{T}$ and let $\mathcal{C} = \mathcal{E} \setminus \mathcal{T}$. Without loss of generality, let the edges that are in $\mathcal{T}$ be indexed from 1 to $N - 1$ and the edges that are in $\mathcal{C}$ be indexed from N to M, and partition the incidence matrix and weight matrix accordingly as $B = [B_\mathcal{T}, B_\mathcal{C}]$ and $W = [W_\mathcal{T}, W_\mathcal{C}]$. Denote $x(t) = (B_\mathcal{T}^\top \otimes I_n)p(t)$ and $y(t) = (B_\mathcal{T}^\top \otimes I_n)v(t)$. In other words, $x(t) = [x_1(t)^\top, \ldots, x_{N-1}(t)^\top]^\top$, where each $x_i(t) = p_{\mathrm{head}(i)}(t) - p_{\mathrm{tail}(i)}$ is the difference between the positions of two agents whose indices constitute an edge in $\mathcal{T}$. Similarly, $y(t) = [y_1(t)^\top, \ldots, y_{N-1}(t)^\top]^\top$, where $y_i(t) = v_{\mathrm{head}(i)}(t) - v_{\mathrm{tail}(i)}$. Let $\xi(t) = [x(t)^\top, y(t)^\top]^\top$. Note that, for each $i, j \in \mathcal{V}$, $p_j(t) - p_i(t)$ can be written as a linear combination of the variables $x_i(t)$ for $i \in \{1, \ldots, M\}$, and, similarly, $v_j(t) - v_i(t)$ can be written as a linear combination of the variables $y_i(t)$. Therefore, the control objective can be formalized in terms of $\xi(t)$ as follows.

Definition 4.1. *Consider the multi-agent system* (4.2) *and the associated network graph $\mathcal{G}$. We say that the multi-agent system achieves* practical convergence *with*

tolerance $\epsilon \geq 0$ *if*

$$\limsup_{t\to\infty} \|\xi(t)\| \leq \epsilon.$$

In particular, if the system achieves practical convergence with tolerance $\epsilon = 0$*, we say that the system achieves* asymptotic convergence*.*

4.2 Self-triggered cloud access scheduling

Each agent schedules its own access to the cloud recursively, i.e., agent i schedules the access $t_{i,k+1}$ when it accesses the cloud at time $t_{i,k}$. The scheduling is based on comparing two time-varying functions of the data downloaded from the cloud with a given threshold function. The threshold function is chosen as

$$\varsigma(t) = \varsigma_\infty + (\varsigma_0 - \varsigma_\infty)e^{-\lambda_\varsigma t}, \tag{4.9}$$

with $\lambda_\varsigma > 0$ and $0 \leq \varsigma_\infty < \varsigma_0$. To define the scheduling rule, we need to introduce some additional notation. Let $\hat{p}(t) = [\hat{p}_1(t)^\top, \dots, \hat{p}_N(t)^\top]^\top$, where $\hat{p}_i(t)$ is defined in (4.8), and similarly for $\hat{v}(t)$. Let

$$\hat{x}(t) = (B_\mathcal{T} \otimes I_n)\hat{p}(t), \tag{4.10}$$
$$\hat{y}(t) = (B_\mathcal{T} \otimes I_n)\hat{v}(t), \tag{4.11}$$
$$\hat{\xi}(t) = [\hat{x}(t)^\top, \hat{y}(t)^\top]^\top. \tag{4.12}$$

Moreover, let

$$\begin{aligned}
\Delta_i(t) &= \int_{t_{i,l_i(t)}}^{t} \int_{t_{i,l_i(t)}}^{\tau} \delta(\sigma)\,\mathrm{d}\sigma\,\mathrm{d}\tau + \int_{t_{i,l_i(t)}}^{t} \delta(\tau)\,\mathrm{d}\tau\,, \\
\Delta(t) &= [\Delta_1(t), \dots, \Delta_N(t)]^\top, \\
\hat{\eta}(t) &= \|\hat{\xi}(t)\| + \|B_\mathcal{T}\| \cdot \|\Delta(t)\|,
\end{aligned} \tag{4.13}$$

Note that (4.13), evaluated for $t = t_{i,k}$, defines the estimate $\hat{\eta}(t_{i,k})$ that agent i receives from the cloud at time $t_{i,k}$. Moreover, let

$$F_{e,r} = \begin{bmatrix} 0_{(N-1)\times(N-1)} & I_{N-1} \\ -k_p R & -k_v R \end{bmatrix}, \tag{4.14}$$

where k_p and k_v are the control gains in (4.6), R is the reduced edge Laplacian of the network graph, and

$$\lambda = -\max\{\mathrm{Re}(s) : s \in \mathrm{eig}(F_{e,r})\}. \tag{4.15}$$

Consider the function

$$\eta(t, \eta_0) = e^{-\lambda(t-t_0)}\eta_0 + \sqrt{N}\|B_\mathcal{T}\| \int_{t_0}^{t} e^{-\lambda(t-\tau)}(\varsigma(\tau) + \delta(\tau))\,\mathrm{d}\tau\,, \tag{4.16}$$

and the coefficients

$$\beta_i = \left(\sqrt{k_p^2 + k_v^2}\right)\left\|(W_{\mathcal{T}} + W_{\mathcal{C}}T^\top)_i\right\|, \tag{4.17}$$

$$\nu_i = \max_{j:i\in\mathcal{N}_j}\left\{\sum_{q\in\mathcal{N}_j} w_{qj}\right\}, \tag{4.18}$$

where $(W_{\mathcal{T}} + W_{\mathcal{C}}T^\top)_i$ denotes the i-th row of $(W_{\mathcal{T}} + W_{\mathcal{C}}T^\top)$. Finally, choose α such that $0 < \alpha < 1$. Then, each agent schedules the cloud accesses as follows:

$$t_{i,k+1} = \inf\left\{t > t_{i,k} : \sigma_{i,k}(t) \geq \varsigma(t) \vee \Omega_{i,k}(t) \geq \frac{\alpha}{\nu_i}\varsigma(t)\right\}, \tag{4.19}$$

where

$$\Omega_{i,k}(t) = k_p \int_{t_{i,k}}^{t}\int_{t_{i,k}}^{\tau} \delta(\sigma)\mathrm{d}\sigma\,\mathrm{d}\tau + k_v\int_{t_{i,k}}^{t}\delta(\tau)\,\mathrm{d}\tau\,, \tag{4.20}$$

$$\begin{aligned}
\sigma_{i,k}(t) = \Bigg\| &\left(\sum_{j\in\mathcal{N}} w_{ij}\right)\big(k_v(t-t_{i,k})u_{i,k} \\
&+ k_p((t-t_{i,k})v_{i,k} + (1/2)(t-t_{i,k})^2 u_{i,k})\big) \\
&+ \sum_{j\in\mathcal{N}_i} w_{ij}(k_v(t'_{j,h_j} - t_{i,k})u_{j,h_j} \\
&+ k_p((t-t_{i,k})v_{j,h_j} \\
&+ (1/2)(t'_{j,h_j} + t_{i,k} - 2t_{j,h_j})(t'_{j,h_j} - t_{i,k})u_{j,h_j} \\
&+ (t''_{j,h_j} - t_{j,h_j+1})(t_{j,h_j+1} - t_{j,h_j})u_{j,h}))\Bigg\| \\
&+ \sum_{j\in\mathcal{N}_i} w_{ij}\left(\int_{t_{j,h_j+1}}^{t''_{j,h_j}} \mu_j^{i,k}(\tau)\,\mathrm{d}\tau\right. \\
&\left.+ \int_{t_{j,h_j+1}}^{t''_{j,h_j}}\int_{t_{j,h_j+1}}^{\tau} \mu_j^{i,k}(\theta)\,\mathrm{d}\theta\,\mathrm{d}\tau\right) \\
&+ \left(\sum_{j\in\mathcal{N}_i} w_{ij}\right)\Omega_{i,k}(t) + \sum_{j\in\mathcal{N}_i} w_{ij}\Omega_{j,h_j}(t),
\end{aligned} \tag{4.21}$$

$$\mu_j^{i,k}(t) = \beta_j\eta(t,\hat{\eta}(t_{i,k})) + \varsigma(t), \tag{4.22}$$

$$t'_{j,h_j} = \min\{t, t_{j,h_j+1}\}, \tag{4.23}$$

$$t''_{j,h_j} = \max\{t, t_{j,h_j+1}\}, \tag{4.24}$$

and where we have denoted $h_j = l_j(t_{i,k})$ for brevity. Recall here that $l_i(t)$ is defined by (4.4). The scheduling rule (4.19)–(4.24) can be interpreted as

follows. After the access $t_{i,k}$, $\Omega_{i,k}(t)$ represents an upper bound on the part of $\|\tilde{u}_i(t)\|$ that is due to the disturbances that have acted on agent i in the interval $[t_{i,k}, t)$. This upper bound is kept under a threshold (here set to $(\alpha/\nu_i)\varsigma(t)$), so that the information deposited by agent i in the cloud can be used to predict its position and velocity within a certain error bound. Similarly, $\sigma_{i,k}(t)$ represents an upper bound on the mismatch $\|\tilde{u}_i(t)\|$, which takes into account both the effect of the disturbances $d_j(t)$ with $j \in \mathcal{N}_i \cup \{i\}$, the piecewise-constant nature of the control signals $u_j(t)$ with $j \in \mathbb{N}_i \cup \{i\}$, and the fact that, for $t > t_{j,h_j+1}$, the current value of u_{j,h_j+1} of $u_j(t)$ is not known by agent i (agent i has downloaded u_{j,h_j} in the latest packet, but has no information on the choice of the control that agent j will adopt at time t_{j,h_j+1}). Indeed, for $t > t_{j,h_j+1}$, agent i uses $\mu_j^{i,k}(t)$ as an upper-bound for $u_j(t)$, hence exploiting the estimate $\hat{\eta}(t_{i,k})$. The upper bound $\sigma_{i,k}(t)$ is kept under $\varsigma(t)$, so that the hypotheses of Lemma 4.2 are satisfied. A new cloud access is triggered when either of $\Omega_{i,k}(t)$ or $\sigma_{i,k}(t)$ are about to cross the assigned threshold. Note that (4.19)–(4.24) can be evaluated by agent i when it accesses the cloud (i.e., at time $t_{i,k}$) and do not require communication with the other agents.

4.3 Main result

Our main result is formalized as the following theorem.

Theorem 4.1. *Consider the multi-agent system* (4.1)*, with control law* (4.6)–(4.8) *and cloud accesses scheduled by* (4.19)–(4.24)*. Let Assumptions 4.1 and 4.2 hold, and let k_p and k_v be such that $F_{e,r}$ is Hurwitz. If $\varsigma_\infty > 0$, the closed-loop system does not exhibit Zeno behavior and achieves practical convergence with radius*

$$\epsilon = \frac{\sqrt{N}\|B_{\mathcal{T}}\|(\varsigma_\infty + \delta_\infty)}{\lambda}, \tag{4.25}$$

where ς_∞ is the asymptotic value of the threshold function (4.9)*, δ_∞ is the asymptotic value of the disturbance bound* (4.3)*, and λ is defined in* (4.15)*. If $\delta_\infty = 0$, $\varsigma_\infty = 0$ and $\lambda_\varsigma < \min\{\lambda, \lambda_\delta\}$, then the closed-loop system does not exhibit Zeno behavior and achieves asymptotic convergence.*

Remark 4.2. *Note that our convergence result* (4.25) *is similar to that obtained in related works on event-triggered coordination of multi-agent system, see for example [65]. Here, however, convergence is obtained by using an asynchronously accessed repository, rather than by direct inter-agent communication.*

Remark 4.3. *Note that, under Assumption 4.2, we can always choose k_p and k_v such that $F_{e,r}$ is Hurwitz. The proof of this remark is formalized as follows.*

Proof. Consider the matrix

$$F = \begin{bmatrix} 0_{N\times N} & I_N \\ -k_p L & -k_v L \end{bmatrix},$$

where L is the Laplacian matrix of the graph $\mathcal{G}$. A well known result in multi-agent coordination is that, under Assumption 4.2, k_p and k_v can always be chosen in such a way that F has exactly $2(N-1)$ eigenvalues with negative real parts (counted with their multiplicities) and a double eigenvalue in zero - see for example [80]. But $F_{e,r} \in \mathbb{R}^{2(N-1)\times 2(N-1)}$, therefore, it has exactly $2(N-1)$ eigenvalues (counted with their multiplicity). Therefore, if we show that F and $F_{e,r}$ have the same nonzero eigenvalues with the same multiplicities, then we can conclude that $F_{e,r}$ is Hurwitz.

The characteristic polynomial of F is

$$\mathcal{P}(\lambda) = \det(\lambda I_{2N} - F) = \det\big(\lambda^2 I_N + (\lambda k_v + k_p)L\big),$$

which for $\lambda \neq 0$ can be written as

$$\mathcal{P}(\lambda) = \lambda^{2N} \det\big(I_N + (\lambda k_v + k_p)/\lambda^2 L\big). \tag{4.26}$$

Now consider the matrix

$$F_e = \begin{bmatrix} 0_{M\times N} & I_M \\ -k_p E & -k_v E \end{bmatrix},$$

where E is the edge Laplacian of $\mathcal{G}$. Similarly as done for F, we can compute the characteristic polynomial of F_e as

$$\mathcal{P}_e(\lambda) = \det\big(\lambda^2 I_M + (\lambda k_v + k_p)E\big), \tag{4.27}$$

which for $\lambda \neq 0$ can be rewritten as

$$\mathcal{P}_e(\lambda) = \lambda^{2M} \det\big(I_M + (\lambda k_v + k_p)/\lambda^2 E\big). \tag{4.28}$$

Since $L = WB^\top$ and $E = B^\top W$, by (4.26) and (4.28) and Sylvester's determinant identity, we have $\mathcal{P}(\lambda)/\lambda^{2N} = \mathcal{P}_e(\lambda)/\lambda^{2M}$ for any $\lambda \neq 0$, which implies that F and F_e have the same nonzero eigenvalues with the same multiplicity. Therefore, we only need to prove that F_e and $F_{e,r}$ have the same nonzero eigenvalues with the same multiplicity. Consider the matrix

$$S = \begin{bmatrix} I_{N-1} & 0_{(N-1)\times(M-N+1)} \\ -T^\top & I_{M-N+1} \end{bmatrix},$$

and note that

$$SES^{-1} = \begin{bmatrix} R & * \\ 0_{(M-N+1)\times(N-1)} & 0_{(M-N+1)} \end{bmatrix}. \tag{4.29}$$

Multiplying the right-hand side of (4.27) by $\det(S)\det\big(S^{-1}\big) = 1$, and using (4.29), we have

$$\begin{aligned}
\mathcal{P}_e(\lambda) &= \det\big(S(\lambda^2 I_M + (\lambda k_v + k_p)E)S^{-1}\big) \\
&= \det\Big(\lambda^2 I_M + (\lambda k_v + k_p)SES^{-1}\Big) \\
&= \lambda^{2(M-(N-1))}\det\big(\lambda^2 I_{N-1} + (\lambda k_v + k_p)R\big) \\
&= \lambda^{2(M-(N-1))}\det\big(\lambda^2 I_{2(N-1)} - F_{e,r}\big) \\
&= \lambda^{2(M-(N-1))}\mathcal{P}_{e,r}(\lambda),
\end{aligned}$$

where $\mathcal{P}_{e,r}(\lambda)$ is the characteristic polynomial of $F_{e,r}$. Therefore, F_e and $F_{e,r}$ have the same nonzero eigenvalues with the same multiplicity. □

The proof of Theorem 4.1 is given in the following three sections of the chapter. Namely, in Section 4.4, we study the convergence properties of the closed-loop system, while, in Section 4.5, we show that the closed-loop system does not exhibit Zeno behavior [73]. Finally, in Section 4.6 we put the results of Sections 4.4 and 4.5 together to state a formal proof of Theorem 4.1.

4.4 Convergence proof

Our first step in the analysis of the closed-loop system is to rewrite the system dynamics in terms of the error vector $\xi(t)$. First, compare the control signals $u_{i,k}$ defined by (4.6) with

$$z_i(t) = \sum_{j\in\mathcal{N}_i} w_{ij}(k_p(p_j(t) - p_i(t)) + k_v(v_j(t) - v_i(t))). \tag{4.30}$$

We can write $z_i(t)$ in terms of the incidence matrix and the weight matrix of the network graph as

$$z_i(t) = ((W_i B^\top) \otimes I_n)(k_p p(t) + k_v v(t)),$$

where W_i denotes the i-th row of W. Letting $z(t) = [z_1(t)^\top, \ldots, z_N(t)^\top]^\top$, we can rewrite (4.30) as

$$z(t) = ((WB^\top) \otimes I_n)(k_p p(t) + k_v v(t)). \tag{4.31}$$

Moreover, substituting $W = [W_{\mathcal{T}}\ W_{\mathcal{C}}]$ and $B = [B_{\mathcal{T}}\ B_{\mathcal{C}}] = B_{\mathcal{T}}[I\ T]$ in (4.31), we have

$$z(t) = ((W_{\mathcal{T}} + W_{\mathcal{C}}T^\top)B_{\mathcal{T}}^\top \otimes I_n)(k_p p(t) + k_v v(t)). \tag{4.32}$$

Using the properties of the Kronecker product, and recalling that $x(t) = (B_{\mathcal{T}}^\top \otimes I_n)p(t)$ and $y(t) = (B_{\mathcal{T}}^\top \otimes I_n)v(t)$, we can rewrite (4.32) as

$$z(t) = ((W_{\mathcal{T}} + W_{\mathcal{C}}T^\top) \otimes I_n)(k_p x(t) + k_v y(t)). \tag{4.33}$$

Left-multiplying both sides of (4.33) by $B_{\mathcal{T}}^\top \otimes I_n$, using again the properties of the Kronecker product, and recalling the definition (2.7) of the reduced edge Laplacian, we have

$$(B_{\mathcal{T}}^\top \otimes I_n)z(t) = (R \otimes I_n)(k_p x(t) + k_v y(t)). \tag{4.34}$$

Let $\tilde{u}_i(t)$ be the mismatch between the control input of agent i and $z_i(t)$, namely,

$$\tilde{u}_i(t) = u_i(t) - z_i(t). \tag{4.35}$$

We denote $\tilde{u}(t) = [\tilde{u}_1(t)^\top, \dots, \tilde{u}_N(t)^\top]^\top$, so that we can rewrite (4.35) as

$$\tilde{u}(t) = u(t) - z(t). \tag{4.36}$$

Left-multiplying both sides of (4.2a) and (4.2b) by $B_{\mathcal{T}}^\top \otimes I_n$, we have

$$\dot{x}(t) = y(t), \tag{4.37a}$$
$$\dot{y}(t) = (B_{\mathcal{T}}^\top \otimes I_n)(u(t) + d(t)). \tag{4.37b}$$

Substituting (4.34) and (4.36) in (4.37a) and (4.37b), we have

$$\begin{aligned}\dot{x}(t) &= y(t),\\ \dot{y}(t) &= -(R \otimes I_n)(k_p x(t) + k_v y(t))\\ &\quad + (B_{\mathcal{T}}^\top \otimes I_n)(\tilde{u}(t) + d(t)),\end{aligned}$$

which, recalling that $\xi(t) = [x(t)^\top\ y(t)^\top]^\top$, can be rewritten as

$$\dot{\xi}(t) = (F_{e,r} \otimes I_n)\xi(t) + (G \otimes I_n)(\tilde{u}(t) + d(t)), \tag{4.38}$$

where $F_{e,r}$ is defined in (4.14) and $G = [0_{(N-1)\times N}^\top\ B_{\mathcal{T}}^\top]^\top$.

The following Lemma 4.1 shows that $\hat{\eta}(t)$ defined by (4.13) constitutes an upper bound for the state error vector $\xi(t)$.

Lemma 4.1. *Under Assumption 4.1, we have* $\|\xi(t)\| \le \hat{\eta}(t)$ *for all* $t \ge 0$, *where* $\hat{\eta}(t)$ *is defined by* (4.13).

Proof. Denote $D_{v,i}(t) = \int_{t_{i,l_i(t)}}^{t} d_i(\tau)\,\mathrm{d}\tau$ and $D_{p,i}(t) = \int_{t_{i,l_i(t)}}^{t}\int_{t_{i,l_i(t)}}^{\tau} d_i(\theta)\,\mathrm{d}\theta\,\mathrm{d}\tau$, let $D_p(t) = [D_{p,1}(t), \ldots, D_{p,N}(t)]$, and similarly for $D_v(t)$. Using (4.1), (4.7) and (4.8), we have

$$p(t) = \hat{p}(t) + D_p(t), \tag{4.39a}$$
$$v(t) = \hat{v}(t) + D_v(t), \tag{4.39b}$$

Left multiplying (4.39a) and (4.39b) by $(B_{\mathcal{T}} \otimes I_n)$, and using (4.10)–(4.12), we have

$$\xi(t) = \hat{\xi}(t) + ((B_{\mathcal{T}} \otimes I_n) \otimes I_2)D(t),$$

where we have denoted $D(t) = [D_p(t)^\top, D_v(t)^\top]^\top$. Taking norms of both sides, and using the triangular inequality, the properties of the Kronecker product, and Assumption 4.1, we have

$$\|\xi(t)\| \leq \|\hat{\xi}(t)\| + \|B_{\mathcal{T}}\|\|D(t)\|. \tag{4.40}$$

Under Assumption 4.1, we have $\|D(t)\| \leq \|\Delta(t)\|$, which substituted in (4.40) yields the desired result. □

Note that $\hat{\eta}(t)$ as defined in (4.13) can be computed by the cloud at any time instant. However, the cloud does not need to compute $\hat{\eta}(t)$ at all time instants, but only when an agent connects to download $\hat{\eta}(t_{i,k})$. As a consequence of Lemma 4.1, we have, in particular,

$$\|\xi(t_{i,k})\| \leq \hat{\eta}(t_{i,k}). \tag{4.41}$$

The following Lemma relates a bound on the control errors $\tilde{u}_i(t)$ to a bound on the state error vector $\xi(t)$ and on the control signals $u_i(t)$.

Lemma 4.2. *Consider the multi-agent system* (4.1)*, and let Assumption 4.1 hold. Suppose that*

$$\|\tilde{u}_i(t)\| \leq \varsigma(t) \tag{4.42}$$

for all $t \in [t_0, t_f)$ and all $i \in \mathcal{V}$, where $\tilde{u}_i(t)$ is defined by (4.35) *and $\varsigma(t)$ is the threshold function* (4.9)*. Let $\eta_0 \geq \|\xi(t_0)\|$. Then, for all $t \in [t_0, t_f)$, we have*

$$\|\xi(t)\| \leq \eta(t, \eta_0), \tag{4.43}$$

where $\eta(\cdot,\cdot)$ is defined by (4.16)*. Moreover, we have*

$$\|u_i(t)\| \leq \beta_i \eta(t, \eta_0) + \varsigma(t) \tag{4.44}$$

for all $t \in [t_0, t_f)$, and all $i \in \mathcal{V}$, where β_i is defined by (4.17)*.*

Proof. The Laplace solution of (4.38) reads

$$\xi(t) = e^{F_{e,r}(t-t_0)}\xi(t_0) + \int_{t_0}^{t} e^{F_{e,r}(t-\tau)}(G \otimes I_n)(\tilde{u}(\tau) + d(\tau))\,\mathrm{d}\tau\,. \tag{4.45}$$

Taking norms of both sides in (4.45), and using (4.42), Assumption 4.1, the properties of the Kronecker product, and the triangular inequality, and observing that $\|e^{F_{e,r}(t-t_0)}\| \leq e^{-\lambda(t-t_0)}$, and that $\|G\| = \|B_{\mathcal{T}}\|$, we have (4.43). Moreover, from (4.35), we have $u_i(t) = z_i(t) + \tilde{u}_i(t)$. Taking norms of both sides, and using the triangular inequality, we have $\|u_i(t)\| \leq \|z_i(t)\| + \|\tilde{u}_i(t)\|$. Selecting the rows corresponding to the i-th agent in (4.33), we have $z_i(t) = ((W_{\mathcal{T}} + W_{\mathcal{C}}T^\top)_i \otimes I_n)(k_p x(t) + k_v y(t))$, where $(W_{\mathcal{T}} + W_{\mathcal{C}}T^\top)_i$ denotes the i-th row of $(W_{\mathcal{T}} + W_{\mathcal{C}}T^\top)$. Taking norms of both sides, and substituting the result in the previous inequality, we have $\|u_i(t)\| \leq \beta_i\|\xi(t)\| + \|\tilde{u}_i(t)\|$. Using (4.42) and (4.43), we obtain (4.44). □

Since (4.41) holds, we can invoke Lemma 4.2 with $t_0 = t_{i,k}$ and $\eta_0 = \hat{\eta}(t_{i,k})$, which leads to the implication

$$\begin{aligned} &\|\tilde{u}_j(t)\| \leq \varsigma(t) \quad \forall t \in [t_{i,k}, t_f), j \in \mathcal{V} \\ &\quad \implies \|u_j(t)\| \leq \mu_j^{i,k}(t) \quad \forall t \in [t_{i,k}, t_f), j \in \mathcal{V}, \end{aligned} \tag{4.46}$$

where $\mu_j^{i,k}(t)$ is defined by (4.22).

The following Lemma 4.3 shows that, under the scheduling rule (4.18)–(4.24), we can guarantee that $\|\tilde{u}_i(t)\| \leq \|\varsigma(t)\|$ for all agents, thus satisfying the hypotheses of Lemma 4.2.

Lemma 4.3. *Consider the multi-agent system* (4.2) *under the control law* (4.5)–(4.8) *and the scheduling rule* (4.19)–(4.24)*. Then, under Assumption 4.1, we have* $\|\tilde{u}_i(t)\| \leq \varsigma(t)$ *for all* $t \geq 0$ *and* $i \in \mathcal{V}$*.*

Proof. Since (4.19) guarantees $\sigma_{i,l_i(t)}(t) \leq \varsigma(t)$ for all $t \geq 0$ and all $i \in \mathcal{V}$, we only need to show that $\|\tilde{u}_i(t)\| \leq \sigma_{i,l_i(t)}(t)$ for all $t \geq 0$ and all $i \in \mathcal{V}$. Without loss of generality, let $l_i(t) = k$, and consider $t \in [t_{i,k}, t_{i,k+1})$. Substituting (4.6) and (4.30) in (4.35), we have

$$\begin{aligned} \tilde{u}_i(t) = \sum_{j \in \mathcal{N}_i} w_{ij}(k_p((\hat{p}_j^{(i,k)} - p_j(t)) - (p_{i,k} - p_i(t))) \\ + k_v((\hat{v}_j^{(i,k)} - v_j(t)) - (v_{i,k} - v_i(t)))). \end{aligned} \tag{4.47}$$

First, consider the term $v_i(t)$ in (4.47). Integrating (4.1b) in $(t_{i,k}, t)$, we have, for $t \in (t_{i,k}, t_{i,k+1})$,

$$v_i(t) = v_{i,k} + (t - t_{i,k})u_{i,k} + \int_{t_{i,k}}^{t} d_i(t)\,\mathrm{d}\tau\,. \tag{4.48}$$

Now consider the term $v_j(t)$ in (4.47). Integrating (4.1b) for agent j in (t_{j,h_j}, t), and using (4.7), we have

$$v_j(t) = \hat{v}_j^{(i,k)} + \int_{t_{i,k}}^{t} u_j(\tau)\,\mathrm{d}\tau + \int_{t_{j,h_j}}^{t} d_j(\tau)\,\mathrm{d}\tau\,. \tag{4.49}$$

Now we need to distinguish two cases. If $t \leq t_{j,h_j+1}$, then (4.49) can be rewritten as

$$v_j(t) = \hat{v}_j^{(i,k)} + (t - t_{i,k})u_{j,h_j} + \int_{t_{j,h_j}}^{t} d_j(\tau)\,\mathrm{d}\tau\,. \tag{4.50}$$

Conversely, if $t > t_{j,h_j+1}$, (4.49) can be rewritten as

$$\begin{aligned} v_j(t) = \hat{v}_j^{(i,k)} &+ (t_{j,h_j+1} - t_{i,k})u_{j,h_j} \\ &+ \int_{t_{j,h_j+1}}^{t} u_j(\tau)\,\mathrm{d}\tau + \int_{t_{j,h_j}}^{t} d_j(\tau)\,\mathrm{d}\tau\,. \end{aligned} \tag{4.51}$$

Using (4.23) and (4.24), we can write (4.50) and (4.51) compactly as

$$\begin{aligned} v_j(t) = \hat{v}_j^{(i,k)} &+ (t'_{j,h_j} - t_{i,k})u_{j,h_j} \\ &+ \int_{t_{j,h_j+1}}^{t''_{j,h_j}} u_j(\tau)\,\mathrm{d}\tau + \int_{t_{j,h_j}}^{t} d_j(\tau)\,\mathrm{d}\tau\,. \end{aligned} \tag{4.52}$$

Now consider the term $p_i(t)$ in (4.47). Integrating (4.1a) in $(t_{i,k}, t)$, and using (4.48), we have, for $t \in (t_{i,k}, t_{i,k+1})$,

$$\begin{aligned} p_i(t) = p_{i,k} &+ v_{i,k}(t - t_{i,k}) + (1/2)(t - t_{i,k})^2 u_{i,k} \\ &+ \int_{t_{i,k}}^{t}\int_{t_{i,k}}^{\tau} d_i(\theta)\,\mathrm{d}\theta\,\mathrm{d}\tau\,. \end{aligned} \tag{4.53}$$

Finally, consider the term $p_j(t)$ in (4.47). Integrating (4.1a) for agent j in $[t_{j,h_j}, t)$, and using (4.8), we have

$$\begin{aligned} p_j(t) = \hat{p}_j^{(i,k)} &+ (t - t_{i,k})v_{j,h_j} \\ &+ \int_{t_{i,k}}^{t}\int_{t_{j,h_j}}^{\tau} u_j(\sigma)\,\mathrm{d}\sigma\,\mathrm{d}\tau + \int_{t_{j,h_j}}^{t}\int_{t_{j,h_j}}^{\tau} d_j(\sigma)\,\mathrm{d}\sigma\,\mathrm{d}\tau\,. \end{aligned} \tag{4.54}$$

Similarly as we did for (4.49), we need to distinguish two cases. If $t \leq t_{j,h_j+1}$, then (4.54) can be rewritten as

$$\begin{aligned} p_j(t) = \hat{p}_j^{(i,k)} &+ (t - t_{i,k})v_{j,h_j} \\ &+ (1/2)(t - t_{i,k})(t + t_{i,k} - 2t_{j,h_j})u_{j,h_j} \\ &+ \int_{t_{j,h_j}}^{t} \int_{t_{j,h_j}}^{\tau} d_j(\sigma)\,\mathrm{d}\sigma\,\mathrm{d}\tau\,. \end{aligned} \tag{4.55}$$

Conversely, if $t > t_{j,h_j+1}$, (4.54) can be rewritten as

$$\begin{aligned} p_j(t) = \hat{p}_j^{(i,k)} &+ (t - t_{i,k})v_{j,h_j} \\ &+ (1/2)(t_{j,h_j+1} - t_{i,k})(t_{j,h_j+1} + t_{i,k} - 2t_{j,h_j})u_{j,h_j} \\ &+ (t - t_{j,h_j+1})(t_{j,h_j+1} - t_{j,h_j})u_{j,h_j} \\ &+ \int_{t_{j,h_j+1}}^{t} \int_{t_{j,h_j+1}}^{\tau} u_j(\sigma)\,\mathrm{d}\sigma\,\mathrm{d}\tau + \int_{t_{j,h_j}}^{t} \int_{t_{j,h_j}}^{\tau} d_j(\sigma)\,\mathrm{d}\sigma\,\mathrm{d}\tau\,. \end{aligned} \tag{4.56}$$

Using (4.23) and (4.24), we can write (4.55) and (4.56) compactly as

$$\begin{aligned} p_j(t) = \hat{p}_j^{(i,k)} &+ (t - t_{i,k})v_{j,h_j} \\ &+ (1/2)(t'_{j,h_j} + t_{i,k} - 2t_{j,h_j})(t'_{j,h_j} - t_{i,k})u_{j,h_j} \\ &+ (t''_{j,h_j} - t_{j,h_j+1})(t_{j,h_j+1} - t_{j,h_j})u_{j,h} \\ &+ \int_{t_{j,h_j+1}}^{t''_{j,h_j}} \int_{t_{j,h_j+1}}^{\tau} u_j(\theta)\,\mathrm{d}\theta\,\mathrm{d}\tau + \int_{t_{j,h_j}}^{t} \int_{t_{j,h_j}}^{\tau} d_j(\theta)\,\mathrm{d}\theta\,\mathrm{d}\tau\,. \end{aligned} \tag{4.57}$$

Substituting (4.48), (4.52), (4.53) and (4.57) in (4.47), taking norms of both

sides, and using the triangular inequality and Assumption 4.1, we have

$$\begin{aligned}
\|\tilde{u}_i(t)\| \leq & \Bigg\| \Bigg(\sum_{j \in \mathcal{N}} w_{ij} \Bigg) (k_v (t - t_{i,k}) u_{i,k} \\
& + k_p ((t - t_{i,k}) v_{i,k} + (1/2)(t - t_{i,k})^2 u_{i,k})) \\
& + \sum_{j \in \mathcal{N}_i} w_{ij} (k_v (t'_{j,h_j} - t_{i,k}) u_{j,h_j} \\
& + k_p ((t - t_{i,k}) v_{j,h_j} \\
& + (1/2)(t'_{j,h_j} + t_{i,k} - 2t_{j,h_j})(t'_{j,h_j} - t_{i,k}) u_{j,h_j} \\
& + (t''_{j,h_j} - t_{j,h_j+1})(t_{j,h_j+1} - t_{j,h_j}) u_{j,h})) \Bigg\| \\
& + \sum_{j \in \mathcal{N}_i} w_{ij} \Bigg(\int_{t_{j,h_j+1}}^{t''_{j,h_j}} \|u_j(\tau)\| \, \mathrm{d}\tau \\
& \quad + \int_{t_{j,h_j+1}}^{t''_{j,h_j}} \int_{t_{j,h_j+1}}^{\tau} \|u_j(\theta)\| \, \mathrm{d}\theta \, \mathrm{d}\tau \Bigg) \\
& + \Bigg(\sum_{j \in \mathcal{N}_i} w_{ij} \Bigg) \Omega_{i,k}(t) + \sum_{j \in \mathcal{N}_i} w_{ij} \Omega_{j,h_j}(t). \qquad (4.58)
\end{aligned}$$

Comparing (4.58) with (4.21), we have that, if $\|\tilde{u}_i(t)\| > \sigma_{i,k}(t)$ for some $t \in (t_{i,k}, t_{i,k+1})$, then it must be $\|u_j(\tau)\| > \mu_j^{i,k}(\tau)$ for some $j \in \mathcal{V}$ and some $\tau \in (t_{j,h_j+1}, t)$. But, by (4.46), the previous inequality implies $\|\tilde{u}_j(\tau')\| > \varsigma(\tau')$ for some $\tau' \in (t_{j,h_j+1}, t)$, which, by (4.19), implies in turn $\|\tilde{u}_j(\tau')\| > \sigma_{j,l_j(\tau')}(\tau')$. Therefore, the condition $\|\tilde{u}_i(t)\| \leq \sigma_{i,l_i(t)}(t)$ cannot be violated by any of the agents if it has not been previously violated by another agent. Since we have $\tilde{u}(0) = 0_{Nn}$, this condition holds for all the agents at time zero, and, therefore, cannot be violated by any of the agents. Hence, we have $\|\tilde{u}_i(t)\| \leq \sigma_{i,l_i(t)}(t)$ for all $t \geq 0$ and all $i \in \mathcal{V}$, which, by (4.19), implies $\|\tilde{u}_i(t)\| \leq \varsigma(t)$ for all $t \geq 0$ and all $i \in \mathcal{V}$. □

4.5 Well-posedness proof

The second step in our analysis is to prove that the closed-loop system is well posed, in the sense that the sequence of the updates $t_{i,k}$ for $k \in \mathbb{N}_0$ does not present Zeno behavior for any of the agents. We are going to distinguish two cases, namely $\varsigma_\infty > 0$ and $\varsigma_\infty = 0$, where ς_∞ is the asymptotic value of the threshold function (4.9).

Lemma 4.4. *Consider the multi-agent system* (4.2)*, with control law* (4.5)–(4.8) *and cloud accesses scheduled by* (4.19)–(4.24)*. Let k_p and k_v be such that $F_{e,r}$ is*

Hurwitz. Suppose $\varsigma_\infty > 0$. Then, under Assumptions 4.1 and 4.2, the closed-loop system does not exhibit Zeno behavior.

Proof. First, note that $\Omega_{i,k}(t)$, for $t \in [t_{i,k}, t_{i,k+1})$, can be computed explicitly as

$$\begin{aligned}\Omega_{i,k}(t) = k_p\Bigg(\frac{\delta_0-\delta_\infty}{\lambda_\delta}e^{-\lambda_\delta t_{i,k}}\Bigg((t-t_{i,k}) \\ - \frac{1-e^{-\lambda_\delta(t-t_{i,k})}}{\lambda_\delta}\Bigg) + \frac{1}{2}\delta_\infty(t-t_{i,k})^2\Bigg) \\ + k_v\Bigg(\frac{\delta_0-\delta_\infty}{\lambda_\delta}e^{-\lambda_\delta t_{i,k}}(1-e^{-\lambda_\delta(t-t_{i,k})}) + \delta_\infty(t-t_{i,k})\Bigg). \end{aligned} \tag{4.59}$$

Observing that $e^{-\lambda_\delta t_{i,k}} \leq 1$, we can bound (4.59) as

$$\begin{aligned}\Omega_{i,k}(t) \leq k_p\Bigg(\frac{\delta_0-\delta_\infty}{\lambda_\delta}(t-t_{i,k}) + \frac{1}{2}\delta_\infty(t-t_{i,k})^2\Bigg) \\ + k_v\Bigg(\frac{\delta_0-\delta_\infty}{\lambda_\delta}(1-e^{-\lambda_\delta(t-t_{i,k})}) + \delta_\infty(t-t_{i,k})\Bigg),\end{aligned} \tag{4.60}$$

while $\varsigma(t) \geq \varsigma_\infty$. Therefore, for $\Omega_{i,k}(t)$ to be larger than $\alpha\varsigma(t)/\nu_i$, the right-hand side of (4.60) must be greater or equal than $\alpha\varsigma_\infty/\nu_i$, which, in turn, requires a strictly positive value of $t - t_{i,k}$. Hence, the triggering condition $\Omega_{i,k}(t) \geq \alpha\varsigma(t)/\nu_i$ cannot be responsible for Zeno behavior. Now we produce a similar argument for the triggering condition $\sigma_{i,k}(t) \geq \varsigma(t)$. First note that, thanks to (4.18) and (4.19), the last two addends of $\sigma_{i,k}(t)$ in (4.21) can be bounded by

$$\Bigg(\sum_{j\in\mathcal{N}} w_{ij}\Bigg)\Omega_{i,k}(t) + \sum_{j\in\mathcal{N}} w_{ij}\Omega_{j,h_j}(t) \leq \alpha\varsigma(t). \tag{4.61}$$

Moreover, for $t \in [t_{i,k}, t_{i,k+1})$, letting $h_j = l_j(t_{i,k})$, we have, from Lemma 4.2,

$$\|u_{i,k}\| \leq \beta_i\eta(t_{i,k}, \|\xi(0)\|) + \varsigma(t_{i,k}), \tag{4.62}$$

$$\|u_{j,h_j}\| \leq \beta_j\eta(t_{i,k}, \|\xi(0)\|) + \varsigma(t_{i,k}). \tag{4.63}$$

Since $\eta(t, \eta_0)$ is an upper-bounded function of t for any η_0, we can denote as $\bar{\eta}$ the upper bound of $\eta(\cdot, \|\xi(0)\|)$, which, by observing also that $\varsigma(t_{i,k}) \leq \varsigma_0$, allows us to further bound (4.62) and (4.63) as

$$\|u_{i,k}\| \leq \beta_i\bar{\eta} + \varsigma_0, \tag{4.64}$$

$$\|u_{j,h_j}\| \leq \beta_j\bar{\eta} + \varsigma_0. \tag{4.65}$$

Reasoning similarly for $\mu_j(t)$ for $t \in [t_{j,h_j+1})$, we have

$$\mu_j(t) \leq \beta_j \bar{\eta} + \varsigma_0. \tag{4.66}$$

Also, note that

$$\|v_{j,h_j} - v_{i,k}\| \leq \|\xi(t_{i,k})\| \leq \bar{\eta}. \tag{4.67}$$

Substituting (4.61) and (4.64)–(4.67) into (4.21), and using the triangular inequality, we have

$$\begin{aligned}\sigma_{i,k}(t) \leq & \left(\sum_{j\in\mathcal{N}} w_{ij}(\beta_j\bar{\eta} + \varsigma_0 + \beta_i\bar{\eta} + \varsigma_0)\right) \\ & (k_v(t - t_{i,k}) + (1/2)k_p(t - t_{i,k})^2) \\ & + \left(\sum_{j\in\mathcal{N}} w_{ij}\right)\bar{\eta}k_p(t - t_{i,k}) + \alpha\varsigma(t).\end{aligned} \tag{4.68}$$

Noticing also that $\varsigma(t) \geq \varsigma_\infty$ for all $t \geq 0$, the triggering condition $\sigma_{i,k}(t) \geq \varsigma(t)$ implies that the first two addends of (4.68) are larger than $(1-\alpha)\varsigma_\infty$, which requires a strictly positive value of $t - t_{i,k}$. Hence, the triggering condition $\sigma_{i,k}(t) \geq \varsigma(t)$ cannot be responsible for Zeno behavior either. □

When the disturbances eventually vanish, the particular case of asymptotic convergence may be considered with the specific choice of $\varsigma_\infty = 0$ and a convergence rate λ_ς slower than λ, which is defined by (4.15) and can be interpreted as the natural convergence rate of the network. This is formally stated in the following Lemma.

Lemma 4.5. *Consider the multi-agent system* (4.1)*, with control law* (4.5)–(4.8) *and cloud accesses scheduled by* (4.19)–(4.24)*. Let Assumptions 4.1 and 4.2 hold, with $\delta_\infty = 0$ in Assumption 4.1. Choose k_p and k_v such that $F_{e,r}$ is Hurwitz, and choose $\varsigma_\infty = 0$ and $\lambda_\varsigma < \min\{\lambda, \lambda_\delta\}$. Then, the closed-loop system does not exhibit Zeno behavior.*

Proof. We write the proof for the case $\lambda > \lambda_\delta$. The structure of the proof is the same for $\lambda < \lambda_\delta$ and $\lambda = \lambda_\delta$, with only the expression (4.75) of the coefficient $\bar{\bar{\eta}}$ taking a slightly different form. Under Assumption 4.1 with $\delta_\infty = 0$ and $\lambda_\varsigma < \lambda_\delta$, we can write

$$\delta(t) \leq \delta_0 e^{-\lambda_\varsigma t}. \tag{4.69}$$

Substituting (4.69) in (4.20), and solving the integrals explicitly, we have

$$\begin{aligned}\Omega_{i,k}(t) \leq & \left(k_p\frac{\delta_0}{\lambda_\varsigma}\left((t - t_{i,k}) - \frac{1 - e^{-\lambda_\varsigma(t-t_{i,k})}}{\lambda_\varsigma}\right)\right. \\ & \left. + k_v\frac{\delta_0}{\lambda_\varsigma}(1 - e^{-\lambda_\varsigma(t-t_{i,k})})\right)e^{-\lambda_\varsigma t_{i,k}}.\end{aligned} \tag{4.70}$$

Since by hypothesis $\varsigma_\infty = 0$, we have

$$\varsigma(t) = \varsigma_0 e^{-\lambda_\varsigma t} = \varsigma_0 e^{-\lambda_\varsigma t_{i,k}} e^{-\lambda_\varsigma (t - t_{i,k})}. \tag{4.71}$$

Comparing (4.70) and (4.71), after dividing both sides by $e^{-\lambda_\varsigma t_{i,k}}$, we have that the triggering condition $\Omega_{i,k}(t) \geq \alpha/\nu_i \varsigma(t)$ requires

$$\begin{aligned} k_p \frac{\delta_0}{\lambda_\varsigma} \left((t - t_{i,k}) - \frac{1 - e^{-\lambda_\varsigma (t - t_{i,k})}}{\lambda_\varsigma} \right) \\ + k_v \frac{\delta_0}{\lambda_\varsigma} (1 - e^{-\lambda_\varsigma (t - t_{i,k})}) \geq \frac{\alpha}{\nu_i} \varsigma_0 e^{-\lambda_\varsigma (t - t_{i,k})}, \end{aligned}$$

which, in turn, requires a strictly positive value of $t - t_{i,k}$. Hence, the triggering condition $\Omega_{i,k}(t) \geq \alpha/\nu_i \varsigma(t)$ cannot be responsible for Zeno behavior. Now we produce a similar argument for the triggering condition $\sigma_{i,k}(t) \geq \varsigma(t)$. Under Assumption 4.1 with $\delta_\infty = 0$, Lemma 4.2 holds, and (4.16) can be written as

$$\begin{aligned} \eta(t, \eta_0) = e^{-\lambda (t - t_0)} \eta_0 + \sqrt{N} \|B_{\mathcal{T}}\| e^{-\lambda t} \\ \left(\frac{\varsigma_0 (e^{(\lambda - \lambda_\varsigma) t} - e^{(\lambda - \lambda_\varsigma) t_0})}{\lambda - \lambda_\varsigma} + \frac{\delta_0 (e^{(\lambda - \lambda_\delta) t} - e^{(\lambda - \lambda_\delta) t_0})}{\lambda - \lambda_\delta} \right) \end{aligned} \tag{4.72}$$

for all $t \geq t_0$. Choosing $t_0 = 0$ and $\eta_0 = \|\xi(0)\|$, from (4.72) we have

$$\eta(t, \|\xi(0)\|) \leq e^{-\lambda t} \|\xi(0)\| + \sqrt{N} \|B_{\mathcal{T}}\| \left(\frac{\varsigma_0 e^{-\lambda_\varsigma t}}{\lambda - \lambda_\varsigma} + \frac{\delta_0 e^{-\lambda_\delta t}}{\lambda - \lambda_\delta} \right) \tag{4.73}$$

for all $t \geq 0$. Since, by hypothesis, $\lambda_\varsigma < \lambda$, (4.73) can be further bounded by

$$\eta(t, \|\xi(0)\|) \leq \bar{\bar{\eta}} e^{-\lambda_\varsigma t}, \tag{4.74}$$

where we have denoted

$$\bar{\bar{\eta}} = \left(\|\xi(0)\| + \sqrt{N} \|B_{\mathcal{T}}\| \left(\frac{\varsigma_0}{\lambda - \lambda_\varsigma} + \frac{\delta_0}{\lambda - \lambda_\delta} \right) \right). \tag{4.75}$$

Substituting (4.74) in (4.62) and (4.63), and observing that $\varsigma(t_{i,k}) = \varsigma_0 e^{-\lambda_\varsigma t}$, we have

$$\|u_{i,k}\| \leq (\beta_i \bar{\bar{\eta}} + \varsigma_0) e^{-\lambda_\varsigma t_{i,k}}, \tag{4.76}$$

$$\|u_{j,h_j}\| \leq (\beta_j \bar{\bar{\eta}} + \varsigma_0) e^{-\lambda_\varsigma t_{i,k}}. \tag{4.77}$$

Reasoning similarly for $\mu_j(t)$ for $t > t_{j,h_j+1}$, we have

$$\mu_j(t) \leq (\beta_j \bar{\bar{\eta}} + \varsigma_0) e^{-\lambda_\varsigma t_{i,k}}. \tag{4.78}$$

Also, note that

$$\|v_{j,h_j} - v_{i,k}\| \leq \|\xi(t_{i,k})\| \leq \bar{\bar{\eta}} e^{-\lambda_\varsigma t_{i,k}}. \tag{4.79}$$

Finally, note that (4.61) remains valid under the hypotheses of this Lemma. Therefore, substituting (4.61) and (4.76)–(4.79) into (4.21), observing that $\varsigma(t) = \varsigma_0 e^{-\lambda_\varsigma t_{i,k}}$ and using the triangular inequality, we have

$$\begin{aligned}\sigma_{i,k}(t) \leq \Bigg(\Bigg(\sum_{j\in\mathcal{N}} w_{ij}(\beta_j \bar{\bar{\eta}} + \varsigma_0 + \beta_i \bar{\bar{\eta}} + \varsigma_0) \Bigg) \\ (k_v(t - t_{i,k}) + (1/2)k_p(t - t_{i,k})^2) \\ + \Bigg(\sum_{j\in\mathcal{N}} w_{ij} \Bigg) \bar{\bar{\eta}} k_p(t - t_{i,k}) + \alpha\varsigma_0 \Bigg) e^{-\lambda_\varsigma t_{i,k}}.\end{aligned} \tag{4.80}$$

From (4.71) and (4.80), after dividing both sides by $e^{-\lambda_\varsigma t_{i,k}}$, we can see that the triggering condition $\sigma_{i,k}(t) \geq \varsigma(t)$, implies

$$\begin{aligned}\Bigg(\Bigg(\sum_{j\in\mathcal{N}} w_{ij}(\beta_j \bar{\bar{\eta}} + \varsigma_0 + \beta_i \bar{\bar{\eta}} + \varsigma_0) \Bigg) \\ (k_v(t - t_{i,k}) + (1/2)k_p(t - t_{i,k})^2) \\ + \Bigg(\sum_{j\in\mathcal{N}} w_{ij} \Bigg) \bar{\bar{\eta}} k_p(t - t_{i,k}) \Bigg) \geq (1 - \alpha)\varsigma_0 e^{-\lambda_\varsigma (t - t_{i,k})},\end{aligned}$$

which, in turn, requires a strictly positive value of $t - t_{i,k}$. Hence the triggering condition $\sigma_{i,k}(t) \geq \varsigma(t)$ cannot be responsible for Zeno behavior either. □

4.6 Proof of Theorem 4.1

From Lemma 4.3, we know that, under the control law (4.5)–(4.8) and the scheduling rule (4.19)–(4.24) the hypotheses of Lemma 4.2 are satisfied.

If $\delta_\infty > 0$, we know from Lemma 4.4 that the closed-loop system does not exhibit Zeno behavior. Therefore, we can take $t \to \infty$ in (4.16) in Lemma 4.2, obtaining $\limsup_{t\to\infty} \|\xi(t)\| \leq \epsilon$, with ϵ given by (4.25).

If $\delta_\infty = 0$, $\varsigma_\infty = 0$ and $\lambda_\varsigma < \min\{\lambda, \lambda_\delta\}$, we know from Lemma 4.5 that the closed-loop system does not exhibit Zeno behavior. Therefore, we can take again $t \to \infty$ in (4.16), obtaining $\limsup_{t\to\infty} \|\xi(t)\| \leq \epsilon$. But since $\delta_\infty = \varsigma_\infty = 0$, (4.25) evaluates to zero, and therefore $\lim_{t\to\infty} \xi(t) = 0$.

4.7 Numerical simulations

In this section, two numerical simulations of the proposed control algorithm are presented, one for a scenario where practical convergence is reached, and

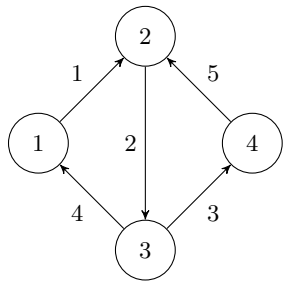


Figure 4.2. A graph with 4 nodes and 5 edges. The nodes and the edges are labelled with their indexes.

one for a scenario where asymptotic convergence is reached. For both simulations, we consider a multi-agent system made up of $N = 4$ agents with state in $\mathbb{R}^2$, which exchange information through a cloud repository according to the graph $\mathcal{G}$ illustrated in Figure 4.2, where all the edges are assigned unitary weights.

The assigned graph contains a spanning tree $\mathcal{T}$ made up of the first three edges. The corresponding reduced edge Laplacian is

$$R = \begin{bmatrix} 2 & 0 & -1 \\ -1 & 2 & 1 \\ 0 & -1 & 1 \end{bmatrix}.$$

The control gains are chosen as $k_p = 0.5$ and $k_d = 1.0$, which leads to $\lambda = -\max\{\mathrm{Re}(s) : \ s \in \mathrm{eig}(F_{e,r})\} = 0.5$ and $\|B_{\mathcal{T}}\| \simeq 2.45$. The disturbances are chosen as

$$d_i(t) = \delta(t) \begin{bmatrix} \cos(2\pi(i/N)t + 2\pi((N-i)/N)) \\ \sin(2\pi(i/N)t + 2\pi((N-i)/N)) \end{bmatrix},$$

where $\delta(t)$ is defined by (4.3) with $\delta_0 = 0.2$, $\lambda_\delta = 0.45$, $\delta_\infty = 0.02$ in the first simulation, and $\delta_\infty = 0$ in the second simulation. It is easy to see that, with these parameters, Assumption 4.1 is satisfied. The threshold function is chosen as (4.9), with $\varsigma_0 = 5.0$, $\lambda_\varsigma = 0.4$, $\varsigma_\infty = 0.5$ for the first simulation, and $\varsigma_\infty = 0$ for the second simulation. Note that, with these choices, the first simulation scenario satisfies the hypotheses of Theorem 4.1 for practical coordination, and the second simulation scenario satisfies the hypotheses of Theorem 4.1 for asymptotic convergence. For the coefficient α that appears in (4.19), we choose $\alpha = 0.05$.

The results of the first simulation are illustrated in Figure 4.3. From Figure 4.3 it looks clear that the multi-agent system only achieves practical convergence, but the norm of the disagreement vector is significantly reduced.

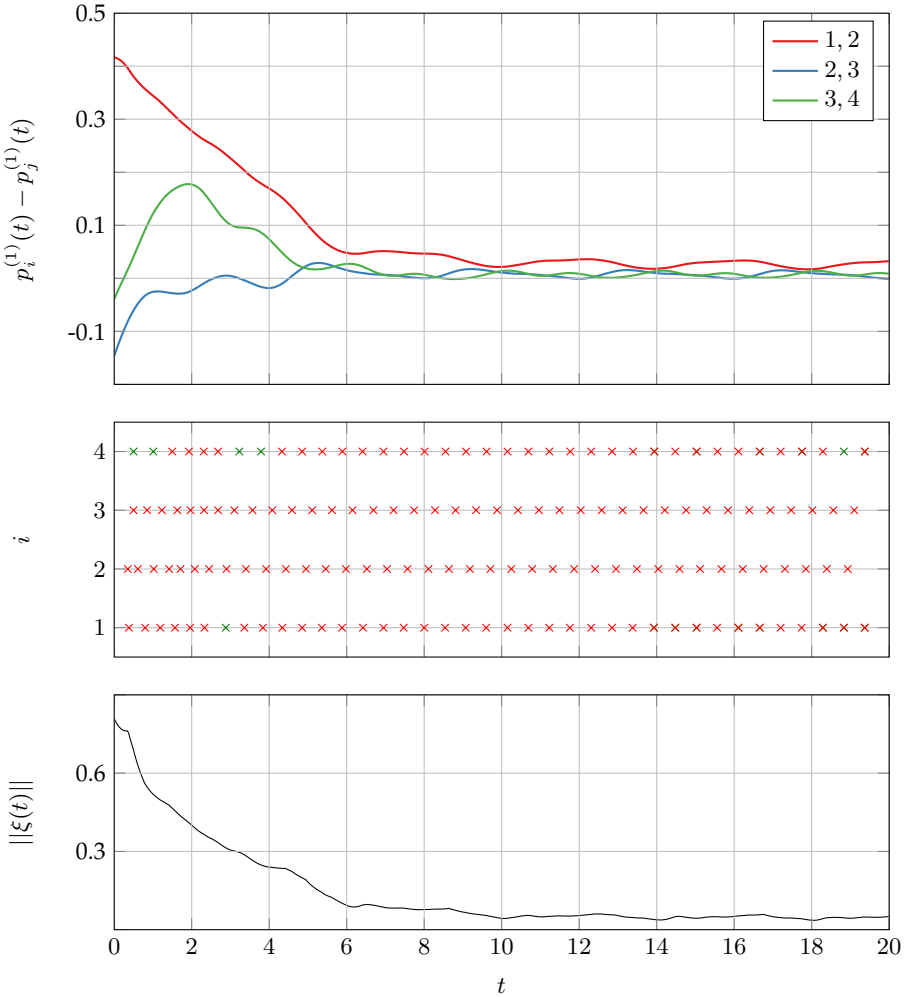


Figure 4.3. Simulation with persistent disturbances. Top: position mismatches across the edges (j, i) in the spanning tree over time. Middle: time instants when each agent accesses the cloud; a green cross denotes an access triggered by $\Omega_{i,k}(t) \geq \alpha/\nu_i \varsigma(t)$; a red cross denotes an access triggered by $\sigma_{i,k}(t) \geq \varsigma(t)$. Bottom: norm of the global disagreement vector $\xi(t)$ over time.

From Figure 4.3, we can also see that the cloud accesses do not accumulate; on the contrary, they seem to become less frequent over time, which corroborates the result that the closed-loop system does not exhibit Zeno behavior. The results of the second simulation are illustrated in Figure 4.4. From Figure 4.4 it looks clear that $\xi(t) \to 0$, which means that asymptotic convergence

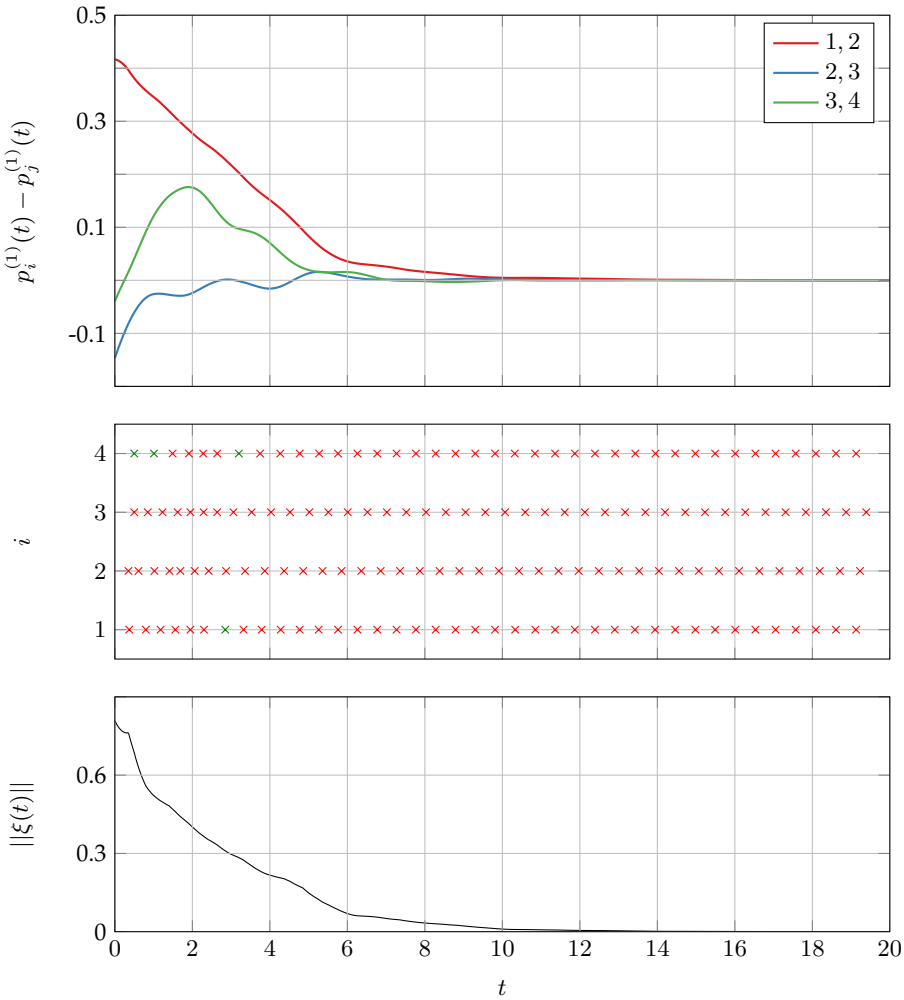


Figure 4.4. Simulation with asymptotically vanishing disturbances. Same subplots as in Figure 4.3.

is reached. From Figure 4.4, we can also see that the cloud accesses do not accumulate even if the threshold function is converging to zero, which again corroborates the result that the closed-loop system does not exhibit Zeno behavior.

4.8 Summary

This chapter has addressed a self-triggered control problem for multi-agent coordination of a team of agents with second-order dynamics. Coordination has been achieved by having the agents asynchronously deposit and retrieve data on a cloud repository, rather than by inter-agent communication. Two control objectives have been considered, namely practical and asymptotic convergence. It has been shown that the proposed control strategy achieves practical convergence in the presence of unknown bounded persistent disturbances, and asymptotic convergence in the presence of unknown disturbances if they slowly vanish. Well-posedness of the closed-loop system has been proved by showing that there is a lower bound for the time interval between two consecutive accesses to the cloud. The proposed scheme can be adopted in all cases when direct communication among agents is interdicted, as illustrated in our motivating example of controlling a fleet of AUVs.

Future work will address possible imperfections in the communication with the repository, such as time delays and packet losses, as well as more complex control objectives, such as coverage and inspection.

Chapter 5

Coverage control of anisotropic sensor networks

In this chapter, we study a coverage problem for a multi-agent network of mobile sensors.

Coverage problems have often been addressed under the assumption that the mobile sensors have a spherical sensing pattern, in the sense that the perception that a sensor has of a point in the surrounding environment only depends on the distance between the sensor and the point. With this assumption, locally optimal solutions are identified as the Voronoi tessellations of the environment under inspection, and one such solution is pursued via a distributed implementation of the Lloyd algorithm—see for example [40]. Conversely, in this thesis, we consider a coverage problem where the idea of discretizing the environment under observation [61–63] is conjugated with the use of sensors with generalized, anisotropic sensing patterns [57, 58], as well as with the use of Voronoi tessellations, which are opportunely redefined according to our generalized setup.

Voronoi tessellations for anisotropic sensing have been considered, for example, in [47, 49], but under the assumption of only elliptical footprints and continuous environments. Conversely, in this thesis, we apply generalized sensing patterns on discretized environments. Using the idea of Voronoi tessellations for anisotropic sensing in discretized environments allows us to define a novel distributed algorithm for anisotropic coverage where communication is limited, pairwise, intermittent, and asynchronous; and where complex algorithms for environment partitioning are not necessary. Thus, the proposed algorithm is suitable for implementation on networks of unmanned vehicles with limited computation capabilities, where motion planning must be executed in concurrence with lower-level control.

The rest of this chapter is organized as follows. In Section 5.1, we intro-

duce some notations and properites relative to unit vectors and used extensively throughout the chapter. (For notations, nomenclature and concepts relative to optimization that are used in this chapter, the reader is referred to [81].) In Section 5.2, we define the abstractions of a *landmark* and a *mobile sensor*, which set the basis for a mathematical formulation of our coverage problem. In Section 5.3 we give a generalized definition of *Voronoi tessellations*, which plays a central role in the convergence analysis of the algorithms proposed in the chapter. In Section 5.4, we formulate the coverage problem as an optimization problem. In Section 5.5, we identify the generalized Voronoi tessellations as a superset of the optimal solutions of the problem. In Section 5.6, we present a distributed algorithm that achieves a local optimum in the form of a Voronoi tessellation, and we demonstrate its convergence properties. In Section 5.7, we discuss the implementation of the proposed algorithm as a distributed program, and in Sections 5.8 and 5.9, we present and discuss respectively a simulation and an experimental evaluation of the proposed algorithm. In Section 5.10, we propose a control law for the motion of the sensors, and we discuss its convergence properties. In Section 5.11, we discuss a simulation of the proposed control law. Section 5.12 concludes the chapter with a summary.

5.1 Notations and properties related to unit vectors

Throughout this chapter, we use unit vectors to denote directions. We denote the set of the unit vectors in $\mathbb{R}^3$ as $\mathcal{S}^2$. Namely, we let

$$\mathcal{S}^2 = \{\hat{n} \in \mathbb{R}^3 : \|\hat{n}\| = 1\}.$$

Sometimes we call $\mathcal{S}^2$ the *unit sphere*. A vector in the unit sphere can always be expressed as a (not necessarily unique) tuple (θ, ϕ), with $\theta \in [0, 2\pi)$ and $\phi \in [0, \pi)$. Namely, for each $\hat{n} \in \mathcal{S}^2$, we have

$$\hat{n} = \begin{bmatrix} \cos\theta \sin\phi \\ \sin\theta \sin\phi \\ \cos\phi \end{bmatrix},$$

for some $\theta \in [0, 2\pi)]$ and some $\phi \in [0, \pi)$. The angles θ and ϕ are called the *spherical coordinates* of $\hat{n}$, and we write $\hat{n} = \mathrm{sph}(\theta, \phi)$. If a unit vector is time-dependent, its time derivative is always orthogonal to the vector, because the norm of the vector must be always equal to one. Namely, we have

$$\dot{\hat{n}} = \mathrm{skew}(\hat{n})\omega \tag{5.1}$$

for some $\omega \in \mathbb{R}^3$. We call the vector ω the *angular velocity* of the unit vector. Here $\mathrm{skew}(\cdot)$ denotes the skew operator,

$$\mathrm{skew}(v) = \begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{bmatrix},$$

where $v = [v_x, v_y, v_z]^\top$. The skew operator has the property that, for each $v, u \in \mathbb{R}^3$, $\mathrm{skew}(v)u$ is equal to $v \times u$, where $\times$ denotes the cross product. Therefore, from (5.1), we have that $\mathrm{skew}(\hat{n})\omega$ is orthogonal to $\hat{n}$ and to ω for any $\hat{n} \in \mathcal{S}^2$ and any $\omega \in \mathbb{R}^3$.

5.2 Landmarks and sensors

In this section, we introduce the definitions of *landmark* and *mobile sensor*.

A *landmark* is an abstraction of a point or a small area of interest, that must be kept under observation, and that may be part of a larger surface. A landmark is formally defined as the tuple $\ell = (q, \hat{m})$, where $q \in \mathbb{R}^3$ is the position of the landmark, and $\hat{m} \in \mathcal{S}^2$ is the orientation of the landmark. More specifically, $\hat{m}$ is a direction that characterizes the orientation of the landmark.

A *mobile sensor* is a tuple $s = (p, \hat{n}, f)$, where $p \in \mathbb{R}^3$ is the position of the sensor, $\hat{n} \in \mathcal{S}^2$ is the orientation, and $f : (\mathbb{R}^3 \times \mathcal{S}^2)^2 \to \mathbb{R}_{\geq 0}$ is a function called the *footprint* of the sensor. The orientation of a sensor is the unit vector corresponding to the direction where the sensor is pointing. The footprint of a mobile sensor is a function that describes the sensor's perception of the surrounding environment. Namely, if $s = (p, \hat{n}, f)$ is a mobile sensor, and $\ell = (q, \hat{m})$ is a landmark, then $f(p, \hat{n}, q, \hat{m})$ is a measure of the quality of the perception that the sensor has of the landmark. Lower values of the footprint correspond to a better perception, and the value zero corresponds to the best possible perception. In some of the control algorithms proposed in this chapter, we assume that we can control the velocity $v \in \mathbb{R}^3$ and the angular velocity $\omega \in \mathbb{R}^3$ of a mobile sensor. The velocity and angular velocity are related to the position and orientation of the sensor by

$$\dot{p} = v, \tag{5.2}$$

$$\dot{\hat{n}} = \mathrm{skew}(\hat{n})\omega. \tag{5.3}$$

The perception that a mobile sensor has of a landmark is given by evaluating the sensor's footprint in the position and orientation of the sensor and of the landmark. This concept is formalized in the following Definition 5.1.

Definition 5.1 (Perception of a landmark attained by a mobile sensor)**.** *The* perception *of a landmark $\ell = (q, \hat{m})$ attained by a mobile sensor $s = (p, \hat{n}, f)$ is defined as $f(p, \hat{n}, q, \hat{m})$, and it is denoted as* $\mathrm{per}(s, \ell)$*. Namely, we let*

$$\mathrm{per}(s, \ell) = f(p, \hat{n}, q, \hat{m}). \tag{5.4}$$

The coverage of a finite set of landmarks attained by a mobile sensor is defined as the sum of the perceptions of the landmarks in the set attained by the sensor. This concept is formalized in the following Definition 5.2.

Definition 5.2 (Coverage of a finite set of landmarks attained by a mobile sensor)**.** *Consider now a finite set $\mathcal{L} = \{\ell_1, \ldots, \ell_M\}$ of landmarks. The* coverage *of the set $\mathcal{L}$ attained by a sensor s is defined as the sum of the perceptions of the landmarks in $\mathcal{L}$ attained by the sensor. Namely, we let*

$$\mathrm{cov}(s, \mathcal{L}) = \sum_{\ell \in \mathcal{L}} \mathrm{per}(s, \ell). \tag{5.5}$$

Finally, the coverage of a finite set of landmarks attained by a set of mobile sensors is defined with respect to a partition of the landmarks among the sensors, and it is given by the sum of the coverages attained by each sensor for its subset of landmarks. This concept is formalized in the following Definition 5.3.

Definition 5.3 (Coverage of a finite set of landmarks attained by a team of mobile sensors)**.** *Consider a team of mobile sensors $\mathcal{S} = (s_1, \ldots, s_N)$ and a finite set of landmarks $\mathcal{L} = \{\ell_1, \ldots, \ell_M\}$. Let $\mathcal{P} = (\mathcal{L}_1, \ldots, \mathcal{L}_N)$ be a partition of $\mathcal{L}$, so that each subset $\mathcal{L}_i$ is assigned to the sensor s_i. We define the* coverage *of the set $\mathcal{L}$ attained by the team $\mathcal{S}$ with respect to the partition $\mathcal{P}$ as the sum of the coverage of $\mathcal{L}_i$ attained by s_i for $i \in \{1, \ldots, N\}$, and we denote it, with some abuse of notation, as* $\mathrm{cov}(\mathcal{S}, \mathcal{P})$*. Namely, we let*

$$\mathrm{cov}(\mathcal{S}, \mathcal{P}) = \sum_{i=1}^{N} \mathrm{cov}(s_i, \mathcal{L}_i).$$

Remark 5.1. *Since the footprint of a sensor is nonnegative by definition, the perception of a landmark attained by a sensor, the coverage of a finite set of landmarks attained by a sensor, and the coverage of a finite set of landmarks attained by a team of mobile sensors are all nonnegative. Note also that a lower value of perception (respectively, coverage) corresponds to a better perception (respectively, coverage) of the landmark (respectively, landmarks).*

Remark 5.2. *In most realistic applications, we can expect M to be one or two orders of magnitude larger than N.*

5.3 Voronoi tessellations

In this section, we introduce a generalized definition of *Voronoi tessellation*. For a classical definition of Voronoi tessellation, see for example [44].

Definition 5.4 (ϵ-Voronoi tessellation of a set of landmarks induced by a team of sensors)**.** *Consider a team of mobile sensors $\mathcal{S} = (s_1, \ldots, s_N)$ and a finite set of landmarks $\mathcal{L} = \{\ell_1, \ldots, \ell_M\}$. Let $\epsilon \geq 0$. An* ϵ-Voronoi tessellation *of $\mathcal{L}$ induced by the sensors $\mathcal{S}$ is a partition $\mathcal{V} = (\mathcal{L}_1, \ldots, \mathcal{L}_N)$ of $\mathcal{L}$ such that,*

$$\forall \ell \in \mathcal{L}_i, \quad \operatorname{per}(s_i, q) \leq \operatorname{per}(s_j, q) + \epsilon \quad \forall s_j \in \mathcal{S}.$$

A 0-Voronoi tessellation is also called simply a Voronoi tessellation*. If $\mathcal{V} = (\mathcal{L}_1, \ldots, \mathcal{L}_N)$ is a Voronoi tessellation, then $\mathcal{L}_i$ is called the* Voronoi cell *of the sensor s_i.*

The intuition behind Definition 5.4 is that a Voronoi cell $\mathcal{L}_i$ of a sensor s_i contains the landmarks that s_i can see better than any other sensor in the team. However, we allow that $\mathcal{L}_i$ also contain landmarks whose perception by some other sensor s_j is better (i.e., smaller) than the perception attained by s_i, as long as the difference between the two perceptions is no larger than ϵ.

Remark 5.3. *Given the landmarks $\mathcal{L}$ and the sensors $\mathcal{S}$, and given $\epsilon \geq 0$, an ϵ-Voronoi tessellation is not necessarily unique.*

Remark 5.4. *Our definition of Voronoi tessellation reduces to the one in [44] for a particular choice of the sensors' footprints, namely for $f(p, \hat{n}, q, \hat{m}) = \|q - p\|^2$.*

Remark 5.5. *If $\mathcal{V}$ is an ϵ-Voronoi tessellation, then $\mathcal{V}$ is also an ϵ'-Voronoi tessellation for any $\epsilon' > \epsilon$. In particular, if $\mathcal{V}$ is a Voronoi tessellation, then it is also an ϵ-Voronoi tessellation for any $\epsilon > 0$.*

5.4 Problem formulation

In this section, we give a formal description of the coverage problem addressed in this chapter.

Consider a team of mobile sensors $\mathcal{S} = (s_1, \ldots, s_N)$, with $s_i = (p_i, \hat{n}_i, f_i)$, and a finite set of landmarks $\mathcal{L} = \{\ell_1, \ldots, \ell_M\}$. Consider N compact sets $\Omega_1, \ldots, \Omega_N$, with $\Omega_i \subset \mathbb{R}^3 \times \mathcal{S}^2$. Each subset Ω_i is the region where the physical motion of sensor s_i is confined. Our objective is to find a partition $\mathcal{P} = (\mathcal{L}_1, \ldots, \mathcal{L}_N)$ of $\mathcal{L}$, and the positions and orientations of the sensors,

such that the coverage of the set $\mathcal{L}$ attained by the team $\mathcal{S}$ with respect to the partition $\mathcal{P}$ is minimized. This objective can be formalized as the following optimization problem:

$$\begin{aligned} \underset{\substack{p_1,\ldots,p_N\in\mathbb{R}^3\\ \hat{n}_1,\ldots,\hat{n}_N\in\mathcal{S}^2\\ \mathcal{P}=(\mathcal{L}_1,\ldots,\mathcal{L}_N)}}{\text{minimize}} \quad & \operatorname{cov}(\mathcal{S},\mathcal{P}), \\ \text{subject to} \quad & (p_i,\hat{n}_i)\in\Omega_i \quad \forall i\in\{1,\ldots,N\}, \\ & \mathcal{L}_i\cap\mathcal{L}_j=\emptyset \quad \forall i,j\in\{1,\ldots,N\}, \\ & \mathcal{L}_1\cup\ldots\cup\mathcal{L}_N=\mathcal{L}. \end{aligned} \tag{5.6}$$

Problem (5.6) is a nonconvex and mixed-integer problem. Consequently, to compute the global optimum may be computationally demanding. A globally optimal solution of Problem (5.6) can be pursued by the following approach. Introduce the scalars $\sigma_{i,j}$ with $i\in\{1,\ldots,N\}$ and $j\in\{1,\ldots,M\}$. For each $\ell_j\in\mathcal{L}$, let $\sigma_{i,j}=1$ if $\ell_j\in\mathcal{L}_i$ and $\sigma_{i,j}=0$ otherwise. Then, Problem (5.6) can be recast as

$$\begin{aligned} \underset{\substack{p_1,\ldots,p_N\in\mathbb{R}^3\\ \hat{n}_1,\ldots,\hat{n}_N\in\mathcal{S}^2\\ \sigma_{1,1},\ldots,\sigma_{N,M}}}{\text{minimize}} \quad & \sum_{i=1}^{N}\sum_{j=1}^{M}\sigma_{i,j}\operatorname{per}(\mathcal{s}_i,\ell_j), \\ \text{subject to} \quad & (p_i,\hat{n}_i)\in\Omega_i \quad \forall i,j=\{1,\ldots,N\}, \\ & \sigma_{i,j}\in\{0,1\} \quad \forall (i,j)\in\{1,\ldots,N\}\times\{1,\ldots,M\}, \\ & \sigma_{1,j}+\ldots+\sigma_{N,j}=1 \quad \forall j\in\{1,\ldots,M\}. \end{aligned} \tag{5.7}$$

If in (5.7) we relax the integrity constraint $\sigma_{i,j}\in\{0,1\}$ to $\sigma_{i,j}\in[0,1]$, we obtain an optimization problem that is still nonconvex, but continuous. Namely, we have

$$\begin{aligned} \underset{\substack{p_1,\ldots,p_N\in\mathbb{R}^3\\ \hat{n}_1,\ldots,\hat{n}_N\in\mathcal{S}^2\\ \sigma_{1,1},\ldots,\sigma_{N,M}}}{\text{minimize}} \quad & \sum_{i=1}^{N}\sum_{j=1}^{M}\sigma_{i,j}\operatorname{per}(\mathcal{s}_i,\ell_j), \\ \text{subject to} \quad & (p_i,\hat{n}_i)\in\Omega_i \quad \forall i,j=\{1,\ldots,N\}, \\ & \sigma_{i,j}\in[0,1] \quad \forall (i,j)\in\{1,\ldots,N\}\times\{1,\ldots,M\}, \\ & \sigma_{1,j}+\ldots+\sigma_{N,j}=1 \quad \forall j\in\{1,\ldots,M\}. \end{aligned} \tag{5.8}$$

The following Lemma 5.1 shows that a globally optimal solution of Problem (5.7) is readily obtained by a gloablly optimal solution of Problem (5.8).

Lemma 5.1. *Let $\mathcal{O}=(p_1,\ldots,p_N,\hat{n}_1,\ldots,\hat{n}_N,\sigma_{1,1},\ldots,\sigma_{N,M})$ be a globally optimal solution of Problem* (5.8)*. For each $j\in\{1,\ldots,M\}$, let $\mathcal{I}_j=\{i\in\{1,\ldots,N\}:$*

$\sigma_{i,j} \in (0,1)\}$. *Then, any solution* $\mathcal{O}^*$ *obtained from* $\mathcal{O}$ *by substituting* $\sigma_{\bar{i},j} = 1$ *for some* $\bar{i} \in \mathcal{I}_j$ *and* $\sigma_{i,j} = 0$ *for* $i \in \mathcal{I}_j \setminus \{\bar{i}\}$ *is also an optimal solution of Problem* (5.8)*, and it is also an optimal solution of Problem* (5.7)*.*

Proof. First note that the set of admissible solutions of Problem (5.7) is a subset of the set of admissible solutions of Problem (5.8). Therefore, if a globally optimal solution of Problem (5.8) is an admissible solution of Problem (5.7), then it is also a globally optimal solution of Problem (5.7). Then, suppose by contradiction that there exists $\ell_j \in \mathcal{L}$ such that $\operatorname{per}(\mathcal{s}_{i_1}, \ell_j) > \operatorname{per}(\mathcal{s}_{i_2}, \ell_j)$ for some $i_1, i_2 \in \mathcal{I}_j$. Then, let $\sigma'_{i_1,j} = 0$, $\sigma'_{i_2,j} = \sigma_{i_2,j} + \sigma_{i_1,j}$ and $\sigma'_{i,l} = \sigma_{i,l}$ for all $(i,l) \notin \{(i_1,j),(i_2,j)\}$. Let $\mathcal{O}'$ be the solution obtained from $\mathcal{O}$ by substituting each $\sigma_{i,j}$ with $\sigma'_{i,j}$. Note that $\mathcal{O}'$ is still an admissible solution of (5.8), since $0 \leq \sigma'_{i_2,j} = \sigma_{i_1,j} + \sigma_{i_2,j} \leq \sigma_{1,j} + \ldots + \sigma_{N,j} = 1$, and $\sigma'_{1,j} + \ldots + \sigma'_{N,j} = \sigma_{1,j} + \ldots + \sigma_{N,j} = 1$. Moreover, we have

$$\begin{aligned}\sum_{i=1}^{N}\sum_{l=1}^{M} \sigma'_{i,l} \operatorname{per}(\mathcal{s}_i, \ell_l) = &\sum_{i=1}^{N}\sum_{l=1}^{M} \sigma_{i,j} \operatorname{per}(\mathcal{s}_i, \ell_l) \\ &+ \sigma_{i_1,j}(\operatorname{per}(\mathcal{s}_{i_2}, \ell_j) - \operatorname{per}(\mathcal{s}_{i_1}, \ell_j)) \\ &< \sum_{i=1}^{N}\sum_{l=1}^{M} \sigma_{i,j} \operatorname{per}(\mathcal{s}_i, \ell_l).\end{aligned}$$

Therefore, $\mathcal{O}'$ is an admissible solution of (5.8) and yields a smaller value of the cost function than $\mathcal{O}$ does. But this is a contradiction, since $\mathcal{O}$ is gloablly optimal by assumption. We must conclude that, for each $j \in \{1,\ldots,M\}$, we have $\operatorname{per}(\mathcal{s}_{i_1}, \ell_j) = \operatorname{per}(\mathcal{s}_{i_2}, \ell_j)$ for all $i_1, i_2 \in \mathcal{I}_j$. Consequently, $\mathcal{O}^*$ is another globally optimal solution of Problem (5.8). Since $\mathcal{O}^*$ is an admissible solution of Problem (5.7), we can conclude that $\mathcal{O}^*$ is also a globally optimal solution for Problem (5.7). □

Thanks to Lemma 5.1, we can solve Problem (5.8) and obtain an optimal solution of Problem (5.6). Note that Problem (5.8) is still a nonconvex optimization problem. However, depending on the nature of the sensors' footprints and of the sets Ω_i, Problem (5.8) may belong to a special class of nonconvex optimization problems, for which there exist algorithms with proven convergence properties. Consider, for example, the following scenarios.

Assumption 5.1. *For each sensor* $\mathcal{s}_i \in \mathcal{S}$*, let*

$$\tilde{f}_i(p,\theta,\phi,q,\hat{m}) = f_i(p, \operatorname{sph}(\theta,\phi), q, \hat{m}),$$

where $\tilde{f}(\cdot,\cdot,\cdot,q,\hat{m})$ *is globally Lipschitz for any* $(q,\hat{m}) \in \mathbb{R}^3 \times \mathcal{S}^2$*. Moreover, let*

$$\Omega_i = \{(p,\hat{n}) \in \mathbb{R}^3 \times \mathcal{S}^2 : \hat{n} = \operatorname{sph}(\theta,\phi),\ g_i(p,\theta,\phi) \leq 0\},$$

where $g_i(\cdot,\cdot,\cdot)$ *is globally Lipschitz.*

Assumption 5.2. *For each sensor $s_i \in \mathcal{S}$, let*

$$\tilde{f}_i(p, \theta, \phi, q, \hat{m}) = f_i(p, \mathrm{sph}(\theta, \phi), q, \hat{m}),$$

where $\tilde{f}(\cdot, \cdot, \cdot, q, \hat{m})$ is convex for any $(q, \hat{m}) \in \mathbb{R}^3 \times \mathcal{S}^2$. Moreover, let

$$\Omega_i = \{(p, \hat{n}) \in \mathbb{R}^3 \times \mathcal{S}^2 : \hat{n} = \mathrm{sph}(\theta, \phi),\ g_i(p, \theta, \phi) \leq 0\},$$

where $g_i(\cdot, \cdot, \cdot)$ is convex.

Under Assumption 5.1, Problem (5.8) is a Lipschitz optimization problem [82]; under Assumption 5.2, Problem (5.8) is a convex optimization problem [81].

However, even under one of Assumptions 5.1 and 5.2, computing a globally optimal solution of Problem (5.8) may be computationally demanding with respect to the available resources, especially if the number of sensors and landmarks is large. In fact, the number of optimization variables in Problem (5.8) scales with the number of sensors times the number of landmarks. In rest of this chapter, we are going to outline several control strategies to attack Problem (5.6) and achieve suboptimal solutions in a distributed way.

Remark 5.6. *There exist several realistic scenarios that can be modelled as particular cases of Problem* (5.6)*. If the landmarks represent each a point or small volume of interest in the space, and are not characterized by any particular orientation, then the sensor footprints do not depend on the landmark orientation $\hat{m}$, and we have $f(p, \hat{n}, q, \hat{m}) = \tilde{f}(p, \hat{n}, q)$. If the sensors are omnidirectional, i.e., their perception of the surrounding environment only depends on the distance between the sensor and the point under observation, then the sensor footprints do not depend on $\hat{n}$ either, and we have $f(p, \hat{n}, q, \hat{m}) = \tilde{f}(\|q - p\|)$. If we want to consider a setup similar to [40], where the sensors' perception degrades with the squared distance from the point under observation, we can simply set $f(p, \hat{n}, q, \hat{m}) = \|q - p\|^2$. Planar settings are obtained by having, for each landmark $(q, \hat{m})$, $q = [q_x, q_y, 0]^\top$ for some $q_x, q_y \in \mathbb{R}$ and $\hat{m} = [m_x, m_y, 0]^\top$ for some $[m_x, m_y]^\top \in \mathcal{S}^1$, and having, for each sensor s_i, $\Omega_i \subseteq \{(p, \hat{n}) \in \mathbb{R}^3 \times \mathcal{S}^2 : p = [p_x, p_y, 0]^\top, p_x, p_y \in \mathbb{R}, \hat{n} = [n_x, n_y, 0]^\top, [n_x, n_y]^\top \in \mathcal{S}^1\}$.*

5.5 Necessary conditions for optimality

Our first step in attacking Problem (5.6) is to individuate some necessary conditions for a solution to be globally optimal. First, we are going to show that any globally optimal solution of (5.6) corresponds to a Voronoi tessellation. This result is formalized in the following Lemma 5.2.

Lemma 5.2. *Let $\mathcal{O} = (p_1, \ldots, p_N, \hat{n}_1, \ldots, \hat{n}, \mathcal{L}_1, \ldots, \mathcal{L}_N)$ constitute an optimal solution of* (5.6)*. Then $\mathcal{V} = (\mathcal{L}_1, \ldots, \mathcal{L}_N)$ is a Voronoi tessellation of $\mathcal{L}$ induced by the sensors $\mathcal{S} = (s_1, \ldots, s_N)$, where $s_i = (p_i, \hat{n}_i, f_i)$.*

Proof. Suppose by contradiction that $\mathcal{V}$ is not a Voronoi tessellation. Then, there exist $s_i, s_j \in \mathcal{S}$ and $\ell \in \mathcal{L}_i$ such that $\mathrm{per}(s_j, \ell) < \mathrm{per}(s_i, \ell)$. Denote as $\mathcal{O}'$ the solution of (5.6) obtained from $\mathcal{O}$ by substituting $\mathcal{L}_i$ with $\mathcal{L}'_i = \mathcal{L}_i \setminus \{\ell\}$ and $\mathcal{L}_j$ with $\mathcal{L}'_j = \mathcal{L}_j \cup \{\ell\}$. Let $\mathcal{V}'$ be the partition of $\mathcal{L}$ obtained from $\mathcal{V}$ by substituting $\mathcal{L}_i$ with $\mathcal{L}'_i$ and $\mathcal{L}_j$ with $\mathcal{L}'_j$. Then, $\mathrm{cov}(\mathcal{S}, \mathcal{V}') = \mathrm{cov}(\mathcal{S}, \mathcal{V}) - \mathrm{per}(s_i, \ell) + \mathrm{per}(s_j, \ell) < \mathrm{cov}(\mathcal{S}, \mathcal{V})$. Consequently, $\mathcal{O}'$ yields a lower value of the cost function than $\mathcal{O}$. But this is a contradiction, since, by hypothesis, $\mathcal{O}$ is an optimal solution of (5.6). □

Next, we are going to show that in a globally optimal solution, the position and orientation of each sensor s_i minimize the coverage attained by that sensor over its set of landmarks $\mathcal{L}_i$. This result is formalized in the following Lemma 5.3.

Lemma 5.3. *Let $\mathcal{O} = (p_1, \ldots, p_N, \hat{n}_1, \ldots, \hat{n}, \mathcal{L}_1, \ldots, \mathcal{L}_N)$ constitute an optimal solution of* (5.6)*. Then, for each sensor $s_i = (p_i, \hat{n}_i, f_i)$, we have*

$$\mathrm{cov}(s_i, \mathcal{L}_i) \leq \mathrm{cov}((p, \hat{n}, f_i), \mathcal{L}_i) \quad \forall (p, \hat{n}) \in \Omega_i.$$

Proof. Suppose by contradiction that there exist $s_i = (p_i, \hat{n}_i, f_i) \in \mathcal{S}$ and $(p, \hat{n}) \in \Omega_i$ such that $\mathrm{cov}((p, \hat{n}, f_i), \mathcal{L}_i) < \mathrm{cov}(s_i, \mathcal{L}_i)$. Denote as $\mathcal{O}'$ the solution of (5.6) obtained from $\mathcal{O}$ by substituting p_i and $\hat{n}_i$) with p and $\hat{n}$. Let $\mathcal{S}'$ be the sensor team obtained by $\mathcal{S} = (s_1, \ldots, s_N)$ by substituting $s_i = (p_i, \hat{n}_i)$ with $s'_i = (p, \hat{n}, f_i)$. Then, denoting $\mathcal{P} = (\mathcal{L}_1, \ldots, \mathcal{L}_N)$, we have $\mathrm{cov}(\mathcal{S}', \mathcal{P}) = \mathrm{cov}(\mathcal{S}, \mathcal{P}) + \mathrm{cov}(s'_i, \mathcal{L}_i) - \mathrm{cov}(s_i, \mathcal{L}_i) < \mathrm{cov}(\mathcal{S}, \mathcal{P})$. Consequently, $\mathcal{O}'$ yields a lower value of the cost function than $\mathcal{O}$. But this is a contradiction, since, by hypothesis, $\mathcal{O}$ is an optimal solution of (5.6). □

5.6 Generalized discrete Lloyd descent

In this section, we present our first coverage algorithm. This algorithm can be seen as a generalization of the *discrete Lloyd descent* [44]. Therefore, we shall refer to this algorithm as the *generalized discrete Lloyd descent (*GDLD*)*.

The GDLD assumes that a team of sensors $\mathcal{S} = (s_1, \ldots, s_N)$ is assigned, with $s_i = (p_i, \hat{n}_i, f_i)$. The footprints $f_1, \ldots, f_N$ of the sensors are fixed, while the positions $p_1, \ldots, p_N$ and the orientations $\hat{n}_1, \ldots, \hat{n}_N$ can be changed at each iteration, since they are optimization variables. A finite set of landmarks $\mathcal{L} = \{\ell_1, \ldots, \ell_M\}$, is also given, and it is initially partitioned as $\mathcal{P} = (\mathcal{L}_1, \ldots, \mathcal{L}_N)$. Moreover, N compact subsets $\Omega_1, \ldots, \Omega_N$ of $\mathbb{R}^3 \times \mathcal{S}^2$ are given, and each sensor s_i, with $i \in \{1, \ldots, N\}$, can be moved within Ω_i. Finally, a tolerance $\epsilon > 0$ is given. The algorithm aims at progressively improving

the coverage of the set $\mathcal{L}$ attained by the sensors $\mathcal{S}$ with respect to the partition $\mathcal{P}$. The coverage is improved by adjusting iteratively the positions and orientations of the sensors, as well as the partition $\mathcal{P}$. The coverage is improved by at least ϵ at each iteration, and the algorithm stops after a finite number of iterations. The partition $\mathcal{P}$ is improved by considering a pair of sensors s_i and s_j at each iteration, and rearranging the landmarks in $\mathcal{L}_i \cup \mathcal{L}_j$ so that each landmark is assigned to the sensor, between s_i and s_j, that has the best perception of that landmark. For each sensor s_i, the algorithm maintains a set $\mathcal{Z}_i \subseteq \mathcal{S} \setminus \{s_i\}$, which is the subset of the other sensors s_j such that some landmarks may be transferred between $\mathcal{L}_i$ and $\mathcal{L}_j$. The algorithm terminates when $\mathcal{Z}_i = \emptyset$ for each sensor. Upon termination, the partition $\mathcal{P}$ is an ϵ-Voronoi tessellation, and the position and orientation of each sensor s_i are optimized with respect to the coverage of $\mathcal{L}_i$. The GDLD uses a routine that optimizes the coverage of a set of landmarks attained by a single mobile sensor. Namely, the algorithm uses a routine that solves the following optimization problem:

$$\begin{aligned} \underset{p,\hat{n}}{\text{minimize}} \quad & \operatorname{cov}(s, \mathcal{L}), \\ \text{subject to} \quad & (p, \hat{n}) \in \Omega, \end{aligned} \tag{5.9}$$

where $s = (p, \hat{n}, f)$, and where the footprint $f : \mathbb{R}^3 \times \mathcal{S}^2 \to \mathbb{R}_{\geq 0}$, the landmarks $\mathcal{L} = (\ell_1, \ldots, \ell_M)$ and the domain $\Omega \subset \mathbb{R}^3 \times \mathcal{S}^2$ are given.

Note that, in most realistic scenarios, there exist well-established routines to solve Problem (5.9). Namely, if Assumption 5.1 is satisfied, Problem (5.9) is a Lipschitz optimization problem. Similarly, if Assumption 5.2 is satisfied, Problem (5.9) is a convex optimization problem. In both these cases, there exist well-established routines to compute an optimal solution (see [82] and [81]). This scenario is formalized in the following Assumption 5.3.

Assumption 5.3. *There exists a routine* `optCov` *that returns a globally optimal solution of Problem* (5.9)*. Namely, if* $(p, \hat{n}) \leftarrow$ `optCov`$(s, \mathcal{L}, \Omega)$*, then* $(p, \hat{n})$ *is a globally optimal solution of Problem* (5.9)*.*

The GDLD is outlined below as Algorithm 5.3. The convergence properties of Algorithm 5.3 are outlined in the following Theorem 5.1.

Algorithm 5.3. Generalized Discrete Lloyd Descent.

1: assign the mobile sensors $\mathcal{S} = (s_1, \ldots, s_N)$, with $s_i = (p_i, \hat{n}_i, f_i)$
2: assign the landmarks $\mathcal{L} = (\ell_1, \ldots, \ell_M)$
3: assign a partition $\mathcal{P} = (\mathcal{L}_1, \ldots, \mathcal{L}_N)$ of $\mathcal{L}$
4: assign $\epsilon > 0$.
5: set $\mathcal{Z}_i = \mathcal{S} \setminus \{s_i\}$ for $i \in \{1, \ldots, N\}$

while $\mathcal{Z}_i \neq \emptyset$ for some $i \in \{1, \dots, N\}$ **do**
  pick s_i such that $\mathcal{Z}_i$ is not empty
  pick $s_j \in \mathcal{Z}_i$
  for $\ell \in \mathcal{L}_i$ **do**
    if $\mathrm{per}(s_j, \ell) < \mathrm{per}(s_i, \ell) - \epsilon$ **then**
      transfer ℓ from $\mathcal{L}_i$ to $\mathcal{L}_j$
    end if
  end for
  for $\ell \in \mathcal{L}_j$ **do**
    if $\mathrm{per}(s_i, \ell) < \mathrm{per}(s_j, \ell) - \epsilon$ **then**
      transfer ℓ from $\mathcal{L}_j$ to $\mathcal{L}_i$
    end if
  end for
  if one or more landmarks have been transferred **then**
    $\mathcal{Z}_i \leftarrow \mathcal{S} \setminus \{s_i\}$
    $\mathcal{Z}_j \leftarrow \mathcal{S} \setminus \{s_j\}$
    $(p_i, \hat{n}_i) \leftarrow \texttt{optCov}(s_i, \mathcal{L}_i, \Omega_i)$
    $(p_j, \hat{n}_j) \leftarrow \texttt{optCov}(s_j, \mathcal{L}_j, \Omega_j)$
  else
    remove s_j from $\mathcal{Z}_i$
    if $s_i \in \mathcal{Z}_j$, remove s_i from $\mathcal{Z}_j$
  end if
end while

Theorem 5.1. *Let Assumption 5.3 hold. Then, algorithm 5.3 terminates in a finite number of iterations. Namely, the number of iterations is upper-bounded by*

$$\nu_{\max} = \left\lceil \frac{2N(\mathrm{cov}_0 - \mathrm{cov}^*)}{\epsilon} \right\rceil + N(N-1),$$

where cov_0 *denotes the value of* $\mathrm{cov}(\mathcal{S}, \mathcal{P})$ *upon initialization,* cov^* *is the value of* $\mathrm{cov}(\mathcal{S}, \mathcal{P})$ *attained by an optimal solution of Problem* (5.6)*, and* $\lceil x \rceil = \min\{n \in \mathbb{N} : x \leq n\}$ *is the ceiling function. When the algorithm terminates,* $\mathcal{P}$ *is an* ϵ*-Voronoi tessellation of* $\mathcal{L}$ *induced by the sensors* $\mathcal{S}$*, and the pose of each sensor* s_i *minimizes* $\mathrm{cov}(s_i, \mathcal{L}_i)$ *within* Ω_i*, namely*

$$\mathrm{cov}(s_i, \mathcal{L}_i) \leq \mathrm{cov}((p, \hat{n}), \mathcal{L}_i) \quad \forall (p, \hat{n}) \in \Omega_i. \tag{5.10}$$

Proof.

Part 1. (The number of iterations is upper-bounded by $\nu_{\max}$.) The algorithm terminates when $\mathcal{Z}_i = \emptyset$ for all $i \in \{1, \dots, N\}$. Denote as $z(t)$ the sum of the

cardinalities of the sets $\mathcal{Z}_i$ after the t-th iteration, with $z(0)$ being the value attained upon initialization. Similarly, introduce $\mathcal{S}(t)$ and $\mathcal{P}(t)$. Upon initialization, we have $\mathcal{Z}_i = \mathcal{S} \setminus \{s_i\}$ for each sensor, implying $z(0) = N(N-1)$, and $\operatorname{cov}(\mathcal{S}(0), \mathcal{P}(0)) = \operatorname{cov}_0$. Each iteration produces one of the two following effects: one among the sets $\mathcal{Z}_1, \ldots, \mathcal{Z}_N$ loses one element; the value of $\operatorname{cov}(\mathcal{S}, \mathcal{P})$ is reduced by at least ϵ, and two of the sets $\mathcal{Z}_1, \ldots, \mathcal{Z}_N$ acquire at most $N-2$ elements each. Therefore, on each iteration, we have either of the two following difference inclusions:

$$\begin{cases} \operatorname{cov}(\mathcal{S}(t^+), \mathcal{P}(t^+)) \leq \operatorname{cov}(\mathcal{S}(t), \mathcal{P}(t)) - \epsilon, \\ z(t^+) \leq z(t) + 2(N-2), \end{cases}$$

or

$$\begin{cases} \operatorname{cov}(\mathcal{S}(t^+), \mathcal{P}(t^+)) = \operatorname{cov}(\mathcal{S}(t), \mathcal{P}(t)), \\ z(t^+) \leq z(t) - 1, \end{cases}$$

with initial conditions given by

$$\begin{cases} \operatorname{cov}(\mathcal{S}(0), \mathcal{P}(0)) = \operatorname{cov}_0, \\ z(0) = N(N-1). \end{cases}$$

The algorithm terminates when $z(t) = 0$. Consider the function

$$V(t) = 2N \operatorname{cov}(\mathcal{S}(t), \mathcal{P}(t)) + \epsilon z(t).$$

Note that $V(t)$ is lower-bounded by $V^* = 2N \operatorname{cov}^*$ and that $V(0) = 2N \operatorname{cov}_0 + \epsilon N(N-1)$. On each iteration, we have either

$$\begin{aligned} V(t^+) \leq & 2N(\operatorname{cov}(\mathcal{S}(t), \mathcal{P}(t)) - \epsilon) + \epsilon(z(t) + 2(N-2)) \\ \leq & V(t) - 4\epsilon \end{aligned}$$

or

$$V(t^+) = 2N \operatorname{cov}(\mathcal{S}(t), \mathcal{P}(t)) + \epsilon(z(t) - 1) \leq V(t) - \epsilon.$$

In both cases, we have $V(t^+) \leq V(t) - \epsilon$. Since $V^* \leq V(t) \leq V(0)$ for all $t \in \mathbb{N} \cup 0$, the number of iterations is upper-bounded by

$$\begin{aligned} \nu_{\max} &= \left\lceil \frac{V(0) - V^*}{\epsilon} \right\rceil \\ &= \left\lceil \frac{2N(\operatorname{cov}_0 - \operatorname{cov}^*) + \epsilon N(N-1)}{\epsilon} \right\rceil \\ &= \left\lceil \frac{2N(\operatorname{cov}_0 - \operatorname{cov}^*)}{\epsilon} \right\rceil + N(N-1). \end{aligned}$$

Part 2. (When the algorithm terminates, $\mathcal{P}$ is an ϵ-Voronoi tessellation.) When the algorithm terminates, we have $\mathcal{Z}_1 = \ldots = \mathcal{Z}_N = \emptyset$. Therefore, for any two sensors s_i, s_j, the latest iteration involving s_i and s_j has not led to transferring any landmark. In particular, this statement implies that $\mathrm{per}(s_i, \ell) \leq \mathrm{per}(s_j, \ell) + \epsilon$ for any $\ell \in \mathcal{L}_i$. Since the reasoning can be applied to any couple of sensors, we have that, by Definition 5.4, $\mathcal{P} = (\mathcal{L}_1, \ldots, \mathcal{L}_N)$ is an ϵ-Voronoi tessellation of $\mathcal{L}$.

Part 3. (When the algorithm terminates, the position and orientation of each sensor s_i minimize $\mathrm{cov}(s_i, \mathcal{L}_i)$.) Each iteration of the algorithm is concluded with a call to the procedure `optCov`, which optimizes the pose of two sensors s_i and s_j. Each sensor needs to be involved at least in $(N-1)$ iterations for the algorithm to terminate (because for each sensor s_i, the set $\mathcal{Z}_i$ may lose up to one element per iteration). Therefore, at the end of the algorithm each sensor s_i has its position and orientation optimized (within Ω_i) with respect to the set of landmarks $\mathcal{L}_i$. Namely, (5.10) holds, which concludes the proof. □

Remark 5.7. *The upper bound on the number of iterations provided by Theorem 5.1 is conservative, because it is based on the worst-case scenario that one landmark is transferred at each iteration, and the transfer leads to an improvement of its perception of ϵ. In general, multiple landmarks are transferred within the same iteration, and when a landmark is transferred its perception improves by a value larger than or equal to ϵ. The main purpose of the theorem is rather to show formally that Algorithm 5.3 is guaranteed to terminate in a finite number of iterations.*

The GDLD is amenable to distributed implementations, in the sense that the algorithm may be deployed as N separate threads, each running on a microprocessor installed on one of the mobile sensors. In fact, the sensors only need to perform local computation (when optimizing their position and orientation) or to communicate pairwise (when transferring landmarks to each other). Moreover, different iterations of the algorithm that involve different sensors may be performed in parallel. However, a distributed implementation of the GDLD presents the following challenges.

- To ensure that the algorithm evolves in a consistent way, some form of synchronization among the different threads is necessary. For example, if a sensor s_i is involved in one iteration of the algorithm with another sensor, it cannot be involved in a different iteration with a third sensor before the first iteration is terminated.
- Each sensor s_i can keep track of its own set $\mathcal{Z}_i$, but, in general, none of the sensors is aware of the state of all the sets $\mathcal{Z}_1, \ldots, \mathcal{Z}_N$ at the same time. This aspect introduces a new challenge in making the sensors aware of the algorithm termination, since we defined the algorithm to be terminated when all the sets $\mathcal{Z}_i$ are empty.

In the following Section 5.7, we are going to propose a distributed, parallel, and asynchronous implementation of the GDLD that takes both these challenges into account.

5.7 Distributed implementation of the generalized discrete Lloyd descent

In this section, we discuss a distributed implementation of the GDLD. The algorithm is implemented as a parallel program, where each sensor executes its own instructions and occasionaly interacts with other sensors. A parallel implementation has inherent advantages: for example, it allows to execute multiple iterations of the algorithm at the same time, as long as the concurrent iterations involve different pairs of sensors.

It is out of the scope of this work to discuss parallel programs and the related programming techniques, since their use is limited to this particular section. We use ideas and terms (such as *locks* and *tokens*) related to parallel programs, and we assume that the reader has some intuitive understanding of them. The interested reader is referred to [83] for an introduction to parallel programs.

We consider a team of N sensors, $\mathcal{S} = (s_1, \dots, s_N)$, and a set of M landmarks, $\mathcal{L} = (\ell_1, \dots, \ell_M)$. The algorithm is initialized as follows. Each landmark is assigned to one of the sensors at random, so that an initial partition $\mathcal{P} = (\mathcal{L}_1, \dots, \mathcal{L}_N)$ is defined. Each sensor s_i is assigned a domain $\Omega_i \subset \mathbb{R}^3 \times \mathbb{S}^2$ for its position and orientation, and a set $\mathcal{Z}_i$ to keep track of the interactions with the other sensors. Each of these sets is initialized as $\mathcal{Z}_i = \mathcal{S} \setminus \{s_i\}$. A common tolerance $\epsilon > 0$ is known to all the sensors.

In this scenario, the landmarks are a shared resource that is used concurrently by different sensors. Therefore, we need to use some synchronization primitives, such as *locks* or *semaphores*, to make sure that the different sensors use this resource consistently. In the proposed implementation, each sensor is assigned a lock. We denote the lock assigned to s_i as $\texttt{lock}_i$. Finally, one of the sensors is assigned a token, which we denote as `token`. The token represents the ability of a sensor to contact another sensor, and redistribute the landmarks assigned to the two sensors. The token is passed from sensor to sensor, so that, at any time, one and only one sensor can contact another sensor and initiate a landmark redistribution. This mechanism introduces some limitation to the parallelism in the sensors' interactions, but guarantees the absence of deadlocks. The initialization procedure is formalized as the following Algorithm 5.4.

Algorithm 5.4. Initialization of the distributed implementation of the GDLD.

1: the sensors $\mathcal{S} = (s_1, \dots, s_N)$, with $s_i = (p_i, \hat{n}_i, f_i)$, are given
2: the landmarks $\mathcal{L} = \{\ell_1, \dots, \ell_M\}$ are given
3: each landmark is assigned to one of the sensors, so that an initial partition $\mathcal{P} = (\mathcal{L}_1, \dots, \mathcal{L}_N)$ of $\mathcal{L}$ is defined
4: assign the tolerance $\epsilon > 0$
5: **for** $s_i \in \mathcal{S}$ **do**
6: assign a domain $\Omega_i \subset \mathbb{R}^3 \times \mathbb{S}^2$ to the sensor
7: $\mathcal{Z}_i \leftarrow \mathcal{S} \setminus \{s_i\}$
8: assign a lock $\texttt{lock}_i$ to the sensor
9: **end for**
10: give a `token` to one of the sensors
11: start all the sensors

We call the set of instructions executed by each sensor the *thread* of that sensor. Each thread consists of a loop that is executed repeatedly until the thread terminates. At the beginning of the loop, the sensor, say s_i, acquires its own lock $\texttt{lock}_i$ and checks if it has the `token`. If the sensor s_i does not have the `token`, then it releases $\texttt{lock}_i$ and sleeps for a given amount of time, allowing other sensors to contact it to transfer landmarks. If the sensor s_i does have the `token`, then it chooses another sensor $s_j \in \mathcal{Z}_i$ (we shall see that, at this point in the program, $\mathcal{Z}_i$ cannot be empty). Then, s_i acquires the lock $\texttt{lock}_j$ of s_j. At this point, s_i has to look for a third sensor s_h to give it the `token`. The `token` can be given to a sensor s_h whose set $\mathcal{Z}_h$ is not empty. The logic behind this rule is that, if $\mathcal{Z}_h$ is empty, then s_h will not be able to contact other sensors to transfer landmarks. Therefore, s_i initializes a temporary set $\mathcal{T}_i \leftarrow \mathcal{S}_i \setminus \{s_i, s_j\}$ to keep track of potential recipients for the `token`. A candidate s_h is picked at random in $\mathcal{T}_i$, and s_i acquires $\texttt{lock}_h$. Now s_i can check $\mathcal{Z}_h$. If $\mathcal{Z}_h$ is not empty, then s_i gives the token to s_h and releases $\texttt{lock}_h$. If $\mathcal{Z}_h$ is empty, s_i releases $\texttt{lock}_h$, removes s_h from $\mathcal{T}_i$, and picks another candidate from $\mathcal{T}_i$. If a suitable recipient is not found in $\mathcal{T}_i$, the sensor s_i tries to pass the `token` to s_j. If $\mathcal{Z}_j$ is also empty, s_i keeps the `token`. Thereafter, s_i redistributes the landmarks between itself and s_j, thus updating the sets $\mathcal{L}_i$ and $\mathcal{L}_j$, in the partition $\mathcal{P}$. If at least one landmark is passed from one sensor to the other, then $\mathcal{Z}_i$ is reset to $\mathcal{S} \setminus \{s_i\}$ and $\mathcal{Z}_j$ is reset to $\mathcal{S} \setminus \{s_j\}$; each of the sensors s_i and s_j now optimizes its position and orientation with respect to its new set of landmarks, using the routine `optCov`. If no landmarks are passed from one sensor to the other, then s_j is removed from $\mathcal{Z}_i$, and s_i is removed from $\mathcal{Z}_j$ if it is contained in $\mathcal{Z}_j$; if s_i has kept the `token` and the set $\mathcal{Z}_i$ is now empty, then s_i notifies all the other sensor that the algorithm is terminated (since all the sets $\mathcal{Z}_1, \dots, \mathcal{Z}_N$ are

known to be empty at this point), acquiring and releasing the corresponding lock (except for s_j, whose lock $\texttt{lock}_j$ is already acquired). Finally, s_i releases $\texttt{lock}_i$ and $\texttt{lock}_j$.

The generic thread of a sensor s_i is given below as Algorithm 5.5.

Algorithm 5.5. The thread run on each sensor s_i in the GDLD

while s_i is not terminated **do**
  acquire $\texttt{lock}_i$
  if s_i has the token **then**
    choose s_j in $\mathcal{Z}_i$
    acquire $\texttt{lock}_j$
    $\mathcal{T}_i \leftarrow \mathcal{S} \setminus \{s_i, s_j\}$
    while $\mathcal{T}_i$ is not empty and s_i has the token **do**
      choose s_h in $\mathcal{T}$
      acquire $\texttt{lock}_h$
      if $\mathcal{Z}_h$ is not empty **then**
        give the token to s_h
      else
        remove s_h from $\mathcal{T}_i$
      end if
      release $\texttt{lock}_h$
    end while
    if $\mathcal{T}_i$ is empty and $\mathcal{Z}_j$ is not empty **then**
      give the token to s_j
    end if
    for $\ell \in \mathcal{L}_i$ **do**
      if $\mathrm{per}(s_j, \ell) < \mathrm{per}(s_i, \ell) - \epsilon$ **then**
        transfer ℓ from $\mathcal{L}_i$ to $\mathcal{L}_j$
      end if
    end for
    for $\ell \in \mathcal{L}_j$ **do**
      if $\mathrm{per}(s_i, \ell) < \mathrm{per}(s_j, \ell) - \epsilon$ **then**
        transfer ℓ from $\mathcal{L}_j$ to $\mathcal{L}_i$
      end if
    end for
    if one or more landmarks were transferred **then**
      $\mathcal{Z}_i \leftarrow \mathcal{Z}_i \setminus \{s_j\}$
      $\mathcal{Z}_j \leftarrow \mathcal{Z}_j \setminus \{s_i\}$
      $(p_i, \hat{n}_i) \leftarrow \texttt{optCov}(\mathcal{L}_i, \Omega_i)$
      $(p_j, \hat{n}_j) \leftarrow \texttt{optCov}(\mathcal{L}_j, \Omega_j)$
    else

36: remove s_j from $\mathcal{Z}_i$
37: if $s_i \in \mathcal{Z}_j$, remove s_i from $\mathcal{Z}_j$
38: **if** s_i has the token and $\mathcal{Z}_i$ is empty **then**
39: **for** $s_h \in \mathcal{S} \setminus (\{s_i\} \cup \{s_j\})$ **do**
40: acquire $\texttt{lock}_h$
41: terminate s_h
42: release $\texttt{lock}_h$
43: **end for**
44: terminate s_i and s_j
45: **end if**
46: **end if**
47: release $\texttt{lock}_i$ and $\texttt{lock}_j$
48: **else**
49: release $\texttt{lock}_i$
50: sleep
51: **end if**
52: **end while**

It is worth noting that, in Algorithm 5.5, the segment of the loop that requires most of the computation (lines 20 to 47) is executed after that s_i has passed the token to another sensor. Therefore, that segment can be executed by many different sensors at the same time.

5.8 Simulation of the generalized discrete Lloyd descent

In this section, we present a simulation of the GDLD. Here we consider a scenario where $N = 4$ mobile cameras have to survey a square-shaped room. The room is formally described as

$$\mathcal{Q} = \{(q_x, q_y, 0) : \ -4 \leq q_x, q_y \leq 4\}.$$

Each sensor can move on the room floor, therefore we set

$$\Omega_i = \{(p_i, \hat{n}_i) : \ p_i \in \mathcal{Q}, \ \hat{n}_i = (n_{i,x}, n_{i,y}, 0), (n_{i,x}, n_{i,y}) \in \mathcal{S}^1\}.$$

The sensors all have the same footprint, which is a model of the vision capabilities of the cameras. The cameras have better visibility of an object if the object is central in their cone of view, and if it is neither too far nor too close. We capture this description with the following footprint:

$$f(p, \hat{n}, q, \hat{m}) = \begin{cases} 1, & \text{if } (q-p) \leq 0, \\ 1 - (q-p)^\top \hat{n}, & \text{if } (q-p) > 0 \text{ and } \|q-p\| \leq 1, \\ 1 - \frac{(q-p)^\top \hat{n}}{\|q-p\|^2}, & \text{otherwise.} \end{cases} \tag{5.11}$$

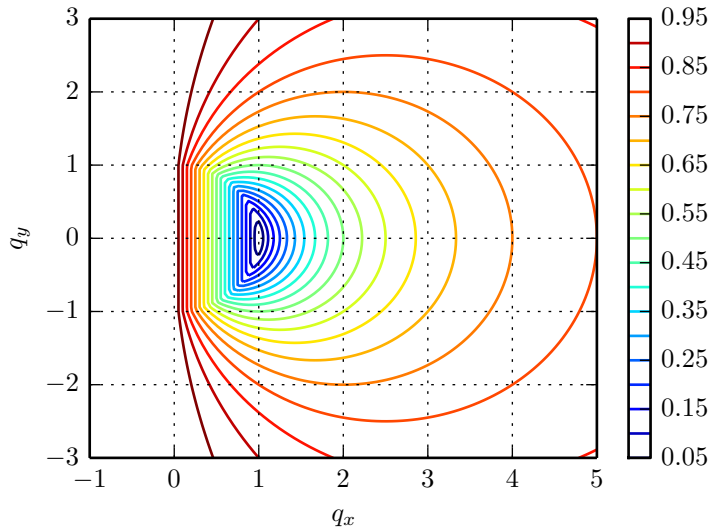


Figure 5.1. Contour plot of footprint (5.11) for $p = 0_3$, $\hat{n} = (1, 0, 0)$ and $q = (q_x, q_y, 0)$, as a function of q_x and q_y.

Note that footprint (5.11) does not depend on the orientation of the landmark, since, for this application, the points in the room are not associated with any orientation in particular. Figure 5.1 illustrates a contour plot of footprint (5.11) for $p = 0_3$, $\hat{n} = (1, 0, 0)$ and $q = (q_x, q_y, 0)$, as a function of q_x and q_y.

The room that the sensors have to survey is abstracted into a set of $M = 400$ landmarks. The landmarks are distributed in the room at equally spaced position, so that each landmark can be seen as a tile in the room floor. Namely, we set $\ell_i = (q_i, \hat{m}_i)$, with $q_i = (q_{i,x}, q_{i,y}, 0)^\top$, for each $i \in \{1, \dots, M\}$. The orientations of the landmarks are not assigned, since they are not accounted for in the sensors' common footprint (5.11). For each sensor $s_i = (p_i, \hat{n}_i)$, the position and orientation are initialized randomly in Ω_i. Each landmark is initially assigned to one of the sensor at random. The GDLD is run with $\epsilon = 10^{-4}$. For practical purposes, the routine `optCov` is implemented as a differential evolution algorithm [84]. The results of the simulation are illustrated in Figure 5.2. In the initial configuration, we have $\mathrm{cov}(\mathcal{S}, \mathcal{P}) \simeq 314$, while, in the final configuration, we have $\mathrm{cov}(\mathcal{S}, \mathcal{P}) \simeq 259$. The final partition of the landmarks is verified to be an ϵ-Voronoi tessellation induced by the final positions and orientations of the sensors.

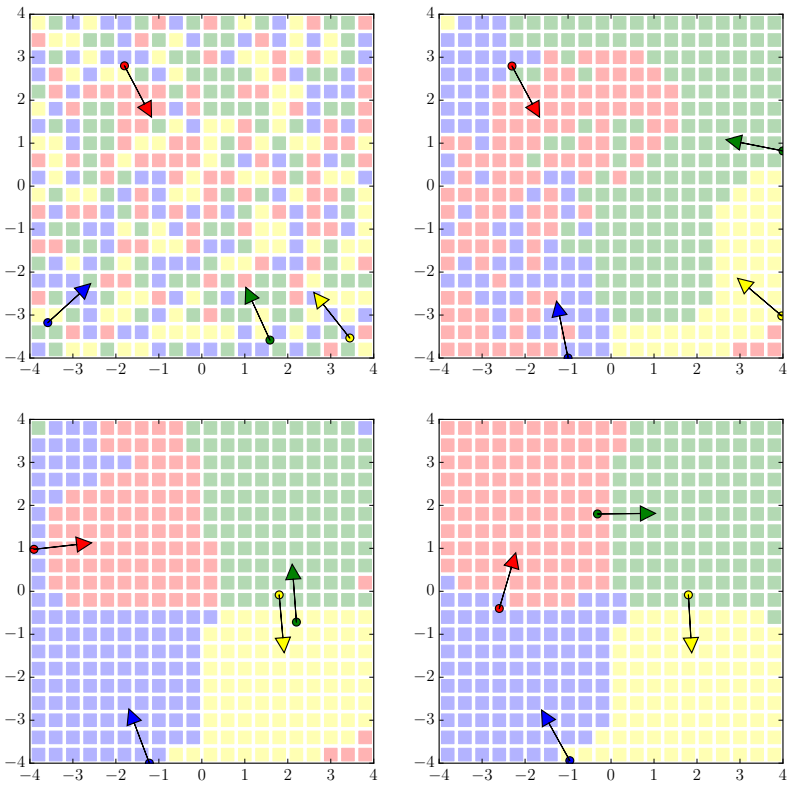


Figure 5.2. Results of the simulation described in Section 5.8. Each agent is represented as a colored arrow, and each landmark is represented as a tile of the same color of the agent that is it currently assigned to. Top left: initial configuration, $\mathrm{cov}(\mathcal{S}, \mathcal{P}) \simeq 314$. Top right: approximately one third of the simulation time elapsed. Bottom left: approximately two thirds of the simulation time elapsed. Bottom right: upon termination, $\mathrm{cov}(\mathcal{S}, \mathcal{P}) \simeq 259$.

5.9 Experimental evaluation of the generalized discrete Lloyd descent

The GDLD has been subject to a preliminary experimental evaluation. The experiments have been conducted in the KTH *Smart Mobility Lab*, with a sensor network comprising one physical aerial robot and three simulated aerial robots. The physical robot is a 3DR IRIS+ quadcopter. Both the physical and the simulated robots are controlled by a position controller running on the Robot Operating System (ROS). The position of the physical robot is measured by means of a motion capture system, while the position of the simulated robots is computed by the simulator, and sent to the controller by ROS. To reproduce the distributed nature of the GDLD, each simulator and each controller is implemented as a separate ROS node, which means that it runs as a separate program thread. In this experiment, the landmarks are represented as colored tiles, which are projected onto the lab floor. We consider 500 landmarks as a discrete abstraction of a 5.0 by 4.0 meter rectangular domain. For practical purposes, the robots have heterogeneous optimization domains. The simulated robots can optimize their pose within the whole rectangular domain, while the real robot is constrained to move within the flying arena, at a reasonable distance from the bounding net, which leads to a 2.2 by 1.0 meter optimization domain. Note that the coverage mission depends on the physical motion of the real robot. In fact, when one of the robots executes an optimization routine, it is given the resulting pose A_i^* as a waypoint, and it needs to navigate to that pose before executing the following steps in the algorithm. Figures 5.3 and 5.4 illustrate four snapshots from one of the experiments that have been performed.

5.10 Gradient descent for coverage improvement

So far in the chapter, we have not taken into account the physical motion of the sensors. In this section, we study how to improve the coverage attained by a mobile sensor by controlling its position and orientation by (5.2) and (5.3). In this context, motion planning is executed continuously, rather than intermittently, by each sensor, while communication is still a discrete event. We need the following Assumption 5.4.

Assumption 5.4. *The sensor footprints are continuously differentiable in all their arguments.*

Under Assumption 5.4, we can write

$$\frac{\mathrm{d}}{\mathrm{d}t} f(p, \hat{n}, q, \hat{m}) = \frac{\partial}{\partial p} f(p, \hat{n}, q, \hat{m})^\top \frac{\mathrm{d}p}{\mathrm{d}t} + \frac{\partial}{\partial \hat{n}} f(p, \hat{n}, q, \hat{m})^\top \frac{\mathrm{d}\hat{n}}{\mathrm{d}t}, \tag{5.12}$$

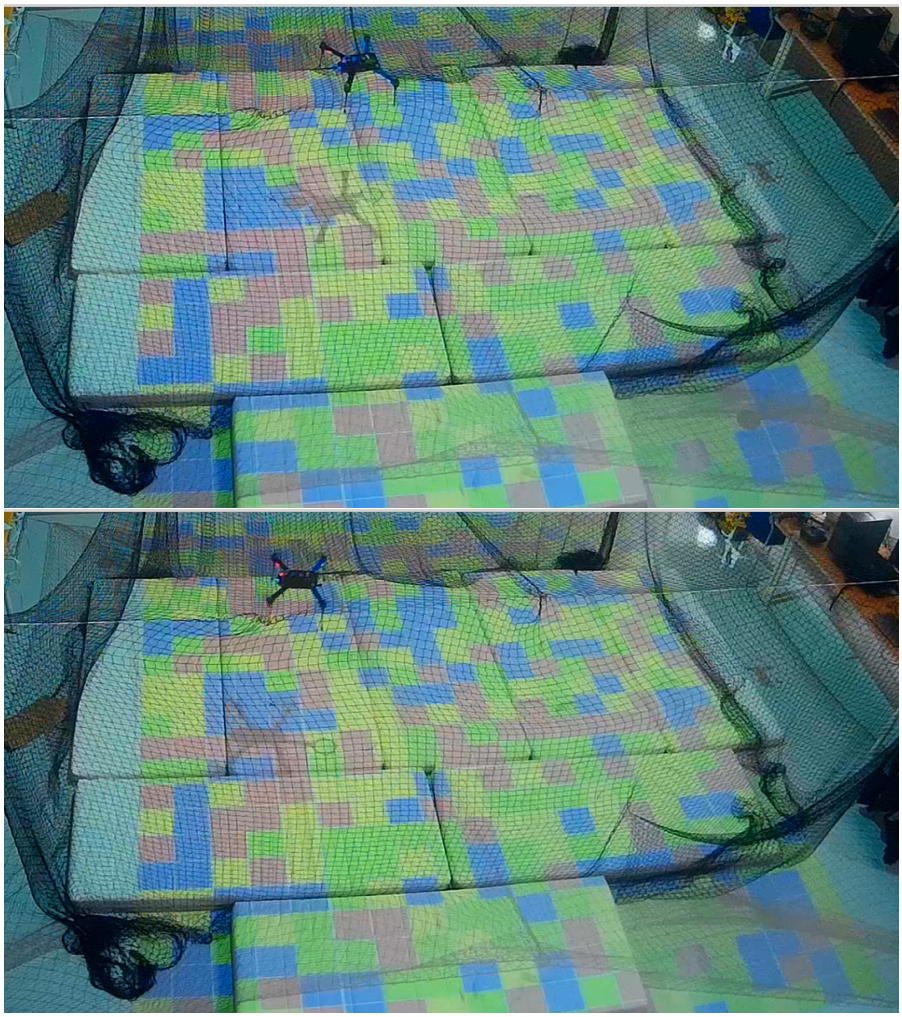


Figure 5.3. Results of the preliminary experimental evaluation of the GDLD described in Section 5.9. The physical robot is responsible for the green landmarks. The rounded arrows represent the positions and orientations generated by the optimization procedures in the coverage algorithm, and constitute reference waypoints for the robots. First part of the experiment; top: start of the experiment; bottom: 18 seconds elapsed.

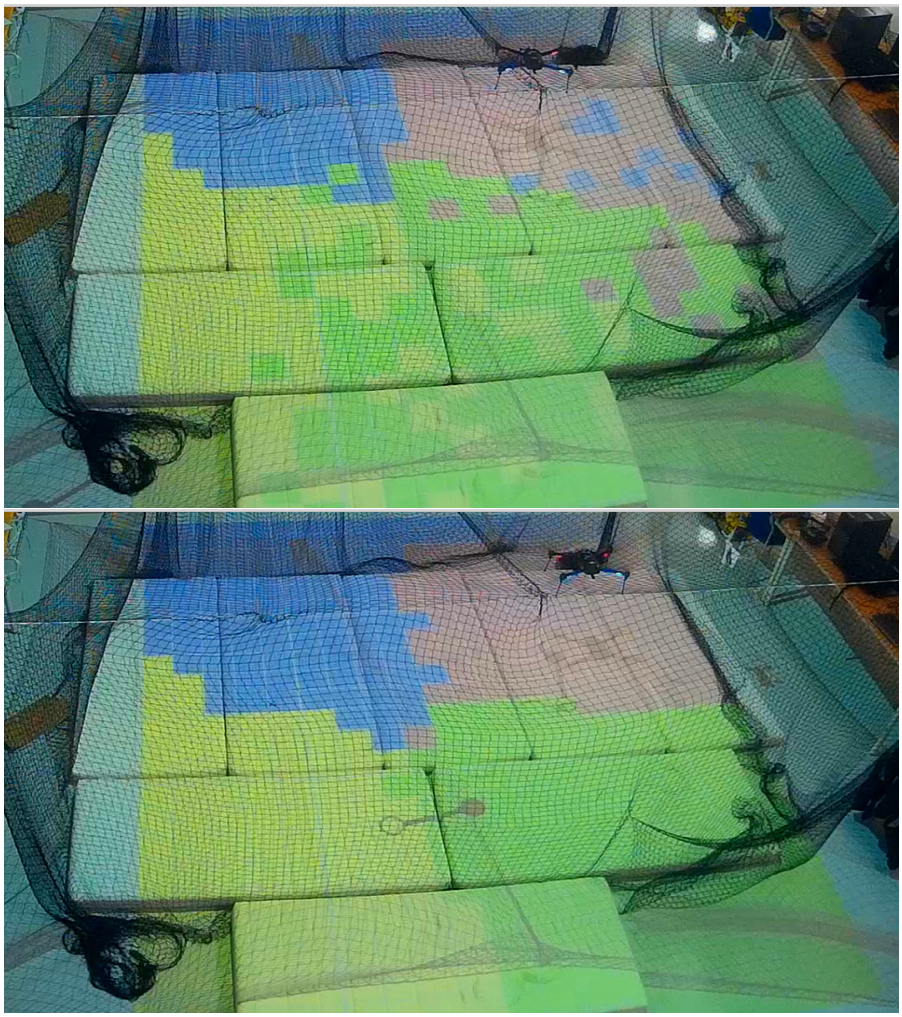


Figure 5.4. Results of the preliminary evaluation of the GDLD described in Section 5.9. The physical robot is responsible for the green landmarks. The rounded arrows represent the positions and orientations generated by the optimization procedures in the coverage algorithm, and constitute reference waypoints for the robots. Second part of the experiment; top: 80 seconds elapsed; bottom: 180 seconds elapsed.

where the partial derivatives $\frac{\partial}{\partial p} f(p, \hat{n}, q, \hat{m})$ and $\frac{\partial}{\partial \hat{n}} f(p, \hat{n}, q, \hat{m})$ are continuous. Consider now a mobile sensor $s = (p, \hat{n})$ and a finite set of landmarks $\mathcal{L}$. Using (5.2)–(5.5) and (5.12), we have

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t} \operatorname{cov}(s, \mathcal{L}) = & \left(\sum_{(q,\hat{m}) \in \mathcal{L}} \frac{\partial}{\partial p} f(p, \hat{n}, q, \hat{m}) \right)^{\top} v \\ & + \left(\sum_{(q,\hat{m}) \in \mathcal{L}} \frac{\partial}{\partial \hat{n}} f(p, \hat{n}, q, \hat{m}) \right)^{\top} \operatorname{skew}(\hat{n}) \omega. \end{aligned} \tag{5.13}$$

From (5.13), we have a natural control strategy to improve the coverage of $\mathcal{L}$ attained by s. Namely, we can choose

$$v = - \sum_{(q,\hat{m}) \in \mathcal{L}} \frac{\partial}{\partial p} f(p, \hat{n}, q, \hat{m}), \tag{5.14}$$

$$\omega = \operatorname{skew}(\hat{n}) \sum_{(q,\hat{m}) \in \mathcal{L}} \frac{\partial}{\partial \hat{n}} f(p, \hat{n}, q, \hat{m}). \tag{5.15}$$

Substituting (5.14) and (5.15) into (5.13), we have

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t} \operatorname{cov}(s, \mathcal{L}) = & - \left\| \sum_{(q,\hat{m}) \in \mathcal{L}} \frac{\partial}{\partial p} f(p, \hat{n}, q, \hat{m}) \right\|^2 \\ & - \left\| \operatorname{skew}(\hat{n}) \sum_{(q,\hat{m}) \in \mathcal{L}} \frac{\partial}{\partial \hat{n}} f(p, \hat{n}, q, \hat{m}) \right\|^2, \end{aligned}$$

which implies

$$\frac{\mathrm{d}}{\mathrm{d}t} \operatorname{cov}(s, \mathcal{L}) \leq 0 \tag{5.16}$$

and

$$\frac{\mathrm{d}}{\mathrm{d}t} \operatorname{cov}(s, \mathcal{L}) = 0 \iff \begin{cases} \sum_{(q,\hat{m}) \in \mathcal{L}} \frac{\partial}{\partial p} f(p, \hat{n}, q, \hat{m}) = 0_3, \\ \sum_{(q,\hat{m}) \in \mathcal{L}} \frac{\partial}{\partial \hat{n}} f(p, \hat{n}, q, \hat{m}) \in \operatorname{Null}(\operatorname{skew}(\hat{n})). \end{cases} \tag{5.17}$$

The monotonicity of $\operatorname{cov}(s, \mathcal{L})$ suggests to use LaSalle's Invariance Principle to deduce the stability of the motion of the sensor under the control law (5.14) and (5.15). This reasoning is formalized in the following Theorem 5.2.

Theorem 5.2. *Consider a mobile sensor $s = (p, \hat{n}, f)$ and a finite set of landmarks $\mathcal{L}$. Under the control law* (5.14) *and* (5.15)*, the coverage of $\mathcal{L}$ attained by s is strictly*

decreasing, and the sensor converges to a position and an orientation $(p^*, \hat{n}^*)$ *such that*

$$\sum_{(q,\hat{m})\in\mathcal{L}} \frac{\partial}{\partial p} f(p^*, \hat{n}^*, q, \hat{m}) = 0_3, \tag{5.18}$$

$$\sum_{(q,\hat{m})\in\mathcal{L}} \frac{\partial}{\partial \hat{n}} f(p^*, \hat{n}^*, q, \hat{m}) \in \mathrm{Null}(\mathrm{skew}(\hat{n}^*)). \tag{5.19}$$

Proof. Consider the function $\mathrm{cov}(s, \mathcal{L})$. Recall that $\mathrm{cov}(s, \mathcal{L})$ is a nonnegative function of p and $\hat{n}$. Moreover, under the control law (5.14) and (5.15), we have (5.16). Therefore, by LaSalle's invariance principle, the system (5.2) and (5.3) converges to the largest invariant set contained within the set

$$\mathcal{I} = \left\{ (p, \hat{n}) : \frac{\mathrm{d}}{\mathrm{d}t} \mathrm{cov}((p, \hat{n}, f), \mathcal{L}) = 0 \right\}.$$

Using (5.14), (5.15) and (5.17), we can see that $v = 0$ and $\omega = 0$ for all $(p, \hat{n}) \in \mathcal{I}$. Therefore, the set $\mathcal{I}$ is itself invariant, and the system converges to $\mathcal{I}$. Using (5.17) again, we can see that, in $\mathcal{I}$, (5.18) and (5.19) hold. Moreover, since $v = 0$ and $\omega = 0$ for all $(p, \hat{n}) \in \mathcal{I}$, the sensor is not moving in $\mathcal{I}$, which means that the system converges to a single point $(p^*, \hat{n}^*)$ in $\mathcal{I}$. □

Control law (5.14) and (5.15) may be used instead of the `optCov` routine in the GDLD. This choice has the disadvantage that, depending on the sensor footprint, the sensor may end up in a local minimum $(p, \hat{n})$ of the objective function $\mathrm{cov}((p, \hat{n}, f), \mathcal{L})$. However, applying (5.14) and (5.15) presents at least two advantages. First, the coverage attained by the sensor is monotonically increasing during the optimization, because the physical motion of the sensor follows the gradient of $\mathrm{cov}(s, \mathcal{L})$. Moreover, the sensor is not required to run a complex optimization routine on board.

5.11 Simulation of the gradient descent for coverage improvement

In this section, we present a simulation of control law (5.14) and (5.15). Here, we consider the scenario where a camera sensor has to inspect the interior of a surface, such as part of a wall and/or a ceiling. The surface to inspect is modelled as a compact surface $\Theta \subseteq \mathbb{R}^3$. For this simulation, we choose the spherical sector defined by

$$\Theta = \left\{ R(\cos\theta \sin\phi, \sin\theta \sin\phi, \cos\phi)^\top : \theta \in \left[0, \frac{\pi}{4}\right], \phi \in \left[0, \frac{\pi}{4}\right] \right\},$$

with $R = 3$. The surface is abstracted into a set of $M = 100$ landmarks. Each landmark represents a facet of interest on the surface. Namely, for each landmark $\ell = (q, \hat{m})$, $q \in \Theta$ is a point on the surface, and $\hat{m}$ is the outward normal to the surface in that point. In this scenario, since the surface is a sphere sector, we have $\hat{m} = q/R$.

The camera sensor is modelled as $s = (p, \hat{n}, f)$, with the following footprint:

$$f(p, \hat{n}, q, \hat{m}) = \|p + \hat{n} - q\|^2 + \|p + \hat{m} - q\|^2. \tag{5.20}$$

Footprint (5.20) captures the idea that a facet $(q, \hat{m})$ of the surface is best seen by the camera when the facet lays in front of the camera ($q = p + \hat{n}$), and the outward normal to the surface is aligned with the sight line of the camera ($\hat{n} = \hat{m}$). In this respect, footprint (5.20) is similar to footprint (5.11). However, footprint (5.20) is continuously differentiable, therefore it satisfies Assumption 5.4. Figure 5.5 illustrates a contour plot of footprint (5.20) for $p = 0_3$, $\hat{n} = (1, 0, 0)$, $q = (q_x, q_y, 0)$, and $\hat{m} = \hat{n}$, as a function of q_x and q_y. As we can predict from (5.20), the level curves are circles centered in $q = p + \hat{n} = (1, 0, 0)$. The results of the simulation are illustrated in Figure 5.6. From Figure 5.6, we can see that the sensor positions itself in front of the surface under inspection at a suitable distance, and aligns its sight line to agree with the curvature of the surface.

5.12 Summary

In this chapter, we have discussed a general framework to address coverage and inspection problems for networks of heterogeneous anisotropic sensors. First, we have defined an abstraction of a mobile sensor and of the environment under inspection. Then, we have used these models to formulate the coverage task as an optimization problem. Thereafter, we have proposed a distributed algorithm, the GDLD, to address the formulated problem and achieve a local minimum in an efficient way. The convergence properties of the proposed algorithm have been demonstrated analytically and with simulations. Finally, we have proposed a control law for the motion of a single sensor that improves the coverage attained by that sensor monotonically. The convergence properties of the proposed control law have been demonstrated analytically and on simulation, and we have also discussed how the proposed control law can be used as a subroutine within the GDLD.

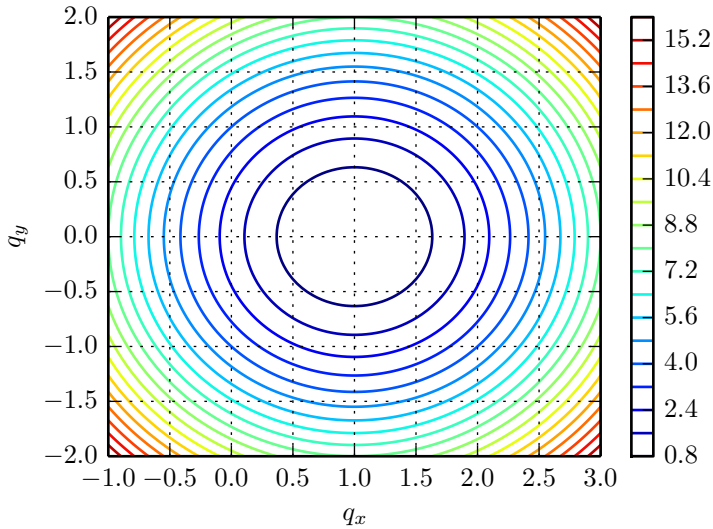


Figure 5.5. Contour plot of footprint (5.20) for $p = 0_3$, $\hat{n} = (1, 0, 0)$, $q = (q_x, q_y, 0)$, and $\hat{m} = \hat{n}$, as a function of q_x and q_y.

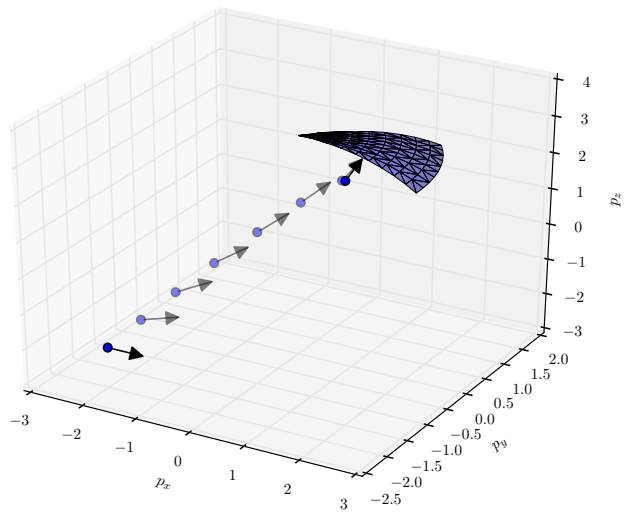


Figure 5.6. Results of the simulation presented in Section 5.11. The sensor $s = (p, \hat{n}, f)$ is represented as an arrow, where the tail is located in p and the tip is pointing to the direction $\hat{n}$. The solid arrows represent the initial and the final configurations, while the shaded arrows represent some intermediate configurations. The shaded surface represents the surface that the sensor has to inspect, and it is obtained by interpolating the landmarks.

Chapter 6

Conclusions and future research

As technology advances, networked systems become more common and more pervasive, and, at the same time, more complex and more challenging to analyze and control. When controlling a networked system, the aim is usually to make a desired collective behavior arise from the individual dynamics of each agent in the system and from local interactions among the agents. In most realistic applications, the communication between one agent and the rest of the network is intermittent, rather than continuous. In this thesis, we have studied several control strategies for networked systems based on intermittent, event-driven communication. In this chapter, we summarize our results and discuss possible directions for future research.

6.1 Conclusions

Event-triggered pinning control of switching networks

We derived a decentralized event-based protocol for pinning control of networks of nonlinear systems with switching topologies. The protocol prescribes piecewise constant control signals. Each agent in the network updates its control signal when a given condition regarding the state of its neighborhood is verified. Namely, a function of the state of the agent's neighborhood is compared to a given threshold function. When the state function overcomes the threshold function, a control update is triggered. To avoid continuous inter-agent communication, each agent can estimate the state of its neighborhood with a predictor, and broadcast its control input to its neighbors upon the control updates.

We quantified the pinning controllability of the network in terms of the agents' local dynamics, of the variations in the netowork topology, and of the variations in the number and location of the pinned nodes. We showed that,

if the parameter that quantifies the pinning controllability of the network is large enough with respect to the Lipshitz constant of the agents' dynamics, and to the convergence rate of the threshold function, then the network converges to the reference trajectory asymptotically without exhibiting Zeno behavior in the sequences of the control updates. We showed also that the convergence rate of the threshold function is a lower bound for the convergence rate of the network to the reference trajectory. Hence, the convergence rate of the threshold function can be used as a parameter in the control design. Namely, a larger convergence rate leads to faster tracking, but tends to induce a larger number of control updates. However, as long as the convergence rate of the threshold function is chosen below a value that scales linearly with the pinning controllability of the network, the closed-loop system does not exhibit Zeno behavior.

Cloud-supported multi-agent coordination

We derived a control algorithm for multi-agent networks where inter-agent communication is completely substituted by the exchange of data on a shared repository hosted on a cloud. In the proposed algorithm, each agent schedules its own accesses to the cloud in a recursive fashion, and independently of the other agents. When an agent accesses the cloud, it uses the information that it has downloaded to schedule the next access. The scheduling is based on the comparison of a time-varying function of the data downloaded from the repository with a time-varying threshold function. Namely, when an agent accesses the repository, it schedules the following access at the earliest time when the function of the downloaded data overcomes the threshold function. The convergence rate of the threshold function is a parameter in the control design, and acts as a lower bound on the convergence rate of the closed-loop system. The proposed setup was applied to a rendezvous problem of mobile agents with second-order dynamics subject to disturbances. We showed that the proposed algorithm achieves bounded convergence if the disturbances are persistent, and asymptotic convergence if the disturbances slowly vanish. In both cases, we showed that the sequence of the accesses to the cloud repository by each agent does not exhibit Zeno behavior, as long as the the convergence rate of the threshold function is smaller than a constant that depends on the topology of the information exchange through the cloud.

Coverage control of anisotropic sensor networks

We designed a family of distributed protocols for coverage control of anisotropic sensor networks. The environment or structure that the sensors have to monitor is abstracted into a finite set of landmarks, where each landmark is the

abstraction of a point or a small area of interest. Each sensor is initially assigned a subset of the landmarks to monitor. To improve the coverage of the environment, each sensor can change its position and orientation, but also transfer the ownership of some of its landmarks to another sensor that has a better perception of those landmarks. Communication between sensors is intermittent and event-based. We showed that the proposed protocol is deadlock-free and leads to monotonic improvement of the global coverage achieved by the sensor network. We showed also that the sensor network converges to a configuration that can be seen as a generalized version of a Voronoi tessellation. The convergence properties of the proposed algorithms have been discussed in terms of convexity and Lipschitzianity of the sensors' footprints. The protocol can easily handle the addition of new sensor and landmarks to the network.

6.2 Future research

There exist many challenging open questions in the topic of multi-agent systems. In this section, we discuss some natural extensions of the work presented in this thesis.

Event-triggered pinning control of switching networks

When dealing with agreement protocols in multi-agent networks, one always needs some assumptions on the connectedness of the network topology. For example, the assumption that we make in our event-triggered control algorithm essentially means that the network must be connected to the reference trajectory sufficiently often. However, we showed that these assumptions are sufficient conditions for asymptotic convergence of the network to the reference trajectory, but not that they are necessary. It would be interesting to study if the same event-triggered protocol, or a similar one, can be used for pinning control of networks with even less conservative connectivity assumptions, such as jointly connected networks.

Other possible extensions include considering networks with directed topologies and more general control protocols.

Cloud-supported multi-agent coordination

The idea of substituting inter-agent communication with the access to a remote repository can be applied in many different multi-agent control tasks. In this thesis, we presented a general framework for cloud-supported multi-agent coordination, but we derived a control law and access scheduling rule only for the case when the objective is the rendezvous of second-order agents.

Future research will address more general coordination objectives for the proposed framework.

Another research challenge arises if we attribute a cost to each cloud access, and we try to account for such cost explicitly in the control design. For example, one may consider a cost function inspired to LQR control, and incorporate an additional impulsive term to penalize the cloud accesses of each agent.

Coverage control of anisotropic sensor networks

Coverage control is widely open to improvements. The existing algorithms based on Voronoi tessellations converge to a local optimum of the cost function that represents the coverage achieved by the sensor network, and the research ground is open for solutions that try to avoid this type of equilibria.

Depending on the sensors' footprints and on the control algorithm, a single sensor may also converge to a local optimum of the cost function that represents the coverage achieved by that sensor. Future research will focus on a footprint design that allows a sensor to avoid the local optima of its own coverage function. Since a footprint is a function of both the position and the orientation of the sensor, the design of a footprint that escapes local optima will relate the concepts of standard convexity and geodesic convexity.

For most practical applications, it is important to incorporate a collision avoidance scheme to coverage control. Collision avoidance is achieved as a side effect in Voronoi-based coverage control for sensors with omnidirectional footprints, but this benefit does not extend to networks of sensor with general footprints. A research challenge is to design a collision avoidance scheme which preserves as many properties as possible of the original control algorithm, such as monotonic improvement of the global coverage function, intermittent communication, and guaranteed absence of deadlocks.

Bibliography

[1] M. E. J. Newman, *Networks: an introduction*. Oxford University Press, 2010.

[2] M. Mesbahi and M. Egerstedt, *Graph Theoretic Methods in Multiagent Networks*. Princeton Univerisity Press, 2010.

[3] S. H. Strogatz, "Exploring Complex Networks," *Nature*, vol. 410, pp. 268–276, 2001.

[4] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[5] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Transactions on Automatic Control*, vol. 51, no. 3, pp. 401–420, 2006.

[6] M. Andreasson, D. V. Dimarogonas, H. Sandberg, and K. H. Johansson, "Distributed control of networked dynamical systems: Static feedback, integral action and consensus," *IEEE Transactions on Automatic Control*, vol. 59, no. 7, pp. 1750–1764, 2014.

[7] R. O. Grigoriev, M. C. Cross, and H. G. Schuster, "Pinning Control of Spatiotemporal Chaos," *Physical Review Letters*, vol. 79, no. 15, 1997.

[8] X. F. Wang and G. Chen, "Pinning control of scale-free dynamical networks," *Physica A*, vol. 310, pp. 521–531, 2002.

[9] X. Li, X. Wang, and G. Chen, "Pinning a Complex Dynamical Network to Its Equilibrium," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 51, no. 10, pp. 2074–2087, 2004.

[10] F. Sorrentino, M. Di Bernardo, F. Garofalo, and G. Chen, "Controllability of complex networks via pinning," *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, vol. 75, no. 4, 2007.

[11] M. Porfiri and M. di Bernardo, "Criteria for global pinning-controllability of complex networks," *Automatica*, vol. 44, no. 12, pp. 3100–3106, 2008.

[12] Q. Song, J. Cao, and W. Yu, "Second-order leader-following consensus of nonlinear multi-agent systems via adaptive pinning control," *Systems & Control Letterss*, vol. 59, pp. 553–562, 2010.

[13] W. Wu, W. Zhou, and T. Chen, "Cluster Synchronization of Linearly Coupled Complex Networks Under Pinning Control," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 56, no. 4, pp. 829–839, 2009.

[14] P. Delellis, M. Di Bernardo, and M. Porfiri, "Pinning control of complex networks via edge snapping," *Chaos*, vol. 21, no. 19, 2011.

[15] Y.-Y. Liu, J.-J. Slotine, and A.-L. Barabási, "Controllability of complex networks," *Nature*, no. 473, pp. 167–173, 2011.

[16] W. P. M. H. Heemels, K. H. Johansson, and P. Tabuada, "An Introduction to Event-triggered and Self-triggered Control," in *IEEE Conference on Decision and Control*, 2012.

[17] P. Tabuada, "Event-triggered real-time scheduling of stabilizing control tasks," *IEEE Transactions on Automatic Control*, vol. 52, no. 9, pp. 1680–1685, 2007.

[18] A. Anta and P. Tabuada, "To sample or not to sample: Self-triggered control for nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 55, no. 9, pp. 2030–2042, 2010.

[19] D. V. Dimarogonas and K. H. Johansson, "Event-Triggered Control for Multi-Agent Systems," in *IEEE Conference on Decision and Control*, 2009.

[20] G. S. Seyboth, D. V. Dimarogonas, and K. H. Johansson, "Control of Multi-Agent Systems via Event-based Communication," in *IFAC World Congress*, 2011.

[21] G. S. Seyboth, D. V. Dimarogonas, and K. H. Johansson, "Event-based broadcasting for multi-agent average consensus," *Automatica*, vol. 49, pp. 245–252, 2013.

[22] X. Wang and M. D. Lemmon, "Event-triggered broadcasting across distributed networked control systems," in *American Control Conference*, pp. 3139–3144, 2008.

[23] D. V. Dimarogonas, E. Frazzoli, and K. H. Johansson, "Distributed Self-Triggered Control for Multi-Agent Systems," in *IEEE Conference on Decision and Control*, 2010.

[24] D. V. Dimarogonas, E. Frazzoli, and K. H. Johansson, "Distributed event-triggered control for multi-agent systems," *IEEE Transactions on Automatic Control*, vol. 57, no. 5, pp. 1291–1297, 2012.

[25] G. Shi and K. H. Johansson, "Multi-agent Robust Consensus - Part II: Application to Distributed Event-triggered Coordination," *IEEE Conference on Decision and Control and European Control Conference*, pp. 5738–5743, 2011.

[26] X. Meng and T. Chen, "Event based agreement protocols for multi-agent networks," *Automatica*, vol. 49, pp. 2125–2132, 2013.

[27] W. Zhu, Z.-P. Jiang, and G. Feng, "Event-based consensus of multi-agent systems with general linear models," *Automatica*, vol. 50, pp. 552–558, 2014.

[28] S. Dormido, D. V. Dimarogonas, J. Sánchez, K. H. Johansson, and M. Guinaldo, "Distributed event-based control strategies for interconnected linear systems," *IET Control Theory & Applications*, vol. 7, no. 6, pp. 877–886, 2013.

[29] H. Li, X. Liao, T. Huang, and W. Zhu, "Event-Triggering Sampling Based Leader-Following Consensus in Second-Order Multi-Agent Systems," *IEEE Transactions on Automatic Control*, vol. 60, no. 7, pp. 1998–2003, 2015.

[30] C. De Persis and P. Frasca, "Robust self-triggered coordination with ternary controllers," *IEEE Transactions on Automatic Control*, vol. 58, no. 12, pp. 3024–3038, 2013.

[31] F. Hao and X. Chen, "Event-triggered average consensus control for discrete-time multi-agent systems," *IET Control Theory & Applications*, vol. 6, no. February, pp. 2493–2498, 2012.

[32] A. Eqtami, D. V. Dimarogonas, and K. J. Kyriakopoulos, "Event-Based Model Predictive Control for the Cooperation of Distributed Agents," in *American Control Conference*, pp. 6473–6478, 2012.

[33] K. Hashimoto, S. Adachi, and D. V. Dimarogonas, "Distributed aperiodic model predictive control for multi-agent systems," *IET Control Theory & Applications*, vol. 9, no. 1, pp. 10–20, 2015.

[34] D. Liuzza, D. V. Dimarogonas, M. di Bernardo, and K. H. Johansson, "Distributed Model Based Event-Triggered Control for Synchronization of Multi-Agent Systems," in *IFAC Symposium on Nonlinear Control Systems*, (Toulose, France), pp. 329–334, 2013.

[35] D. B. Edwards, T. A. Bean, D. L. Odell, and M. J. Anderson, "A leader-follower algorithm for multiple AUV formations," in *IEEE/OES Autonomous Underwater Vehicles*, pp. 40–46, 2004.

[36] P. V. Teixeira, D. V. Dimarogonas, K. H. Johansson, and J. Sousa, "Event-based motion coordination of multiple underwater vehicles under disturbances," in *IEEE OCEANS*, 2010.

[37] P. V. Teixeira, D. V. Dimarogonas, and K. H. Johansson, "Multi-agent coordination with event-based communication," in *American Control Conference*, (Marriot Waterfront, Baltimore, MD, USA), 2011.

[38] D. J. Stilwell and B. E. Bishop, "Platoons of underwater vehicles," *IEEE Control Systems Magazine*, vol. 20, no. 6, pp. 45–52, 2000.

[39] E. Fiorelli, N. E. Leonard, P. Bhatta, D. A. Paley, R. Bachmayer, and D. M. Fratantoni, "Multi-AUV control and adaptive sampling in Monterey Bay," *IEEE Journal of Oceanic Engineering*, vol. 31, no. 4, pp. 935–948, 2006.

[40] J. Cortés, S. Martínez, T. Karata, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[41] S. Martinez, J. Cortes, and F. Bullo, "Motion Coordination with Distributed Information," *IEEE Control Systems Magazine*, vol. 27, no. 4, pp. 75–88, 2007.

[42] M. Zhong and C. G. Cassandras, "Distributed Coverage Control and Data Collection with Mobile Sensor Networks," *IEEE Transactions on Automatic Control*, vol. 56, no. 10, pp. 2445–2455, 2011.

[43] J. Le Ny and G. J. Pappas, "Adaptive Deployment of Mobile Robotic Networks," *IEEE Transactions on Automatic Control*, vol. 58, no. 3, pp. 654–666, 2013.

[44] Q. Du, V. Faber, and M. Gunzburger, "Centroidal Voronoi Tessellations: Applications and Algorithms," *SIAM Review*, vol. 41, no. 4, pp. 637–676, 1999.

[45] Y. Stergiopoulos and A. Tzes, "Spatially distributed area coverage optimisation in mobile robotic networks with arbitrary convex anisotropic patterns," *Automatica*, vol. 49, no. 1, pp. 232–237, 2013.

[46] Y. Stergiopoulos and A. Tzes, "Cooperative positioning/orientation control of mobile heterogeneous anisotropic sensor networks for area coverage," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[47] A. Gusrialdi, S. Hirche, T. Hatanaka, and M. Fujita, "Voronoi based coverage control with anisotropic sensors," in *American Control Conference*, pp. 736–741, 2008.

[48] K. Laventall and J. Cortes, "Coverage control by multi-robot networks with limited-range anisotropic sensory," in *American Control Conference*, 2008.

[49] A. Gusrialdi, T. Hatanaka, and M. Fujita, "Coverage Control for Mobile Networks with Limited-Range Anisotropic Sensors," in *IEEE Conference on Decision and Control (CDC)*, pp. 4263–4268, 2008.

[50] J.-S. Marier, C.-A. Rabbath, and N. Lechevin, "Visibility-limited Coverage Control Using Nonsmooth Optimization," in *American Control Conference (ACC)*, 2012.

[51] Y. Kantaros, M. Thanou, and A. Tzes, "Visibility-oriented Coverage Control of Mobile Robotic Networks on Non-convex Regions," in *IEEE International Conference on Robotics and Automation*, 2014.

[52] Y. Kantaros, M. Thanou, and A. Tzes, "Distributed coverage control for concave areas by a heterogeneous Robot-Swarm with visibility sensing constraints," *Automatica*, vol. 53, pp. 195–207, 2015.

[53] M. Thanou, Y. Stergiopoulos, and A. Tzes, "Distributed coverage using geodesic metric for non-convex environments," in *IEEE International Conference on Robotics and Automation*, pp. 933–938, 2013.

[54] Y. Stergiopoulos, M. Thanou, and A. Tzes, "Distributed Collaborative Coverage-Control Schemes for Non-Convex Domains," *IEEE Transactions on Automatic Control*, vol. 60, no. 9, pp. 2422–2427, 2015.

[55] I. I. Hussein and D. M. Stipanovic, "Effective Coverage Control using Dynamic Sensor Networks," in *IEEE Conference on Decision and Control (CDC)*, 2006.

[56] I. I. Hussein and D. M. Stipanovic, "Effective Coverage Control for Mobile Sensor Networks With Guaranteed Collision Avoidance," *IEEE Transactions on Control Systems Technology*, vol. 15, no. 4, pp. 642–657, 2007.

[57] D. Panagou, D. M. Stipanovic, and P. G. Voulgaris, "Vision-based dynamic coverage control for nonholonomic agents," in *IEEE Conference on Decision and Control (CDC)*, 2014.

[58] D. Panagou, D. M. Stipanovic, and P. G. Voulgaris, "Dynamic coverage control in unicycle multi-robot networks under anisotropic sensing," *Frontiers in Robotics and AI*, vol. 2, pp. 1–17, 2015.

[59] P. Frasca, R. Carli, and F. Bullo, "Multiagent coverage algorithms with gossip communication: control systems on the space of partitions," in *American Control Conference (ACC)*, 2009.

[60] F. Bullo, R. Carli, and P. Frasca, "Gossip Coverage Control for Robotic Networks: Dynamical Systems on the Space of Partitions," *SIAM Journal on Control and Optimization*, vol. 50, no. 1, pp. 419–447, 2012.

[61] J. W. Durham, R. Carli, and F. Bullo, "Pairwise Optimal Discrete Coverage Control for Gossiping Robots," in *IEEE Conference on Decision and Control (CDC)*, 2010.

[62] J. W. Durham, R. Carli, P. Frasca, and F. Bullo, "Discrete Partitioning and Coverage Control for Gossiping Robots," *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 364–378, 2012.

[63] J. W. Durham, R. Carli, P. Frasca, and F. Bullo, "Dynamic partitioning and coverage control with asynchronous one-to-base-station communication," *IEEE Transactions on Control of Network Systems*, vol. 3, no. 1, pp. 24–33, 2016.

[64] A. Adaldo, F. Alderisio, D. Liuzza, G. Shi, D. V. Dimarogonas, M. di Bernardo, and K. H. Johansson, "Event-triggered pinning control of complex networks with switching topologies," in *IEEE Conference on Decision and Control*, pp. 2783–2788, 2014.

[65] A. Adaldo, F. Alderisio, D. Liuzza, G. Shi, D. V. Dimarogonas, M. di Bernardo, and K. H. Johansson, "Event-triggered pinning control of switching networks," *IEEE Transactions on Control of Network Systems*, vol. 2, no. 2, pp. 204–213, 2015.

[66] A. Adaldo, D. Liuzza, D. V. Dimarogonas, and K. H. Johansson, "Control of multi-agent systems with event-triggered cloud access," in *European Control Conference*, 2015.

[67] A. Adaldo, D. Liuzza, D. V. Dimarogonas, and K. H. Johansson, "Multi-agent trajectory tracking with self-triggered cloud access," *Accepted for publication in the IEEE Conference on Decision and Control*, 2016.

[68] A. Adaldo, D. Liuzza, D. V. Dimarogonas, and K. H. Johansson, "Cloud-supported coordination of second-order multi-agent systems," *Submitted to the IEEE Transactions on Control of Network Systems*.

[69] A. Adaldo, D. V. Dimarogonas, and K. H. Johansson, "Discrete partitioning and intermittent communication for anisotropic coverage and inspection missions," *To be submitted to the 2017 World Congress of the International Federation of Automatic Control (IFAC)*.

[70] H. Eves, *Elementary Matrix Theory*. Dover Publications, 1996.

[71] Z. Zeng, X. Wang, and Z. Zheng, "Convergence analysis using the edge Laplacian: Robust consensus of nonlinear multi-agent systems via ISS method," *International Journal of Robust and Nonlinear Control*, vol. 26, pp. 1051–1072, 2015.

[72] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge University Press, 2nd ed., 2012.

[73] K. H. Johansson, M. Egerstedt, J. Lygeros, and S. Sastry, "On the regularization of Zeno hybrid automata," *Systems & Control Letters*, vol. 38, pp. 141–150, 1999.

[74] H. Khalil, *Nonlinear Systems*. Prentice Hall, second ed., 2002.

[75] T. Matsumoto, "A Chaotic Attractor from Chua's Circuit," *IEEE Transactions on Circuits and Systems*, vol. 31, no. 12, pp. 1055–1058, 1984.

[76] L. Paull, S. Saeedi, M. Seto, and H. Li, "AUV navigation and localization: A review," *IEEE Journal of Oceanic Engineering*, vol. 39, no. 1, pp. 131–149, 2014.

[77] M. T. Hale and M. Egerstedt, "Differentially private cloud-based multi-agent optimization with constraints," in *Proceedings of the American Control Conference*, (Chicago, IL, USA), 2015.

[78] C. Nowzari and G. J. Pappas, "Multi-agent coordination with asynchronous cloud access," in *American Control Conference*, 2016.

[79] D. Zelazo and M. Mesbahi, "Edge agreement: Graph-theoretic performance bounds and passivity analysis," *IEEE Transactions on Automatic Control*, vol. 56, no. 3, pp. 544–555, 2011.

[80] K. Liu, G. Xie, and L. Wang, "Consensus for second-order multi-agent systems with inherent nonlinear dynamics under directed topologies," *Systems & Control Letters*, vol. 62, 2013.

[81] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[82] J. D. Pintér, *Global Optimization in Action*. Kluwer Academic Publishers, 1996.

[83] A. B. Downey, *The Little Book of Semaphores*. Green Tea Press, 2008.

[84] K. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Springer, 2005.